electronics

MDPI

Article

An Effective Sharding Consensus Algorithm for Blockchain Systems

Runyu Chen, Lunwen Wang, Chuang Peng and Rangang Zhu *

College of Electronic Engineering, National University of Defense Technology, Hefei 230037, China
* Correspondence: zhurangang17@nudt.edu.cn

Abstract: Sharding is the widely used approach to the trilemma of simultaneously achieving decentralization, security, and scalability in traditional blockchain systems. However, existing schemes generally involve problems such as uneven shard arithmetic power and insecure cross-shard transaction processing. In this study, we used the Practical Byzantine Fault Tolerance (PBFT) as the intra-shard consensus and, here, we propose a new sharding consensus mechanism. Firstly, we combined a jump consistent hash algorithm with signature Anchorhash to minimize the mapping of the node assignment. Then, we improved the process of the cross-shard transaction and used the activity of nodes participating in intra-shard transactions as the criterion for the shard reconfiguration, which ensured the security of the blockchain system. Meanwhile, we analyzed the motivation mechanism from two perspectives. Finally, through theoretical analysis and related experiments, we not only verified that the algorithm can ensure the security of the entire system, but also further clarified the necessary conditions to ensure the effectiveness of the shards and the system on the original basis.

Keywords: blockchain; sharding; Practical Byzantine Fault Tolerance; consensus mechanism; jump consistent hash algorithm; Anchorhash; timestamp

Citation: Chen, R.; Wang, L.; Peng, C.; Zhu, R. An Effective Sharding Consensus Algorithm for Blockchain Systems. *Electronics* **2022**, *11*, 2597. https://doi.org/10.3390/electronics11162597

Academic Editor: Xue (Shelley) Lin

Received: 14 June 2022
Accepted: 18 August 2022
Published: 19 August 2022

1. Introduction

Blockchain is a new economic organization model built on a series of technologies. It was created in 2009 during the construction of the Bitcoin system [1]. It has gradually become a global economic hotspot over the past 10 years and has been widely used in many applications [2], including IoT [3], cloud computing [4], smart cities [5], and supply chains [6].

Blockchain technology is developing rapidly, but there are fundamental and practical obstacles to its wider applicability, the most critical issue being the trilemma of decentralization, security, and scalability [7].

In order to improve the scalability of the system on the basis of ensuring decentralization and security, existing methods can be divided into two solutions: off-chain and on-chain. The off-chain solution avoids the computational cost of traditional blockchain systems, requiring each honest node to receive, store, and send relevant data, and to reach an agreement among all nodes while guaranteeing the total order of valid data. This scheme has applicability in some cases, but in many application scenarios, a face solution is required, i.e., recording, validating, and saving all data. A sharding blockchain system is an on-chain solution designed to improve the scalability of traditional blockchain systems in order to achieve the same level of security and decentralization.

Based on the composition of each part of the sharding blockchain, this paper studies the sharding blockchain. The main contributions are as follows:

1. We propose a sharding construction method based on signature Anchorhash and jump consistent hash function to improve the rationality and security of sharding while minimizing remapping caused by sharding changes;

2. We improved cross-shard transaction processing and shard reconstruction in our study and, here, propose a shard reconstruction method based on the activity of the nodes participating in intra-shard transactions;
3. We calculated the effectiveness of each shard and the whole system using hypergeometric distribution, and quantitatively analyzed the impact of certain factors on the effectiveness of the shards by relying on the literature [8].

2. Background and Related Work

Sharding is an innovative aspect of database systems [9], where it describes a method of dynamically partitioning a database into parts (called shards), each managed by a different node in a distributed system. In a sharding blockchain system, nodes in the network are dynamically partitioned into shards (subsets), and each shard individually performs part of the storage, communication, and computing tasks. The sharding blockchain system can be roughly composed of seven parts: node selection, epoch randomness, node allocation, intra-shard consistency, cross-shard transaction processing, shard reconfiguration, and the incentive mechanism [10]. The schematic diagram is shown in Figure 1.

