Hindawi
Security and Communication Networks
Volume 2022, Article ID 1955141, 13 pages
https://doi.org/10.1155/2022/1955141

Research Article

A Novel Semifragile Consensus Algorithm Based on Credit Space for Consortium Blockchain

Xiaohong Deng,[1,2,3] **Zhiqiong Luo**,[2] **Yijie Zou**,[2] **Kangting Li**,[2] **and Huiwen Liu**[1]

[1]*School of Electronics and Information Engineering, Gannan University of Science and Technology, Ganzhou 341000, China*
[2]*School of Information Engineering, Jiangxi University of Science and Technology, Ganzhou 341000, China*
[3]*Key Laboratory of Cloud Computing and Big Data, Ganzhou 341000, China*

Correspondence should be addressed to Zhiqiong Luo; 6720200799@mail.jxust.edu.cn

Received 24 January 2022; Accepted 28 March 2022; Published 18 April 2022

Academic Editor: Jiewu Leng

Nowadays, blockchain is known as a new generation of secure information technologies for realizing business and industrial sustainability, and consensus algorithm is the key technology of blockchain. In order to solve the problem of "oligarchy" nodes and excessive punishment for nodes in existing credit consensus algorithms, a novel semifragile consensus algorithm based on the credit space for consortium blockchain is proposed in this paper. Firstly, the accounting node selection mechanism based on credit space is proposed. The credit value of the node is calculated according to a novel credit evaluation model, and then the credit space of the node is allocated according to the size of the credit value. Afterward, a random algorithm is used to select the accounting node in the credit space. This mechanism effectively inhibits the generation of "oligarchy" nodes and maintains the enthusiasm of nodes. Secondly, this paper proposes a semifragile hierarchical punishment mechanism, which punishes the malicious nodes with severe measures and gives the nonmalicious nodes the opportunity to continue participating in the consensus. So, this semifragile punishment mechanism solves the problem of excessive punishment of nodes. Experimental simulation results demonstrate that the proposed consensus algorithm has randomness while maintaining the credit incentive among nodes. In addition, the node's punishment mechanism is more reasonable. This algorithm has better security and can be well applied to consortium blockchain scenarios.

1. Introduction

In 2008, Satoshi Nakamoto publicly published Bitcoin [1]. Afterward, with the crazy of Bitcoin, blockchain as a core technology of Bitcoin has received extensive research attention [2]. Blockchain has the characteristics of decentralization, hard tamperability, traceability, and transparency, which solves the data monopoly and security problems current in the existing centralized platform [3]. At the same time, many studies have found that blockchain has many innovative applications in the field of IoT and sensor networks. For instance, Satapathy et al. [4] proposed a secure architecture based on open blockchain, which can solve some of the challenges in IoT applications, like issues with confidentiality and privacy of data; Mrinal et al. [5] proposed a blockchain-based wireless sensor network for secure vehicle tracking, reducing the need for an Internet connection and eliminating the use of continuous GPS tracking; that is, it can effectively protect the privacy of commuters and the security of collected data. Therefore, blockchain is known as a new generation of secure information technology. As the core part of the blockchain system, the consensus algorithm is the mechanism for each node of the blockchain to reach consensus on the block information of the whole network [6]. More precisely, it can ensure whether the latest block is correctly added to the blockchain. It is worth mentioning that the performance efficiency and security of the entire blockchain system will be affected by the merits of the consensus algorithm [7]. Similarly, consensus algorithm has always been the key technology of decentralized system, which is widely used in resource-constrained edge computing fields. For instance, Zeng et al. [8] proposed a scheme by utilizing the idle resources in volunteer vehicles to handle the overloaded issues in VEC servers; the scheme can reduce the offloading cost of vehicles and improve the utility of VEC servers; Zeng et al. [9] proposed a new vehicle edge computing framework based on software-defined networks, which introduces the reputation to measure the contribution of each vehicle. The proposed scheme not only brings more benefits to the edge server side but also reduces the average delay a lot.

Generally, different blockchain frameworks use different consensus algorithms. In summary, a common classification divides blockchain into three categories, including public blockchain, consortium blockchain, and private blockchain [10]. The number of nodes in the public blockchain is large, so the transaction speed will be slower. On the contrary, there are fewer nodes in private blockchain and consortium blockchain than in public blockchain, and the transaction speed will be faster [11]. However, the permissions in the private blockchain are controlled by a few nodes, which deviates from the original intention of decentralization [12]. Compared with the private blockchain, the permission design requirements in the consortium blockchain are more complicated and more credible. Now, relevant researches show that the consortium blockchain has more practical value in the fields of IoT applications and medical scenarios. For example, Thomas et al. [13] proposed an anonymous identity and access control system based on consortium blockchain, which improves the security of cross-domain identity authentication in the Internet of Things; Huang et al. [14] proposed a medical data privacy protection and safe sharing scheme based on consortium blockchain, which can effectively ensure the safety of patients' medical information and can safely share information.

At present, the consensus algorithm of consortium blockchain is mainly represented by the Practical Byzantine Fault Tolerance (PBFT) protocol [15]. PBFT has a high transaction speed; however, with the number of nodes increasing, the network overhead of PBFT will increase rapidly, and the consumption of computing power will be high [16]. Moreover, PBFT selects the leader node according to the continuous switching of view number, which may select malicious nodes as the leader node, resulting in poor system security [17]. As such, in order to solve the problem of malicious nodes becoming accounting nodes, researchers have proposed a credit mechanism to generate accounting nodes. The credit value was calculated on the basis of the node's performance in the system, and the node with a higher credit value preferentially became the accounting node [18]. For instance, Li et al. [19] proposed a consortium consensus algorithm based on credit (CCAC), which calculated the credit value of nodes by the contribution of node participation consensus, and selected a node to become an accounting node in turn according to the size of node credit. Notably, a consensus mechanism based on credit reduced the consumption of algorithm computing power and improved the efficiency of consensus. However, this is not effective for the node with a small credit value and easily leads to low enthusiasm of nodes. Wang et al. [20] proposed a proof of work algorithm based on credit model (CPoW) and designed a node credit model based on BP neural network, which effectively reduced the huge resource consumption of repeated calculation in the production process of new blocks. Unfortunately, generating new blocks according to the order of credit value was easy to produce "oligarchy" nodes. Li et al. [21] proposed a dynamic hierarchical Byzantine fault-tolerant consensus mechanism based on credit (DHBFT). The presented reward and punishment plan could effectively reduce the possibility of malicious nodes becoming the leader node, but it could easily cause node with high credit values to be selected as the master node, which lacks fairness and easily causes other nodes to be less motivated. Liu et al. [22] proposed a master-slave multichain blockchain consensus mechanism based on reputation, which introduced credit value evaluation into the consensus mechanism based on proof of stake. In addition, it designed a joint consensus mechanism that integrates multiple consensus mechanisms, which improved the throughput of the transaction and ensured the consistency and nontamperability of the data. However, the punishment for all malicious nodes was too heavy, resulting in nodes being unable to normally participate in the consensus for a long time. Bugday et al. [23] proposed a reputation-based consensus group learning model to calculate the credit value based on the weight value of all nodes in the trust committee, which could effectively avoid malicious nodes, but the weight value of malicious nodes is large. Once a node had malicious acts, the credit value of this node would fall to a very low level, and it was difficult to continue to join the consensus. Huang et al. [24] proposed a credit-based proof of work mechanism for IoT devices, which improved security and enhanced transaction efficiency. Similarly, the punishment for malicious nodes was to reduce the credit value directly to a negative value, which made it difficult for nodes to participate in normal consensus.