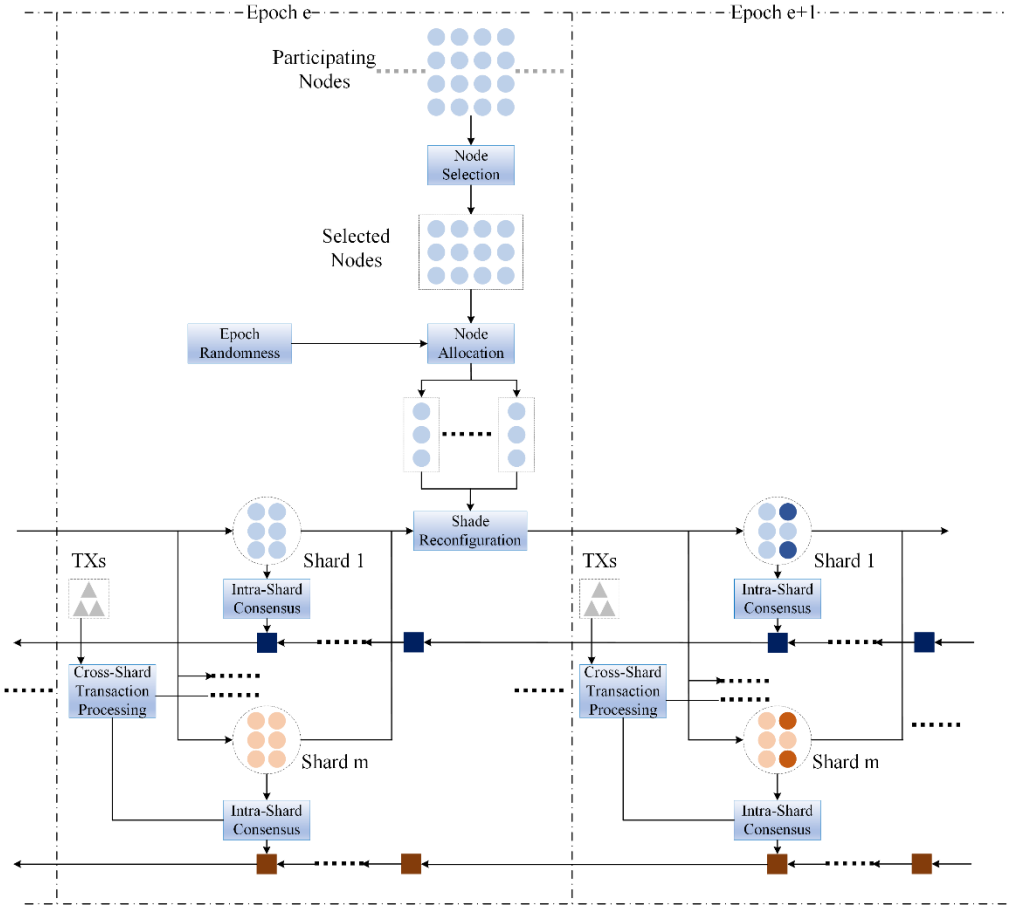


Figure 1. The schematic diagram of a sharding blockchain.

The following is a brief introduction to the construction method, typical solutions, and existing problems of the sharding blockchain.

2.1. Shard Construction

The sharding blockchain has the following three characteristics: communication sharding, computing sharding, and storage sharding.

Communication sharding refers to the division of the entire network into different shards. Each shard is handled by a corresponding committee. The members of each

committee only need to conduct the internal communication most of the time. Other clients and nodes within each shard can obtain the current state of the blockchain by communicating with the intra-shard committee.

Computing sharding means that each sharding committee is only responsible for processing its corresponding transaction, such as judging its corresponding sharding based on the transaction ID and verifying the legitimacy of the transaction for the distributed consensus algorithm in the transaction operation committee, in order to decide whether these transactions can be added to the blockchain. Computing sharding enables different transactions to be processed by different committees in parallel. When the number of nodes in the network increases, more committees can be added, so that different transactions can be processed in parallel by different committees at the same time. The transaction processing performance is enhanced with the number of nodes in the network, which in turn enables the scalability of transaction processing. The main sharding methods are the UTXO-based ledger model [11] and account-based ledger model [12,13].

Storage sharding means that different sharding committees store the processed transactions in shards. Each sharding committee is only responsible for processing the transactions corresponding to this shard and placing the transactions on the blockchain dedicated to this shard in order to reduce the storage burden of the nodes. In this case, cross-shard communication is unavoidable. Compared with the other two sharding mechanisms, storage sharding is the most difficult to implement: firstly, the possible centralization risks must be avoided; secondly, one must ensure that the cross-shard communication does not exceed the performance benefits of the storage shards to ensure high data availability [14]; and thirdly, the impact of the shard reconfiguration on storage needs to be considered.

No matter what type of sharding method is used, the communication between nodes plays a very important role. The schematic diagram is shown in Figure 2. Nodes in the same shard only need to perform intra-shard communication most of the time and send some key information to the shard's coordinator. The coordinator is usually responsible for the cross-shard communication and intra-shard consensus, and each shard has at least one coordinator. In general, coordinators need to have stronger communication capabilities than other intra-shard nodes.

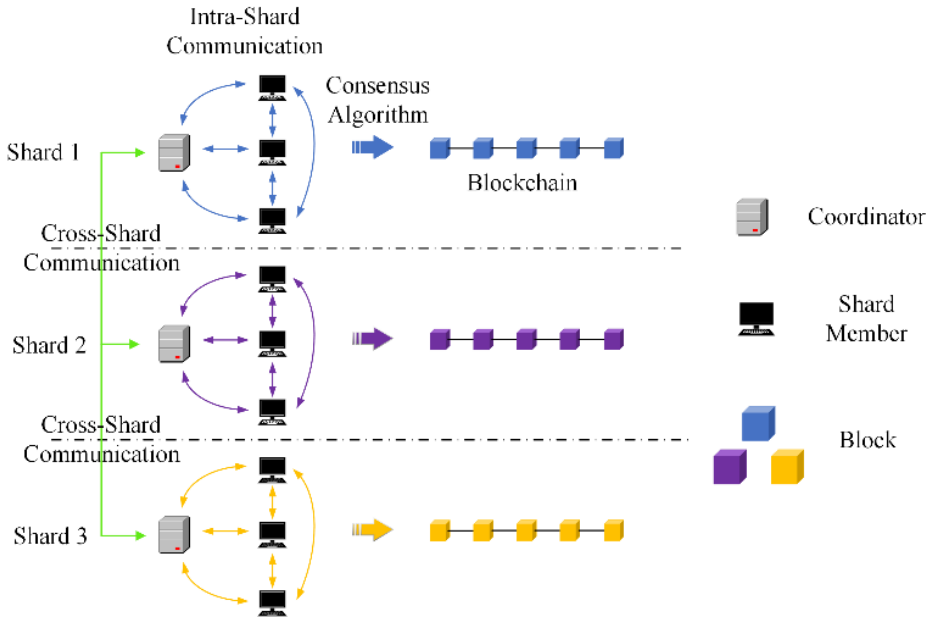


Figure 2. Communications in sharding blockchain systems.

2.2. Typical Schemes

The sharding consensus mechanism was proposed by Luu et al. [15], and ELASTICO is the earliest sharding consensus mechanism. It assumes that the network model is

a synchronous network. The algorithm mainly includes five parts: node identity establishment, node sharding confirmation, internal committee consensus, broadcast confirmation of the blocks, and random number generation. Kokoris-Kogias et al. proposed Omniledger [16], which added committee reconfiguration to ELASTICO and proposed an anti-locking solution for cross-shard transactions, which effectively solved ELASTICO's low operating efficiency and cross-shard transactions. Chainspace [17] was proposed by Al-Bassam et al. A smart contract application platform was built on the basis of sharding, which fostered the communication sharding and computing sharding of transactions and smart contracts. Rapidchain [18] was proposed by Zamani et al. This system used the message diffusion algorithm to replace the gossip communication protocol adopted by the traditional blockchain network and optimized the processing method of cross-shard transactions, which not only enabled computing sharding and communication sharding, but also enabled state sharding. It also enabled the consensus within its period to be close to the fast response characteristic.

In addition, Dange [19] used database sharding to achieve the scalability of the blockchain and used trusted random numbers generated by trusted hardware as the criteria for the node assignment, which ensured the activity and security of the whole protocol, to some extent. Kwak proposed a hierarchical negotiation mechanism based on service area sharding [20] to improve the performance degradation due to the increased system flow. Yang [21] used sharding technology combined with a hybrid model to set up an online e-voting system, which ensured the security of the voting process.

2.3. Existing Problems

While achieving transaction processing scalability, the sharding consensus introduces some new problems [22], including:

Firstly, cross-shard transaction processing. Transactions that contain multiple inputs require multiple shards to work together not only to process transactions safely and efficiently, but also to prevent problems, such as double-spending attacks and transaction lockups, and other problems during processing.

Secondly, the dynamic management of the number of sharded nodes. One option is to periodically adjust the composition of the shard committee members to effectively resist the control of the committee by possible malicious nodes. With respect to the other option, there may be differences in the number of committees corresponding to different shards, and when the communication between two shards is too frequent, and we can merge them into one shard to improve the transaction processing speed.

Thirdly, the detection and repair of malicious shards. On the one hand, the uneven distribution of computing power among the shards directly leads to the emergence of malicious committees. On the other hand, if the process of assigning new members to different committees is biased by the adversary, some shards may fail to work properly.

Fourthly, the establishment of motivation mechanisms. The establishment of motivation mechanisms needs to take into account the differences between shard blockchains and ordinary blockchains. Not only do we need to design rewards for different nodes based on the intra-shard consensus, but we also need to consider the impact of the cross-shard transaction processing.

3. Overview of the Algorithm

In this section, we divide the whole process of the shard consensus mechanism into five parts: node assignment (including node selection and ephemeral randomness), intra-shard consensus, cross-shard transaction processing, shard reconfiguration, and the motivation mechanism. In order to guarantee the strong consistency of the sharding blockchain system and ensure that all shards can effectively deal with possible malicious attacks, we use the traditional PBFT [23] consensus algorithm as the intra-shard consistency protocol and, on this basis, we introduce the process of the remaining four parts of the sharding consensus algorithm in detail.

3.1. Node Assignment

Node assignment mainly consists of two parts: node allocation in the initial sharding stage and node redistribution, when the number of shards changes. Since the intra-shard consensus protocol uses the PBFT consensus algorithm, there is no need to additionally consider the differences in computing power of the nodes themselves. For the assignment of nodes in the initial sharding stage, in order to avoid the possible 1% attack due to the dispersion of computing power (that is, due to the sharding operation performed in the system, the computing power of some shards may be weak due to the uneven distribution of nodes, in which case a malicious node only needs to control a small number of nodes in the shard in order to make the entire shard fail), all nodes need to be evenly distributed across all the shards. For the node redistribution, when the number of shards changes, we must reduce the number of nodes that need to be remapped for the sake of ensuring the randomness of the node assignment.