To sum up, although the existing consensus algorithm based on credit has improved the efficiency and security of consensus, there are still problems that it is easy to generate "oligarchy" nodes and the punishment for nonmalicious nodes is too large. In order to solve the above problems, this paper proposes a semifragile consortium blockchain consensus algorithm based on credit space. The main contributions of this paper are as follows:

(1) An accounting node selection mechanism based on credit space is proposed. A credit evaluation model is formulated to calculate the credit value of the node, and the credit space of the node is allocated based on the credit value. Based on the credit space, an algorithm for randomly selecting accounting nodes is designed. The nodes with large credit space have a high probability of becoming accounting nodes. At the same time, a threshold equation for the number of accounting nodes is set for the problem of "oligarchy" nodes so that the number of times of becoming accounting nodes is limited.

(2) A semifragile hierarchical punishment mechanism is designed. Nodes with good working conditions are

in the *normal* layer, and the nodes with malicious behaviours are placed in the *prison* layer for "custody." Furthermore, we judge whether the node is malicious or nonmalicious; for malicious nodes, the "custody" time will be longer, and for nonmalicious nodes, they can be returned to the *normal* layer beyond the "custody" time. Therefore, the nonmalicious nodes have the opportunity to participate in the following consensus, and this mechanism can reduce the existence rate of malicious nodes.

2. Problem Statement

2.1. Problem of "Oligarchy" Nodes. Among the existing consensus algorithms based on credit, most of the accounting nodes are selected according to the size of the credit value, which is easy to produce "oligarchy" nodes, and the incentive degree for nodes with small credit value is not enough, such as CCAC algorithm [19]. The credit value of each node is calculated after the credit evaluation of the node, then the credit value is sorted from largest to smallest, and an accounting node is selected in this order, which can easily lead to the production of "oligarchy" nodes and cause other nodes to be less motivated. In this paper, we test the proportion of "oligarchy" nodes as accounting nodes in the total consensus times for CCAC to verify the adverse effects of "oligarchy" nodes on the network, and the results are shown in Table 1.

It can be seen from Table 1 that, with the number of consensuses increasing, the number of "oligarchy" nodes becoming accounting nodes also accounts for an increasing proportion, which can easily cause other nodes to be less motivated to work. Therefore, this paper proposes a mechanism for selecting accounting nodes based on credit space, which can effectively inhibit the generation of "oligarchy" nodes and increase the enthusiasm of nodes.

2.2. Problem of Node's Excessive Punishment. In view of the existing consortium blockchain consensus algorithm based on credit, the punishment for malicious nodes is too severe. More precisely, they do not judge whether the malicious behaviour of a node is deliberate or not, and the credit value of the nodes is always severely reduced so that these nodes cannot continue to participate in the consensus, typically such as the consensus algorithm in [22]. A PoS consensus mechanism based on credit value is proposed, and a credit value evaluation method is designed. The punishment equation for the credit value of malicious nodes is as follows:

$$trust_h^i = -trust_{h-1}^i, \tag{1}$$

where $trust_h^i$ represents the credit value of node i at the end of the hth cycle and $trust_{h-1}^i$ represents the credit value of node i at the end of the h-1th cycle. It can be seen from equation (1) that the credit value of the malicious node will be directly reduced to a negative value, making it difficult for the node to continue to participate in the following consensus. Besides, references [23, 24] mentioned in the Introduction also have the same punishment for malicious nodes. Both have too harsh punishments for malicious nodes, and normal consensus cannot be carried out for a long time. In this paper, we compare these algorithms to test the change in credit value of nodes with malicious behaviours, and the experimental results are shown in Figure 1.

It can be seen from Figure 1 that the credit value of malicious nodes in [22] will rapidly decrease from positive value to negative value, which is difficult to continue to participate in consensus for a long time. Although the algorithm in [23] did not reduce to a negative value, the credit value is very close to 0 and cannot compete with the credit value of normal nodes. In [24], the credit value of the malicious node is always below 0, and it is difficult to continue the normal consensus. By comparing the changes in the credit value of nodes with malicious behaviour in these three algorithms, it can be seen that they cannot participate in normal consensus for a long time for nodes with malicious behaviour. Therefore, this paper proposes a semifragile hierarchical punishment mechanism. This mechanism can make it difficult for malicious nodes to participate in consensus again, but nonmalicious nodes can continue to participate in consensus within a short amount of time.

3. Proposed Algorithm

3.1. Credit Evaluation Model. The credit of nodes represents the working performance of nodes in the process of participating in consensus [18]. The credit evaluation model proposed by the CCAC only considers the number of valid and invalid blocks generated by the accounting node and the time required to add on the chain [19]. However, it does not consider the time when the node is passive and offline from the block. In this paper, the credit evaluation of nodes will be carried out according to the four indicators of the number of transactions in the valid block, the time of the chain, the off-chain time, and the generation of invalid blocks. The credit evaluation indicators are given in Table 2.

Combined with these credit indicators, the data will be standardized so that the data can be calculated uniformly. In this paper, the minimum-maximum planning method is used to standardize the data. This method is the linear transformation of the original data, and the maximum max and minimum min will be set. After calculation by the standardized equation, the data range will fall between [0, 1], and then the following credit value is calculated. The computation equation is as follows:

$$i' = \frac{i - \min}{\max - \min}, \tag{2}$$

where max and min are obtained by preprocessing. In particular, the result of preprocessing is based on 100 consensus experiments in this paper. In the process of consensus experiments, the data of these indicators will be obtained, and max and min are the maximum and minimum values of data in each indicator. When an indicator in a node needs to be measured, it is only necessary to put the data into

Table 1: Percentage statistics of "oligarchy" nodes become accounting nodes.

Total consensus number	The proportion of times that nodes become accounting nodes (%)
400	52.4
600	58.3
800	62.8
1000	78.1

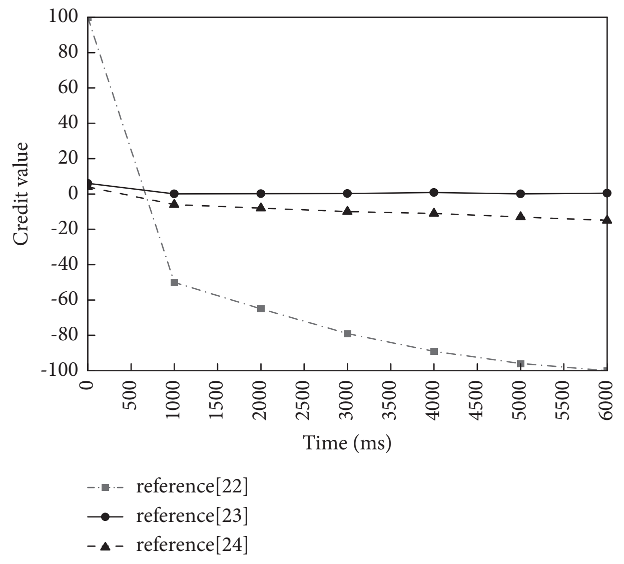

Figure 1: The credit value change diagram of the malicious nodes of each algorithm, recorded in a 6000 ms period. These dots represent the credit value of the malicious node taken every 500 ms.

the (2) to obtain the standardized value: i'_{num}, i'_{time}, $i'_{off-time}$, $i'_{invalid}$.