A consistent hashing algorithm can effectively solve the abovementioned problem. These methods use the irregularity of the calculation result of the hash function to achieve the purpose of randomness. They first perform a hash operation on a key feature value of a node to take a mode, and then divide the nodes into corresponding shards based on the result. The existing consistent hashing algorithms are mainly hash ring, the jump consistent hashing algorithm [24] and the Maglev [25] hashing algorithm. The performance of the three methods is shown in the Table 1, where "+" indicates a better performance, "−" indicates a poor performance, and "O" indicates an average performance [26].

Table 1. Performance comparison of existing consistent hash algorithms.

Algorithms	Balance	Minimal Remapping	Complexity	Weighted Mappings
Hash Ring	O	+	$\sum_{i=2}^{n} \frac{1}{i}$	+
Jump Con Hash	+	+	$\sum_{i=2}^{n} \frac{1}{i}$	+
Maglev-Hash	+	−	$\sum_{i=2}^{n} \frac{n}{i}$	+

Gal proposed a new hashing algorithm, Anchorhash [26], in 2021, which differs from other hash consistency algorithms in that its decision depends on past events of the system. Its general principle is shown in Figure 3.

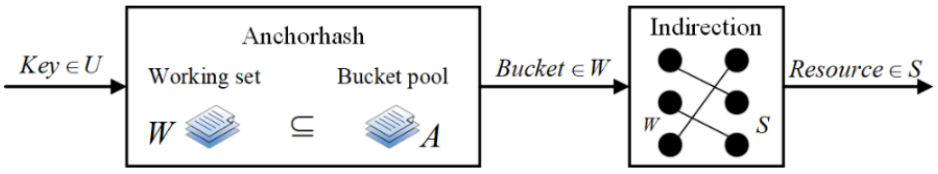


Figure 3. Anchorhash schematic diagram.

After establishing a connection with the sharding blockchain, the working set can be understood as sharding. The increase and removal of the working set can be understood as the increase and decrease of the number of shards, and the key can be understood as the characteristics of each transaction. However, in sharding blockchains, if we only use Anchorhash, we are essentially only relying on previous events to determine the future transactions. The probability of the entire system being attacked may be greatly increased. Therefore, according to the characteristics of sharding blockchain systems, this study combined the jump consistent hash algorithm and the Anchorhash algorithm to improve the node assignment in three aspects.

3.1.1. Initialization Phase

The study reported in [27] introduced the mapping method and the remapping method brought about by the change of the working set, but it did not explain the initialization of the algorithm. In fact, we can roughly demonstrate the initialization of the Anchorhash algorithm, as shown in Figure 4.

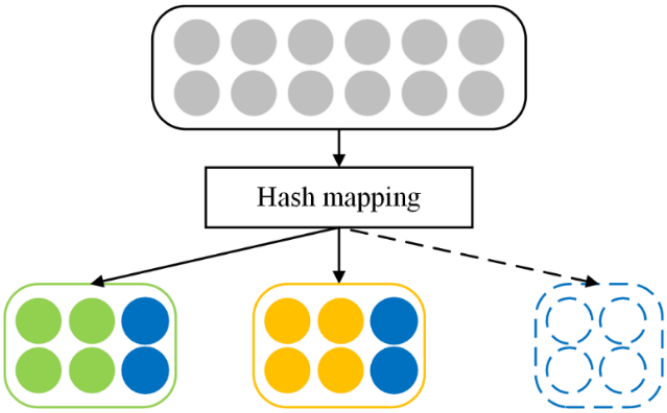


Figure 4. Initialization phase.

Taking Figure 4 as an example, 12 nodes in the system are evenly mapped to 3 different working sets through the hash operation, because working set 3 is not in use in the initial stage. Therefore, the corresponding node mapped to working set 3 needs to be mapped to the remaining two working sets that are working normally through the hash operation again. If working set 3 is enabled at a certain point, it is only necessary to remap the corresponding node to this shard according to the historical information.

However, in a sharding blockchain, the total number of shards is not fixed, and it is necessary to comprehensively consider the number of nodes and the actual situation of the communication. Therefore, we must use the idea of random number generation of proof of vote (PoV) to realize the random mapping of all nodes.

First, we use the node's own signature $sign(i)$ and the timestamp $time(i)$ corresponding to the node to perform the XOR calculation in order to obtain the random number source R_{source}:

$$R_{source} = time(i) \oplus sign(i) \tag{1}$$

Suppose that the method of intercepting the last 32 bits of the string is $SubStringEnd32(string)$, and a random number R is obtained, where N represents the number of shards:

$$R = StrToInt(SubStringEnd32[Hash(R_{source})])modN \tag{2}$$

Based on this, the random mapping of all nodes can be assigned according to the number of shards in the current system, avoiding the situation of secondary mapping. On the other hand, because of the differences between the nodes' own signatures and the unpredictable mapping time, R_{source} is unpredictable, and R also has a strong randomness. In this case, all nodes can be distributed to all shards more effectively and evenly, so that the computing power of all the shards is equal and the risk of them being attacked is reduced.

3.1.2. Changes in the Number of Shards

Anchorhash deals with increasing or decreasing numbers of working sets by introducing stack Z. Stack Z records the deletion order of different ordinal working sets. For example, when $Z = \{3 \leftarrow 2 \leftarrow 1\}$, this means that the chronological order of the deletion of the working set is 1, 2, and 3. Therefore, when the number of working sets changes, it will restore the corresponding working sets in turn, according to the order in which the

previous working sets were deleted. In the above example, the restoration order of the working sets should be 3, 2, and 1.

Although the abovementioned method can minimize the extent of remapping caused by the changes, its strong regularity enables malicious nodes in the system to easily complete attacks by grasping onto the changing laws of sharding, which may pose a great threat to the security. Based on this, this paper introduces the concept of the timestamp on the basis of Anchorhash. We used the randomness of time to solve the security problems involved in the application of Anchorhash to the blockchain, so that the change in the number of shards is no longer fixed, and the predicted attack is avoided. We assumed that all nodes are eventually evenly distributed to all shards; that is, the computing power of all shards is equal. The following is a brief description of the process of the deletion and addition of the shards:

- Deletion of shards:

  The decision about whether or not to delete a shard needs to be determined according to the order of three factors: the number of transactions in the system, the cross-shard communication, and the work efficiency of the shard.
  First, we should consider the number of transactions E in the system. Assuming that the number of transactions that each shard N_i can process at the same time is Ca, when the condition of $E < (N-1)Ca$ is satisfied, the operation of deleting shards needs to be performed in the system. Then, it is necessary to count the number of cross-shard communications between shards within a period of time and filter out one or more groups of shards with the most frequent cross-shard communications. Finally, we must compare the working efficiency of each shard in one or more sets of shards and delete the shard with the least work efficiency.

- Addition of shards:

  If Z exists, the added shards are randomly selected in Z. Assuming that shards need to be added at time τ, take $Z = \{x \leftarrow y \leftarrow z\}$ as an example:
  First, obtain $Z = \{x(0) \leftarrow y(1) \leftarrow z(2)\}$, according to the order of the elements in Z.
  Then, use the formula to calculate the shards that need to be added:

$$Add\ shard = StrToInt(SubStringEnd16[Hash(\tau)])mod((length(Z)) \quad (3)$$

  If Z does not exist, add any shard at random.

3.1.3. Node Remapping

Anchorhash uses historical data to realize the node remapping. Although this method can minimize the extent of the remapping, it is not suitable for blockchain systems. In addition, Anchorhash's treatment of the node remapping is not comprehensive. It only considers the remapping caused by the increase or decrease of the working set when the stack exists.

Based on this, considering the security of the system and the complexity of remapping, we adopted the jump consistent hash function to perform the remapping operation of the nodes, regardless of the existence of Z. Compared with the traditional algorithm, this method not only fully considers all possible situations in the process of node remapping, but also works through the combination of the two algorithms. Although the complexity of the time of the remapping process is increased to a certain extent, it can effectively solve the security problems involved in Anchorhash.

The algorithm of the jump consistent hash function is shown in Algorithm 1:

Algorithm 1 Jump Consistent Hash Algorithm

Input: Node number j; shard number i
Output: Shard number i'
Int JumpConHash (int j, int i)
1: random. seed (j)
2: int a = -1, i' = 0
3: **while** $b < i$;
4: a = i'
5: double r = random.next()
6: i' = floor((i'+1)/r)
7: **return** i'

The JumpConHash function returns the number of the corresponding shard after the node is remapped. If it is assumed that the nodes are evenly distributed after the remapping, we can draw the following conclusions:

- When $i = 1$, it returns $i' = 0$, which means that all nodes are mapped to the shard numbered 0;
- When $i = 2$, theoretically, each shard needs to have $j/2$ mappings; that is, $j/2$ nodes are required for remapping;
- When i changes from n to $n + 1$, there are $j/(n + 1)$ nodes in the whole system that need to be remapped.

Next, we will discuss the nodes that need to be remapped.

When the number of shards decreases, no matter whether the stack exists, we only need to remap the nodes in the shard to be deleted, so that they are evenly distributed in the remaining shards. When the number of shards increases, if the stack does not exist, we should perform the remapping operation on all nodes, and randomly remap some nodes to the newly added shards, according to the ratio of active nodes to inactive nodes, 1:1. Otherwise, we should first use (3-3) to determine which shard will be restored, and the corresponding active nodes will be directly transferred back to the shard. After that, all the other nodes that have been remapped due to the reduction in the number of shards in the previous stage need to be remapped again, until the members of the newly added shards reach the expected value.

Since the intra-shard consensus protocol is the PBFT consensus algorithm, we can ignore the problem of the different computing powers of the nodes in the system. According to the preconditions of PBFT, it is necessary to ensure that the total number U in the node system satisfies $U \geq 3f + 1$, where f represents the number of malicious nodes in the system. When all nodes are allocated to i shards, according to the method shown above, due to the randomness of the assignment, we cannot be sure that the number of internal nodes in each shard satisfies $U_i \geq 3f_i + 1$. Thus, there may be a scenario where the Byzantine nodes in some shards exceed the threshold. The case in which the Byzantine nodes exceed the threshold is depicted in Figure 5, assuming the same number of nodes in all the shards.