After getting the standardized data, some data may be positive or negative. Then, these data are added together, and finally, a value x that reflects the quality of the credit value is obtained, as shown in (3). Subsequently, the credit value $C(x_i)$ is to be accumulated or deleted by node i by (4).

$$x_i = i'_{num} + i'_{time} + i'_{off-time} + i'_{invalid}, \tag{3}$$

$$C(x_i) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-\mu)^2/2\sigma^2} dx, -\infty < x < \infty, \tag{4}$$

where x_i represents the number after processing of the standardized data mentioned above. $C(x_i)$ represents the credit value of node i. Besides, μ is the mean value calculated from the data obtained in the preprocessing, and σ is the variance calculated after preprocessing.

Similarly, the credit value of the accounting node that works hard will be accumulated, as shown in (5). The initial credit value of each node is 1. When node i becomes an accounting node for the first time, its credit value is equal to the initial credit value plus the $C(x_i)$ calculated by (4). Moreover, when node i is selected as the accounting node again, its credit value is the sum of the newly calculated credit value and the previously obtained. The equation for calculating the credit value of node i as an accounting node for the nth time is as follows:

$$Credit_i^n = \begin{cases} 1 + C(x_i)\, n = 1, \\ Credit_i^{n-1} + C(x_i)\, n \neq 1. \end{cases} \tag{5}$$

In contrast, for nodes with malicious behaviour, it will be subtracted after calculating the corresponding credit value, as shown in (6). For the initial malicious nodes, its credit value is the initial credit value minus $C(x_i)$ calculated by (4), that is, the credit value obtained after work. But for the malicious nodes in the consensus process, the credit value subtracts the new credit value from the previous credit value. The calculation equation of the mth malicious credit value of node i is as follows:

$$Credit_i^m = \begin{cases} 1 - C(x_i)\, m = 1, \\ Credit_i^{n-1} - C(x_i)\, m \neq 1. \end{cases} \tag{6}$$

Through the credit evaluation model for node credit evaluation, the nodes in working well condition can get a larger credit value. That is, their opportunity to become an accounting node will be greater, which creates a benign network environment for nodes actively participating in consensus.

3.2. Credit Space. In this paper, the credit space is used as the basis for selecting an accounting node. After a node obtains its credit value, its corresponding credit space is allocated according to the proportion of the credit value in the entire space. That is, the greater the credit value of the node, the greater the allocated space, and the greater the probability of becoming accounting node. For this reason, this method can better motivate nodes to work. Furthermore, the random algorithm also ensures the randomness of the algorithm, and it does not mean that nodes with larger credit space will certainly become accounting nodes. The credit space of node i can be computed by

$$C\_Space_i = \frac{Credit_i}{\sum_{i=1}^{n} Credit_i} \times L, \tag{7}$$

where $Credit_i$ represents the credit value of node i and $\sum_{i=1}^{n} Credit_i$ is the sum of the credit values of each node in this round. L is the total length of the credit space. Figure 2 is a graph of the change of credit space when a node is selected as an accounting node. Figure 2(a) shows the distribution of credit space of a certain round of nodes. Assuming that the pointer is randomly selected to node 3, the credit value of node 3 becomes larger after being selected as an accounting node and packaged block successfully. Obviously, according to the calculation equation of credit space, its credit space will also become larger. So node 3 has a greater probability in the next selection of accounting node, which is like playing a

Table 2: Credit evaluation indicators.

Indicator name	Explanation
i_{time}	The time when the node generates a block
$i_{off-time}$	The time the node leaves the blockchain
i_{num}	The number of transactions in a valid block
$i_{invalid}$	The number of invalid blocks generated by nodes in consensus

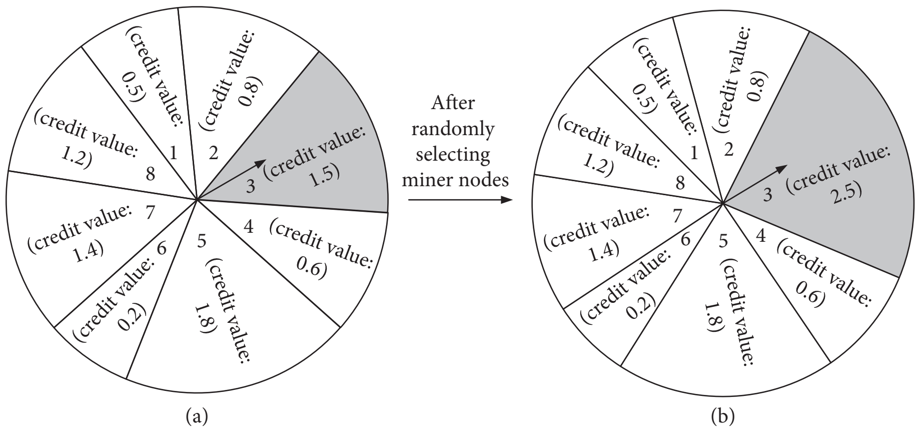

Figure 2: Credit space change diagram. Each sector in the figure represents the credit value of each node, and the pointer is like a turntable to represent random selection. (a) Distribution of credit space of each node in a certain round. (b) Distribution of credit space after selecting miner nodes.

roulette game. The greater the credit space of node 3, the greater the probability of the pointer pointing to node 3. Since the size of the whole credit space is fixed, each node's credit space is calculated according to the proportion of credit value, so the credit space of other nodes will be proportionally reduced.

What is more, after the node is selected as an accounting node, the corresponding packaging work must be completed. The packaging work involves the block structure, which is used to store and verify the credit value. The block structure includes the following:

(1) Blockhead: block version number, hash value of the previous block, timestamp, and random number.

(2) Blockchain time: record the time accounting nodes successfully package block into chain, which helps to verify whether the credit value is accumulated correctly.

(3) The hash value of the block's transaction data: record the transaction data generated by the accounting node of the block.

(4) Credit array in block: record the credit value obtained by nodes on blocks.

(5) Counting array in block: record the number of times a node becomes an accounting node.

Once completed the packaging work, the credit value will be calculated and accumulated in the original credit value. When the next accounting node selection begins, the credit space will be allocated according to the size of the credit value. However, there is a problem at present. Nodes with larger credit values may always be selected as accounting nodes, which leads to the generation of "oligarchy" nodes. Therefore, a threshold for becoming an accounting node is set. When it exceeds the current number threshold, it cannot continue to be selected as an accounting node. The threshold equation is as follows:

$$\tau = \frac{\sum_{i=1}^{Num} num_i}{Num} + t, \tag{8}$$

where τ is a threshold, Num represents the number of nodes, num_i denotes the number of node i becoming an accounting node, and $\sum_{i=1}^{Num} num_i$ represents the total number of times that all nodes become accounting nodes. The threshold calculated by the equation will change with the number of nodes becoming accounting nodes in the whole consensus network. When the threshold of this round increases, nodes may still be selected as accounting nodes. The constant t in the equation will be obtained through experiments, and the specific value is explained in the subsequent experimental part.