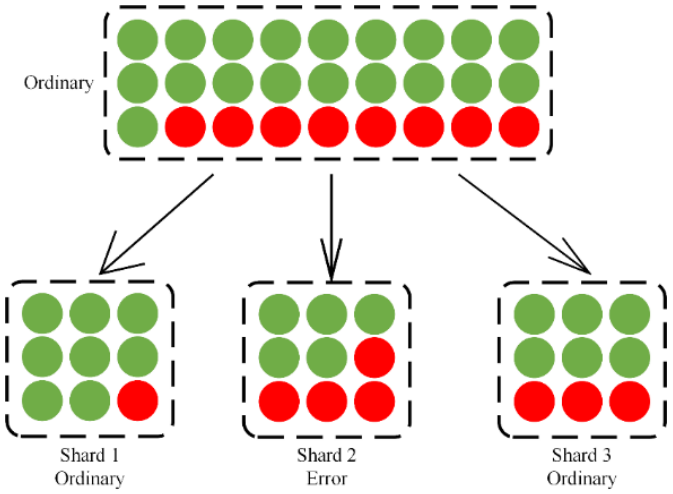


Figure 5. Risks of shard failure. Figure 5 shows that there is a certain probability that the proportion of Byzantine nodes in the shards will exceed 1/3 after sharding. In Figure 5, after the node assignment, the number of Byzantine nodes in shards 1 and 3 exceeds 1/3, resulting in a failure to reach a consensus within the shards. Section 4.2 discusses the relationship between the effectiveness of the intra-shard consensus and the number and size of the shards.

3.2. Cross-Shard Transaction Processing

When committees exist in the shards, the most common ways to handle cross-shard transactions are the two-phase commit (2PC)-based and transaction split-based methods. Compared to the former one, the latter only considers the multi-input and single-output situation. In fact, some transactions may include multiple outputs. In this case, if the multi-input and multi-output situation is split into a single-input and single-output, the whole process will be very complicated. In addition, transaction splitting results in an increase in the number of transactions, which in turn greatly increases the processing and storage overhead of the entire system. Based on this, in this paper, we improved the traditional method based on a two-stage submission (as shown in Figure 6). The improved method is mainly composed of four parts: the transaction processing stage, confirmation stage, audit stage, and submission stage. The specific process is shown in Figure 7.

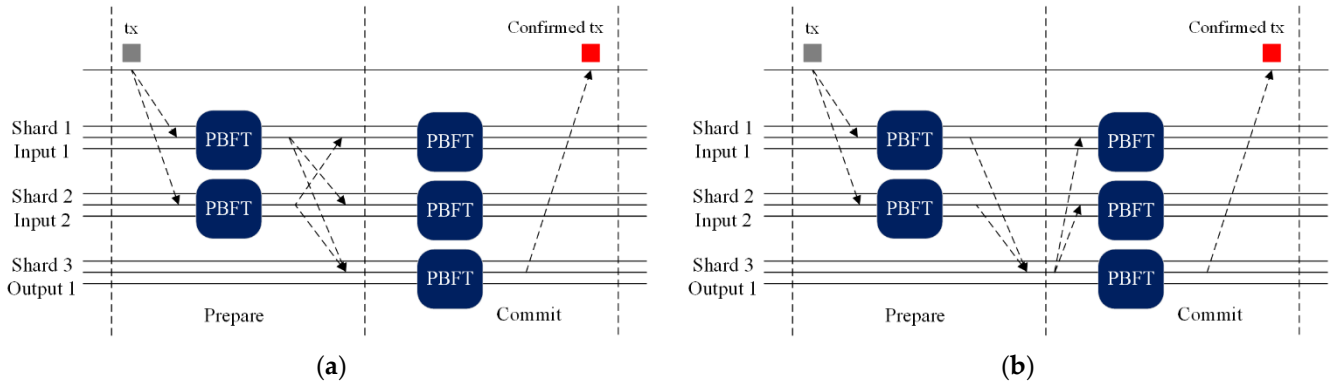


Figure 6. The flowchart of the shard-driven basic 2PC. (**a**) All input shards as the coordinators. (**b**) All output shards as the coordinators.