3.3. Semifragile Hierarchical Punishment Mechanism.

Generally, the punishment of malicious nodes in the existing credit mechanism is too severe, which directly reduces the credit value of malicious nodes and makes it too difficult to continue to participate in consensus. Consequently, this paper proposes a semifragile hierarchical punishment mechanism. Semifragile refers to the ability to distinguish whether a node with malicious acts is deliberate or nondeliberate. In our algorithm, the nodes judged as nondeliberate are given the opportunity to reparticipate in consensus.

In order to determine whether the malicious node is deliberate or nondeliberate, this paper judges the node by the number of malicious acts. When the number of malicious acts of a node is less than m, it is judged as a nondeliberate node, and vice versa. With respect to the value of m, this paper counts the number of malicious nodes through many experiments, and the results are shown in Table 3.

It can be seen from Table 3 that the nodes with the number of malicious acts less than or equal to 2 account for 98.53%, basically covering most of the nodes. Therefore, this paper selects 2 as the value of m, which is determined as the critical value of nonmalicious nodes.

The specific process of the semifragile hierarchical punishment mechanism is as follows; the general process is shown in Figure 3. First, all nodes will be placed in the *normal* layer, and then the credit space of nodes is calculated to select the accounting node. Moreover, the credit evaluation of the accounting node will be carried out. If the node has malicious behaviour, the node will be placed in the *prison* layer after calculating the credit value. It is worth mentioning that the nodes in the *prison* layer have no chance to be selected as accounting nodes and only the nodes in the *normal* layer have the chance to allocate the credit space to be selected as accounting nodes. The nodes in the *prison* layer will allocate the "custody" time according to the number of malicious acts. During this period, the nodes still need to participate in the data synchronization of the cluster. In particular, if the node is found to have malicious behaviour such as not performing block data synchronization or not working, it will continue to increase the "custody" time. After the time has passed, it is determined whether the node is deliberate or nondeliberate. If the node is a nondeliberate node, it will return to the *normal* layer and give it the opportunity to be selected as an accounting node again. Otherwise, the malicious node will continue to be punished.

About the node's "custody" time, when the number of nodes performing malicious acts increases, the time will increase obviously with the number of times. According to this characteristic, this paper uses the following function:

$$T = e^{x}. \tag{9}$$

The function is monotonically increasing, where T is the "custody" time and x is the number of malicious acts. It can be seen from (9) that T is monotonically increasing, that is, when x increases, that is, when the number of malicious acts increases, the time increases exponentially. In contrast, for nonmalicious nodes, only one or two malicious activities are performed, and the "custody" time is relatively appropriate. It conforms to the principle of the proposed punishment mechanism, gives a good buffer to the nodes that do not deliberately perform malicious acts, and then gives the opportunity to participate in the consensus.

4. Algorithm Design

Firstly, the credit value of all nodes is initialized to 1. In the beginning, each node is placed in the *normal* layer, and each participating node is numbered. Moreover, the corresponding credit space is allocated according to the credit value of each node. Obviously, the size of the space allocated by each node is the same, and the total space is unchanged. Then the credit array Cn and count array Cc are constructed. Cn is used to store the credit value of the node, and Cc is the number of times the storage node has become an accounting node. Thereafter, begin the cycle of selecting the accounting nodes. The process of credit consensus is shown in Figure 4. As shown in Figure 4, the whole process can be divided into four steps: initialization stage, cyclic selection of accounting node stage, constructing block stage, and checking the new block stage.

TABLE 3: Statistics of the number of malicious acts. The proportion indicates the proportion of nodes with different times of malicious acts in the total nodes.

Number of malicious acts	Proportion of the number of nodes (%)
≤0	96.25
≤1	97.84
≤2	98.53
>2	1.47

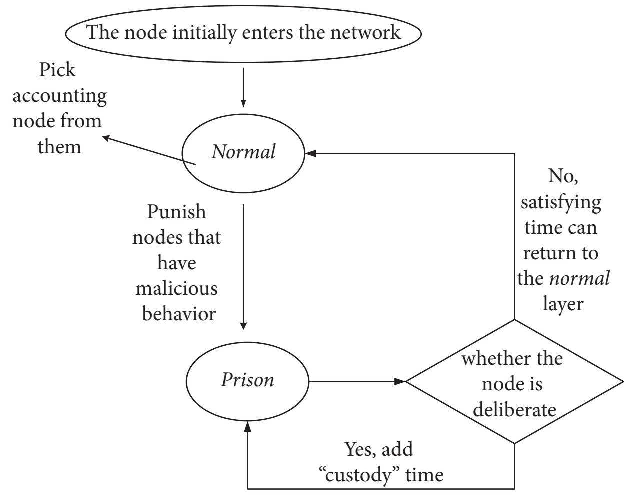
The node initially enters the network
Pick accounting node from them
Normal
Punish nodes that have malicious behavior
No, satisfying time can return to the *normal* layer
Prison
whether the node is deliberate
Yes, add "custody" time

FIGURE 3: Semifragile hierarchical punishment mechanism. Malicious nodes will judge whether the node is deliberate and take corresponding measures.

4.1. Initialization Stage. In the initial stage, the initial credit value of each participating node in the *normal* layer is set to 1, and the total credit space length is set to 100. The credit space of each node is calculated by equation (7), and the number of participating nodes is assigned. Thereafter, the credit array Cn and the count array Cc are constructed to store the credit value of the node and the number of nodes becoming accounting nodes, respectively. Algorithm 1 shows how to allocate the node's credit space.

4.2. Cyclic Selection of Accounting Node Stage. Through Algorithm 1, we have obtained the credit space of each node. Then, the algorithm randomly selects the accounting node.

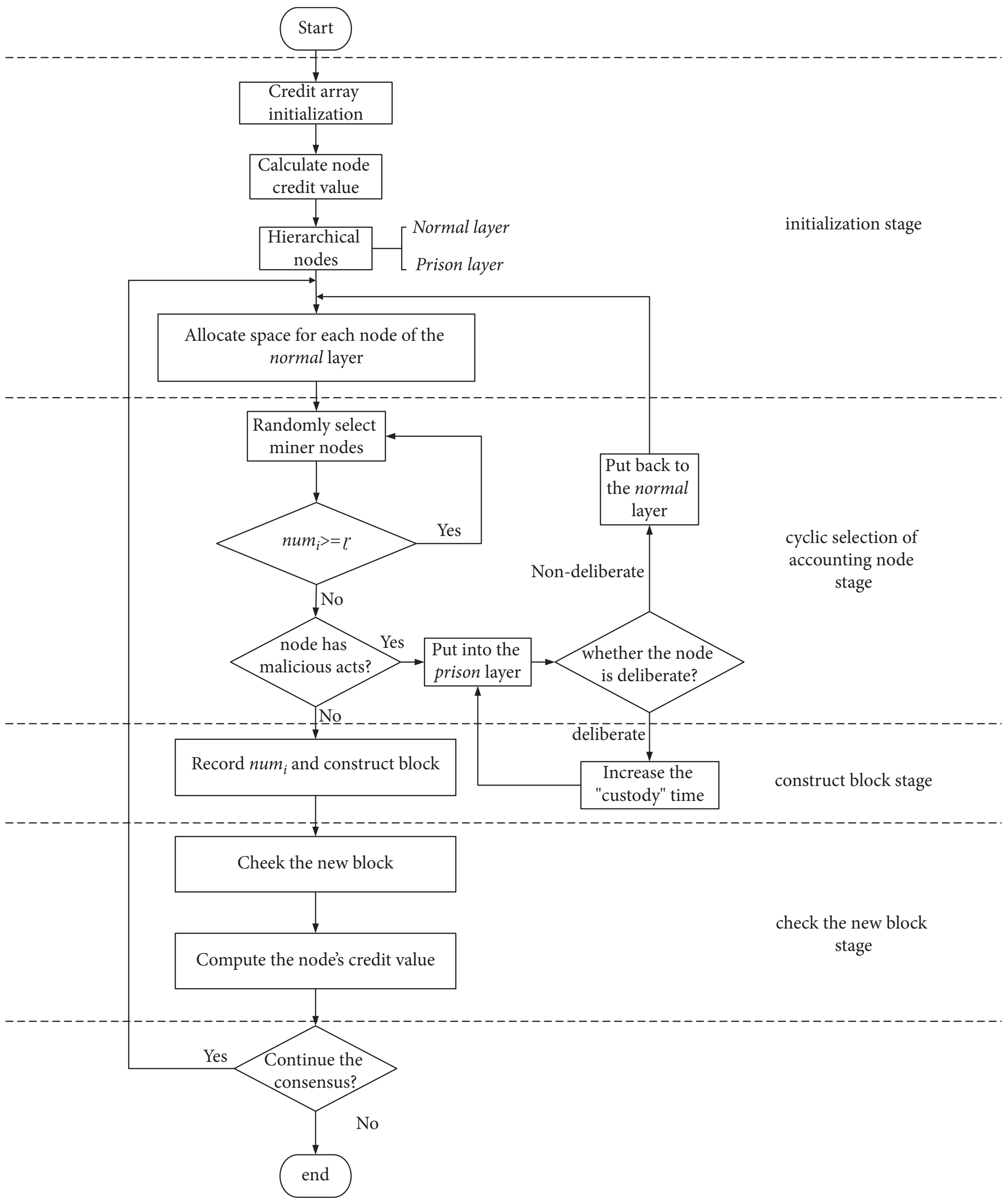
Start
Credit array initialization
Calculate node credit value
Hierarchical nodes
Normal layer
Prison layer
initialization stage
Allocate space for each node of the normal layer
Randomly select miner nodes
num_i>=τ
Yes
No
node has malicious acts?
Yes
Put into the prison layer
whether the node is deliberate?
Non-deliberate
Put back to the normal layer
cyclic selection of accounting node stage
No
deliberate
Record num_i and construct block
Increase the "custody" time
construct block stage
Cheek the new block
Compute the node's credit value
check the new block stage
Yes
Continue the consensus?
No
end

Figure 4: Credit consensus process. This flowchart describes how nodes select accounting nodes and how to punish malicious nodes.

More precisely, each interval represents each node's credit space, and the algorithm selects accounting node by setting a random number and judging which interval the random number falls into. As a result, the node represented by this interval is selected as accounting node. Subsequently, the accounting node will complete the corresponding work and obtain the corresponding credit value. Generally, if an accounting node does not work, the node will be punished beyond the given time and enter the next accounting node's selection. When the next accounting node selection is conducted, the corresponding space will be allocated according to the credit value. If the credit value is larger, it is easier to obtain the packaging right. Since it is randomly selected, there will be nodes with low credit values that get the right to package. In order to avoid the generation of "oligarchy" node, τ is set. If num_i exceeds τ, node i cannot be selected as an accounting node. But it does not mean that i cannot be selected as an accounting node anymore because τ will change with $\sum_{i=1}^{Num} num_i$ in the whole consensus network. When τ becomes larger, node i may still be selected as the accounting node. Algorithm 2 gives the process of randomly selecting an accounting node.

If the node has malicious acts, the node will be placed in the *prison* layer for "custody." "Custody" time will be

```
Input: $Cn$ (node's credit value array)
Output: spaceArray (node's credit space array)
 (1)   spaceArray[] = {0}; //Initialization of Credit Space Array
 (2)     for $i = 1$ to $n$ do //Traversing all nodes
 (3)       if JudgePrsion ($Cn[i]$) //Judge if the node is in the prison layer, if it is, do not allocate
 (4) space
 (5)         continue;
 (6)       end if
 (7)       if $i = = 0$ //When $i$ is the first node in space
 (8)         spaceArray[$i$]=($Cn$[i]/countSum ($Cn$)) $*$ spaceLength; //Calculating the length of credit
 (9) space
(10)         continue;
(11)       end if
(12)       spaceArray[$i$] = ($Cn$[i]/countSum ($Cn$)) $*$ spaceLength + spaceArray[$i-1$]; //The length of
(13) credit space after becoming an accounting node, $Cn$ is an array of normal layers
(14) end for
```

Algorithm 1: Node layering and credit space allocation algorithm.

calculated according to (9). Then, it will determine whether the time has expired. If it has expired, determine whether the node is a malicious node. If it is not a malicious node, it will be released back to the *normal* layer. If it is a malicious node, continue to stay in the *prison* layer for "custody." If it has not expired, it will continue to stay in the *prison* layer. Algorithm 3 gives the penalty mechanism of malicious nodes.

4.3. Construct Block Stage. Once the accounting node is selected, the accounting node will broadcast the constructed block to all adjacent nodes. Afterward, adjacent nodes will receive the new block and broadcast it to the whole network after successful verification. When the block is verified, the block will be added to each node's blockchain copy. After all nodes have received and verified the block, the work of the next block construction will proceed.

4.4. Check the New Block Stage. After selecting the accounting node and completing the related transactions on the block, the node will broadcast the generated block to the whole network and then verify the credit value. Once the verification is correct, the node will obtain the corresponding credit reward. On the contrary, if the node has malicious behaviour, the behaviour will be recorded in the block, and the corresponding credit punishment will be carried out.

5. Experimental Results and Analysis

In our experiments, we use Golang programming language and JetBrainsGoLand 2020.3.4 for the simulation test. First, we use the Go language to write a single-machine multinode platform to simulate the consensus process. Then, we compare the performance of the consensus algorithm proposed in this paper with CCAC algorithm [19], CPoW algorithm [20], and master-slave multichain algorithm [22], and test the number of malicious nodes, punishment mechanism, and the consensus delay of nodes. Finally, the images are drawn according to the experimental data for comparative analysis.

5.1. Threshold Equation Constant Experiment. This experiment is to analyze the value of threshold equation constant. We selected 40 nodes for 600 consensuses and tested the average time consumption to select accounting nodes under different threshold equation constants.