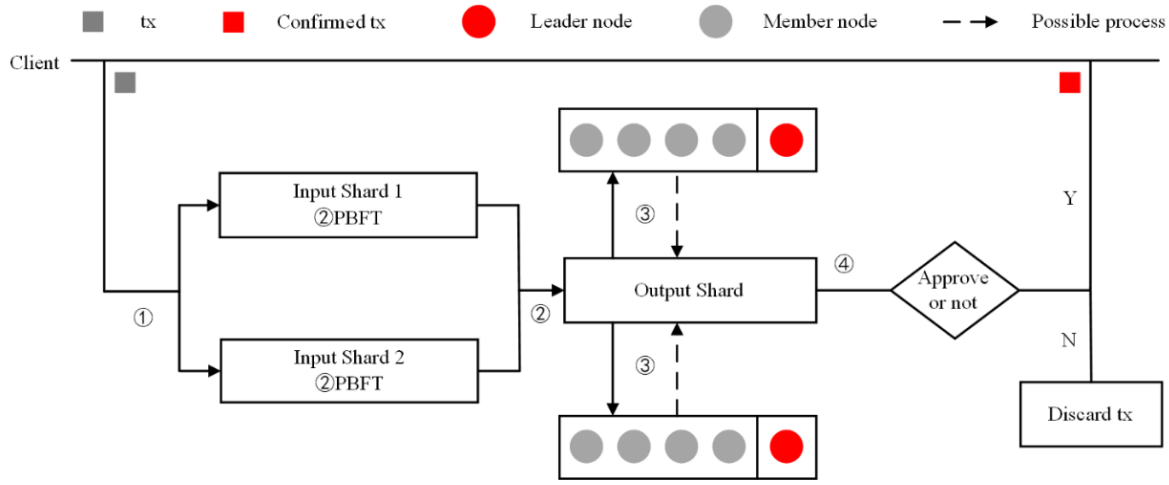


Figure 7. Cross-shard transaction processing based on four phases.

- **Transaction processing phase.** In the transaction processing phase, the client sends the transactions to be confirmed to the corresponding input shard.
- **Confirmation phase.** The confirmation phase mainly completes the verification and submission of transactions by the input shards. After each input shard receives the transaction sent by the client, the intra-shard consistency protocol PBFT is triggered to verify the validity of the transaction. After the audit is completed, the master node of each shard will send the verified information to the output shard of the corresponding transaction in the form of $<tx, T, C, i, 1>$. If the verification fails, a message in the form of $<tx, T, C, i, 0>$ is sent. In this message, tx represents the summary information of the transaction processed in the shard, T represents the time when information is sent, C represents the set of numbers of all the input shards responsible for processing the transaction, i represents the number of the shard, and "1" represents the verification passed, while "0" means the verification failed.
- **Review phase.** The output shard first classifies the received information, according to C, and checks whether the verification information of the corresponding transaction is consistent (whether it is "1"), according to the classification result. It then calculates the sending time interval Δt of each set of information, $\Delta t = maxT - minT$, and compares it with the theoretical detection delay ΔT. If $\Delta t > \Delta T$, the warning information will be sent to the corresponding input shard, and the corresponding shard will review all the nodes; otherwise, the output shard processes all the received information and sends the processed information $<Confirmed\ tx, T>$ to all nodes except the master node in all input shards. After the node in the input shard receives the information sent by the output shard, it carries out a final confirmation of the information. If the information is correct, it will not give any response, or it will reply $<$Error, $i>$.
- **Submit phase.** The output shard starts timing after sending $<Confirmed\ tx, T>$. If the information sent by the relevant node is not received within $2\Delta T$, $Confirmed\ tx$ will be submitted to the client, and the cross-shard communication will be completed. If an $<$Error, $i>$ message is received, the number of received messages needs to be counted. When the amount of information received by the output shard is greater than 1/2 of the amount of the information sent, or the amount of information received from a shard is greater than 1/3 of the amount of information sent to the shard, the output shard discards the transaction. In other cases, the output shard will also commit the $Confirmed\ tx$ to the client.

Comparing Figures 7 and 8, it can be seen that the newly proposed method of cross-shard transaction processing not only avoids the high communication complexity caused by the multiple calls of the intra-shard consistency algorithm in the traditional algorithm,

to a certain extent, but also can better cope with various possible malicious attacks and situations where the master node is a malicious node.

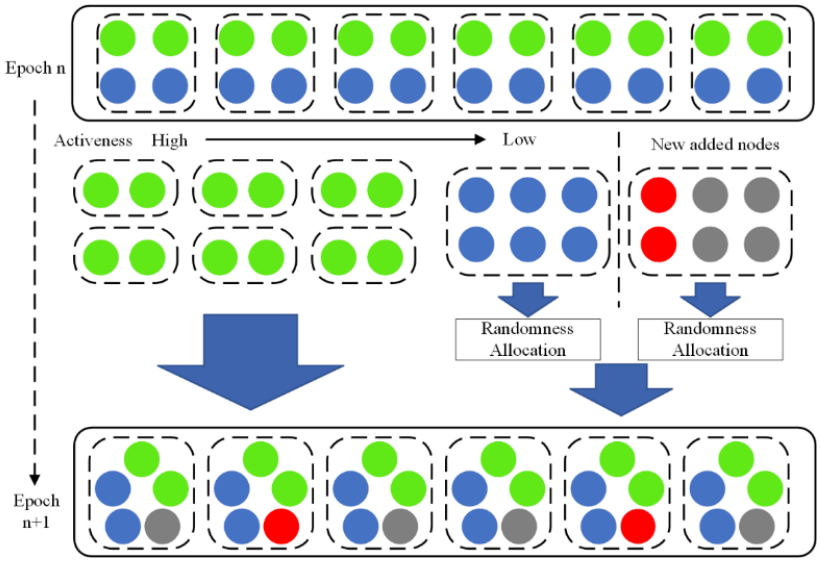


Figure 8. Shard reconfiguration process.