It can be seen from Figure 5 that the average time consumption of selecting accounting nodes with a constant 3 is the least, while the average time consumption of other nodes is relatively high. Therefore, we choose constant 3 as the value of t in the threshold equation.

5.2. Statistics of Accounting Node Number. In this experiment, we test the number of times nodes become accounting nodes to verify the credit evaluation model and the mechanism of selecting accounting nodes. First, set 20 nodes, conduct consensus on them 600 times, and select accounting nodes. The experimental results are shown in Figure 6. As can be seen from the data in Figure 6, each node can become an accounting node in the consensus process. Some nodes have become accounting nodes only 5 or 6 times, and some nodes have become accounting nodes 18 or 20 times. This shows that the proposed algorithm can reflect the role of credit value and ensure the randomness of selecting accounting nodes through credit space.

In order to test whether the threshold τ can better limit the "oligarchy" node, this paper tests 20 nodes, carries out 1500 consensuses on them, selects the accounting node, and then records the number of rounds of threshold change and the highest number of accounting node in this round. The experimental results are shown in Figure 7. As shown in Figure 7, the threshold τ will change with the number of nodes becoming accounting nodes in the whole network, and the number of accounting nodes is also limited to the threshold τ. The number of accounting nodes increases more and more slowly and requires a longer consensus time. This

```
Input: spaceArray (Node's Credit Space Array)
Output: i (accounting node's serial number)
 (1)  while (nodeSelect) //nodeSelect is whether to select the miner to complete the identifier,
 (2)  the initial value is true
 (3)      rand.Seed (time.Now().Unix()); //Set random number time seed
 (4)      randomSize = randomFloat (0, spaceLeangth); //Random number selected in space
 (5)      node = judgeSelect (spaceArray, randomSize); //Determine which node is selected
 (6)      if CoutArray [node] ≤ Exceeded //The requirement cannot exceed the threshold
 (7)          nodeSelect = false; //The selection is complete, jump out of the loop, otherwise
 (8)  continue to choose
 (9)          Cc[i]++; //Count value plus 1
(10)          return i;
(11)  end if
(12)  end while
```

Algorithm 2: Random selection accounting node algorithm.

```
Input: U (the set of malicious nodes)
Output: prisonArray (prison layer array)
 (1)  prisonArray[] = {}; //Initialize the prsion layer
 (2)  while (node in U)
 (3)  if JudgeMalicious (node) //Determine whether the node is malicious
 (4)      time = pow (e, x); //Calculate penalty time
 (5)      insert (node, prisonArray, time); //Put the node in jail and record the punishment time
 (6)      node++; //Pointer moved to the next malicious node
 (7)      continue;
 (8)  end if
 (9)  if JudgeTimeOut (node) //Determines whether the node penalty time expires
(10)      if (maliciousCount ≤ 2) //Determines whether the node is a malicious node
(11)              remove (node, prisonArray); //Remove the node
(12)              node++; //Pointer moved to the next malicious node
(13)              continue;
(14)          else
(15)              stayPrison (node); //Leave the node in the prison layer
(16)              node++; //Pointer moved to the next malicious node
(17)              continue;
(18)          end if
(19)  end if
(20)  end while
```

Algorithm 3: Punishment algorithm for malicious nodes.

shows that the proposed threshold mechanism can effectively restrain the emergence of "oligarchy" nodes.

5.3. Semifragile Hierarchical Punishment Mechanism Experiment. In order to prove that the punishment mechanism proposed in this paper can effectively avoid malicious nodes destroying the consensus process, we do an experiment to test the number of malicious nodes in different algorithm. The experimental results are given in Figure 8. At first, 1000 nodes are set in the system, and 273 malicious nodes are set and labelled artificially in these nodes. With the increase of consensus times, it can be found that the number of labelled malicious nodes in each algorithm is gradually decreasing, but it should be noted that the number of malicious nodes in the proposed algorithm in this paper has a more obvious decline. From Figure 8, it can be seen that when the 70th consensus is carried out, the number of labelled malicious nodes in the algorithm proposed in this paper is reduced to 32, and the number of malicious nodes in other algorithms is more than that of this algorithm. This shows that the credit evaluation model proposed in this paper will gradually reduce the credit value of the malicious nodes. At the same time, the hierarchical punishment mechanism will also punish malicious nodes, which further restrains malicious nodes from doing evil. With the increase in the number of consensuses, the probability of selecting the malicious nodes as accounting nodes will be greatly reduced; as such, it will make the blockchain system more safe and reliable.

In order to further test the performance of the proposed semifragile hierarchical punishment mechanism, this paper does an experimental test to determine the malicious

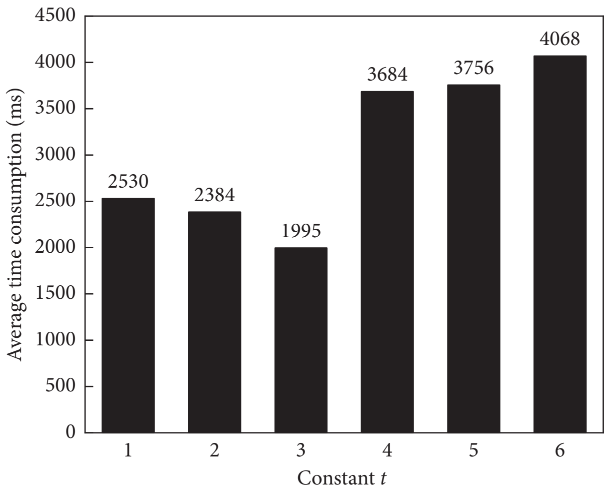

Figure 5: The experimental statistics of the constant t. The average time consumption of each constant in 600 consensuses is recorded.

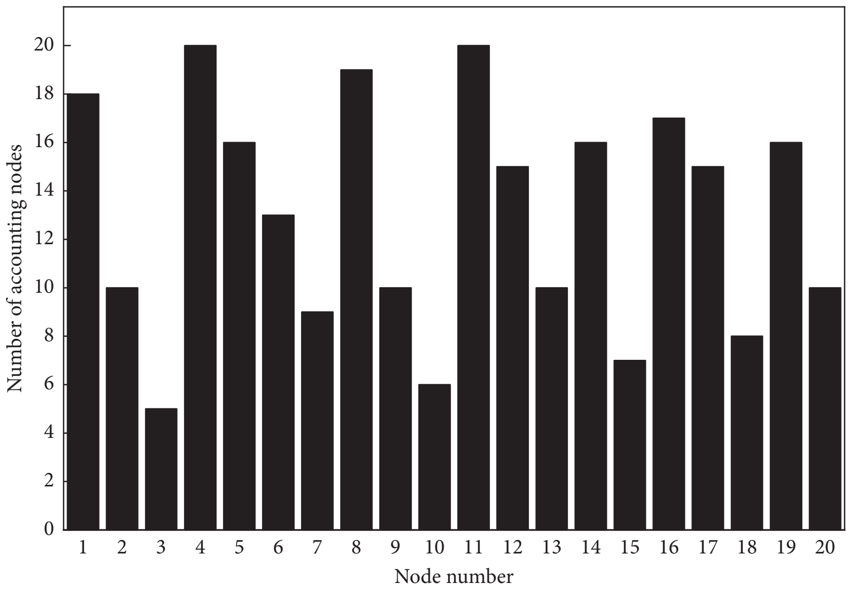

Figure 6: Statistics of the number of accounting nodes. The number of times each node becomes an accounting node in 600 consensuses is recorded.

behaviours of nodes. Firstly, a deliberate node and a nondeliberate node are marked, respectively, and they are placed in the *prison* layer. According to the proposed mechanism, nondeliberate nodes will be put back to the *normal* layer over time to continue to join the consensus, we record their credit values to observe the work of the node, and the results are shown in Figure 9.

It can be seen from Figure 9 that the credit value of a nondeliberate node decreases after malicious acts. After putting it into the *prison* layer, the credit value remains unchanged. If it is put back to the *normal* layer after exceeding the "custody" time, it can normally participate in the consensus. However, the credit value of the deliberate node declines after committing malicious acts. It is worth mentioning that if the deliberate node continues to commit malicious acts in the *prison* layer, the "custody" time will be double. It shows that the mechanism gives an opportunity to nondeliberate nodes and does not reduce its credit value to the point of being unable to participate in the consensus. That is, it makes nondeliberate nodes become normal nodes, while deliberate nodes are punished accordingly.

5.4. Consensus Delay. The consensus delay comparison results are shown in Figure 10. As can be seen from Figure 10, with the increase of consensus times, consensus delay increases gradually. The consensus delay of the CCAC algorithm is the lowest, the consensus delay of the proposed algorithm is only higher than that of CCAC, and the consensus delay of the CPoW algorithm is the highest. This is because the proposed algorithm in this paper selects accounting nodes based on the credit space and introduces the hierarchical punishment mechanism, which results in higher delay than CCAC. However, the consensus delay of the algorithm is within an acceptable range, and it does not

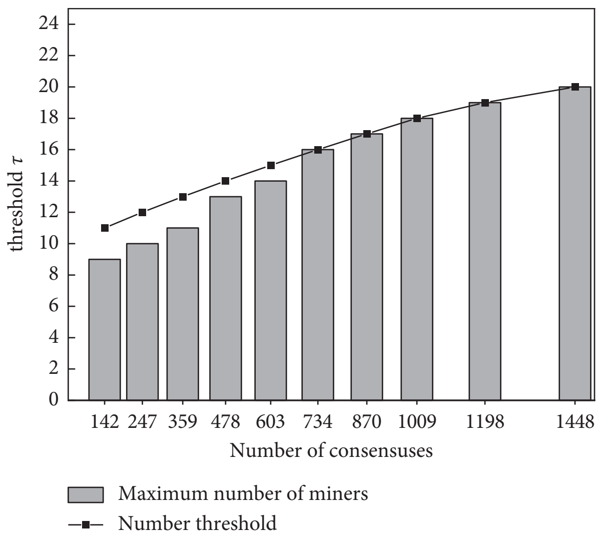

Figure 7: The change of the threshold τ. Each interval of the abscissa represents the current number of rounds when the number of thresholds has changed.

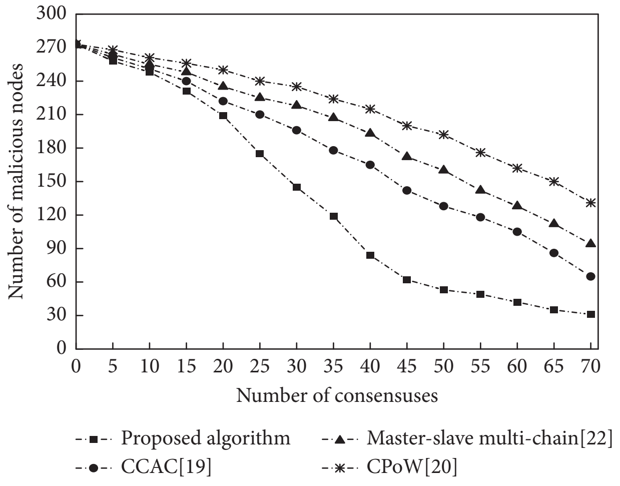

Figure 8: The number change of malicious nodes, recorded in 70 consensuses.

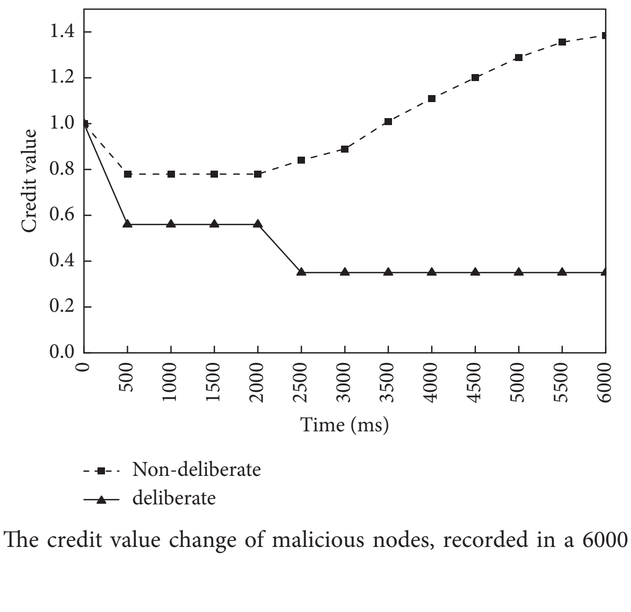

Figure 9: The credit value change of malicious nodes, recorded in a 6000 ms period.

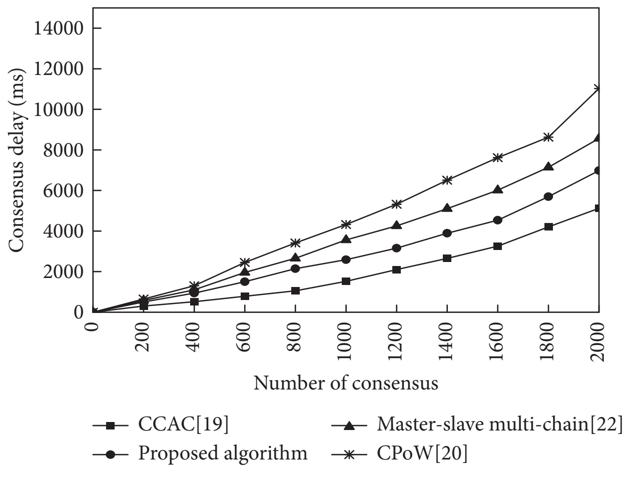

Figure 10: Comparison of consensus delay, recorded in 2000 consensuses.

affect the normal operation of the entire blockchain system. Compared with the proposed algorithm in this paper, CPoW is more difficult to solve the hash problem with the increase of blockchain length. Therefore, CPoW will consume a lot of computing power and have a high consensus delay. Because the master-slave multichain algorithm is based on the PoS algorithm, compared with CPoW, it saves a lot of energy consumption without mining. However, compared with the algorithm proposed in this paper, its consensus process is more complex and prone to bifurcation. Thus, the consensus delay is higher than that of the proposed algorithm in this paper.

5.5. Limitation. It can be seen from the results of the above experiments that the algorithm proposed in this paper can suppress the "oligarchy" nodes and deal with the deliberate nodes very well, but there are still some limitations. In this part of the credit evaluation model, the evaluation indicators set are not complete enough, so the evaluation of the nodes may not be comprehensive enough. This is a relatively limited point, and there is room for improvement in the future.

6. Conclusion

This paper proposed a semifragile consortium blockchain consensus algorithm based on credit space. According to the working situation of the node, we designed a credit evaluation model to calculate the credit value of the node and allocated the credit space. Besides, we proposed a randomly select mechanism for the accounting node based on the credit space, which solved the problem of insufficient incentive in the consensus algorithm and ensured the randomness of the node to become an accounting node. The experimental results show that the consensus mechanism in this paper has randomness while ensuring credit incentive; it enhances the security of the algorithm. In addition, it is more reasonable for the node penalty mechanism and has better performance in consensus efficiency, which is suitable for consensus in the consortium blockchain. Nonetheless, the algorithm still has shortcomings in the determination of malicious nodes and the design of the "custody" time equation of the semifragile hierarchical punishment mechanism. The next step will continue to conduct in-depth research on these two aspects.

Data Availability

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

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (nos. 61762046 and 62166019), the Science and Technology Research Project of Education Department of Jiangxi Province (no. GJJ209412), and the National Innovation and Entrepreneurship Training Program for College Students (no. 201913434005).

References

[1] S. Nakamoto, "Bitcoin: a peer-to-peer electronic cash system," 2008, https://bitcoin.org/bitcoin.pdf.

[2] S. Zhang and .-H. Lee, "Analysis of the main consensus protocols of blockchain," *ICT express*, vol. 6, no. 2, pp. 93–97, 2020.

[3] M. X. Du, X. F. Ma, Z. Zhang, W. Xiangwei, and C. Qijun, "A Review on Consensus Algorithm of blockchain," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 2567–2572, Banff, AB, Canada, October 2017.

[4] U. Satapathy, B. K. Mohanta, S. S. Panda, S. Sobhanayak, and D. Jena, "A secure framework for communication in internet of things application using hyperledger based blockchain," in *Proceedings of the 2019 10th international conference on computing, communication and networking technologies (ICCCNT)*, pp. 1–7, Kanpur, India, July 2019.

[5] M. Mrinal, A. Garg, V. D. Maikandavel, and A. Panja, "Blockchain secured vehicle tracking using wireless sensor network," *Ilköğretim Online*, vol. 20, no. 1, pp. 2472–2480, 2021.

[6] G. T. Nguyen and K. Kyungbaek, "A survey about consensus algorithms used in blockchain," *Journal of Information processing systems*, vol. 14, no. 1, pp. 101–128, 2018.

[7] Y. A. Min, "A study on performance evaluation factors of permissioned blockchain consensus algorithm," *Jouranl of Information and Security*, vol. 20, no. 1, pp. 3–8, 2020.

[8] F. Zeng, Q. Chen, L. Meng, and J. Wu, "Volunteer assisted collaborative offloading and resource allocation in vehicular edge computing," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3247–3257, 2021.

[9] F. Zeng, Y. Chen, L. Yao, and J. Wu, "A novel reputation incentive mechanism and game theory analysis for service caching in software-defined vehicle edge computing," *Peer-to-Peer Networking and Applications*, vol. 14, no. 2, pp. 467–481, 2021.

[10] 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, no. 9, Article ID 113385, 2020.

[11] S. Pahlajani, A. Kshirsagar, and V. Pachghare, “Survey on Private Blockchain Consensus Algorithms,” in *Proceedings of the International Conference on Innovations in Information and Communication Technology (ICIICT)*, pp. 1–6, Chennai, India, April 2019.

[12] A. Zhang and X. Lin, “Towards secure and privacy-preserving data sharing in e-health systems via consortium blockchain,” *Journal of Medical Systems*, vol. 42, no. 8, pp. 1–18, 2018.

[13] H. Thomas and P. Alex, “Verifiable Anonymous Identities and Access Control in Permissioned Blockchains,” Massachusetts Institute of Technology,” 2016, http://arxiv.org/abs/1903.04584.

[14] H. Huang, P. Zhu, F. Xiao, X. Sun, and Q. Huang, “A blockchain-based scheme for privacy-preserving and secure sharing of medical data,” *Computers & Security*, vol. 99, no. 12, pp. 102010–102023, 2020.

[15] S. J. Alsunaidi and F. A. Alhaidari, “A Survey of Consensus Algorithms for Blockchain Technology,” in *Proceedings of the 2019 International Conference On Computer And Information Sciences (ICCIS)*, pp. 1–6, Sakaka, Saudi Arabia, April 2019.

[16] Y. Wu, P. Song, and F. Wang, “Hybrid consensus algorithm optimization: a mathematical method based on POS and PBFT and its application in blockchain,” *Mathematical Problems in Engineering*, vol. 2020, 2020.

[17] X. Zheng, W. Feng, M. Huang, and S. Feng, “Optimization of PBFT algorithm based on improved C4. 5,” *Mathematical Problems in Engineering*, vol. 2021, 2021.

[18] D. Wang, C. Jin, H. Li, and M. Perkowski, “Proof of activity consensus algorithm based on credit reward mechanism,” in *Proceedings of the International Conference on Web Information Systems and Applications*, pp. 618–628, Guangzhou, China, September 2020.

[19] S. Z. Li, L. Huang, X. H. Deng, Z. Q. Wang, and H. W. Liu, “Consortium chain consensus algorithm based on credit,” *Application Research of Computers*, vol. 38, no. 8, pp. 2284–2287, 2021.

[20] Z. Wang, Y. L. Tian, Q. X. Li, and X. YANG, “Proof of work algorithm based on credit model,” *Journal on Communications*, vol. 39, no. 8, pp. 185–198, 2018.

[21] F. Li, K. Liu, J. Liu, Y. Fan, and S. Wang, “DHBFT: dynamic hierarchical Byzantine fault-tolerant consensus mechanism based on credit,” in *Proceedings of the Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference On Web And Big Data*, pp. 3–17, Tianjin, China, August 2020.

[22] H. Z. Liu, S. S. Li, W. L. Lv, and S. J. Wei, “Master-slave multiple-blockchain consensus based on credibility,” *Journal of Nanjing University of Science and Technology*, vol. 44, no. 3, pp. 325–331, 2020.

[23] A. Bugday, A. Ozsoy, S. M. Öztaner, and H. Sever, “Creating consensus group using online learning based reputation in blockchain networks,” *Pervasive and Mobile Computing*, vol. 59, no. 10, pp. 101056–101070, 2019.

[24] J. Huang, L. Kong, G. Chen, L. Cheng, K. Wu, and X. Liu, “B-IoT: Blockchain Driven Internet of Things with Credit-Based Consensus Mechanism”” in *Proceedings of the 2019 IEEE 39th International Conference On Distributed Computing Systems (ICDCS)*, pp. 1348–1357, Dallas, TX, USA, October 2019.