3.3. Shard Reconfiguration

Sharding blockchain systems require periodic reconfigurations of the system, including the adjustment of the members and the transfer of states between shard members after the reconfiguration. This is because, if the original structure is stable, attackers can easily launch attacks on particular nodes after a period of observation, thus allowing the proportion of nodes controlled by the adversary to exceed a predefined security threshold, e.g., 1/3 of the PBFT. This will directly lead to the failure of the corresponding slice, and further destroy the vitality and security of the whole system. Based on this information, we propose a shard reconfiguration method based on the activity of the nodes' participation in the transaction. The general process is shown in Figure 8. The main flow is as follows.

3.3.1. Node Activity Determination and Node Classification

First, we used the following formula to calculate the node activity (A_j):

$$A_j = \alpha \frac{M_j}{M} + \beta \frac{m_j}{M_j} \tag{4}$$

where j represents the node number; M indicates the total number of transactions to be processed in the consensus phase; and M_j denotes the number of transactions that node j is involved in. In the equation, $M_j \geq \left\lceil \frac{1}{2} M \right\rceil$, m_j indicates the number of transactions that passed authentication, and α, β are weight parameters, $\alpha + \beta = 1$.

Then, we arranged A_j in descending order according to the results, where half of the modes with higher activity are active nodes, and the remaining nodes are inactive nodes. Compared with the traditional methods, the node activity in this study not only depended on the transactions but was also related to the validity of these transactions. The main reason for this is that, even if a particular node is involved in all transactions, if none of these transactions pass the validation, all the work is invalid, which not only causes a waste of resources but also, on the other hand, means that the node is not essentially involved in any transaction.

In addition, in principle, the nodes within the shard prefer to process transactions with a higher probability of passing the validation, in order to obtain higher payoffs, and not to choose those are doubtful. Therefore, we not only need to pay attention to the size

of $\frac{m_j}{M_j}$, but must also consider the relationship between $\frac{M_j}{M}$ and $\frac{m_j}{M_j}$. Specifically, all nodes that are defined as active must satisfy $\frac{m_j}{M_j} \geq max\left(\frac{M_j}{M}, 50\%\right)$. Those that have a lower $\frac{m_j}{M_j}$ cannot be listed as active nodes and, subsequently, need to be further observed.

3.3.2. Shard Reconfiguration

Active nodes are those that are actively involved in transaction processing and have a high transaction verification rate. Thus, we did not change the active nodes in the previous stage for the following reasons. Firstly, we can determine that the active node is not malicious and has a positive effect on the transaction processing in the shard. Retaining it can ensure the security of the system, to a certain extent. Secondly, active nodes tend to participate in processing more transactions, and keeping them in the corresponding shard can ensure the stability of the initial shard transaction processing in the next stage. Finally, when new nodes are added to the shard, these nodes can serve to defend possible corruption attacks [28].

Inactive nodes are those that are not actively involved in transaction processing or have a low validation rate for the transactions they are involved in, so that they will be reallocated to a shard by random assignment after one particular stage. The reasons for this are as follows: firstly, inactive nodes are often more vulnerable to corruption attacks due to their low participation, and, therefore, random arrangement can ensure the real-time changes in the stored data in such nodes and thus ensure the security of the shard; and secondly, this process can cause nodes to participate in transaction processing in different shards and stimulate the nodes to participate in transaction processing through the differences in the motivation mechanism.

For the newly joined nodes, the allocation is performed in the same way as that of the inactive nodes. The reasons for this are as follows: firstly, we cannot be sure whether all these nodes are honest and, therefore, random allocation can disperse all nodes to the respective shards in order to minimize the proportion of malicious nodes; and secondly, the nodes do not have access to prior data related to the shard, and thus they are more vulnerable to corruption attacks by malicious nodes. Random assignment can reduce the proportion of nodes that are subjected to corruption attacks and ensure the security of the shard.

3.4. Motivation Mechanism

The motivation mechanism of the blockchain is used to encourage nodes to participate in the protocol. In general, nodes participating in the protocol need to consume a certain amount of communication bandwidth and computing power [29], and if they do not receive the corresponding rewards, the nodes will lose the incentive to participate in the protocol. Therefore, it is necessary to establish a motivation mechanism to maintain the liveness of honest nodes in the system.

Since the sharding blockchain system is more complex than the traditional blockchain, the setting of the motivation mechanism not only needs to consider the reward and punishment mechanisms, but also needs to comprehensively consider different shards and different node types. Since there are few existing studies on the motivation mechanisms of sharded blockchains, this paper briefly introduces the precautions for setting the motivation mechanism of sharding blockchains from two perspectives:

- **Motivation mechanism between shards.** The number of rewards obtained by the same type of shard is determined by the efficiency of its own transaction processing. The higher the efficiency, the more bonuses can be shared under certain conditions. The rewards received by different types of shards should be determined by the total number of transactions processed by the shard. In principle, for a transaction that needs to be processed across shards, the total number of rewards for all input shards and all output shards should be consistent.

- **Motivation mechanism within shards.** The motivation mechanisms of different nodes in the shard need to comprehensively consider the identity of the node in the intra-shard consistency protocol, the enthusiasm of the node to participate in the transaction, and the proportion of the corresponding transaction that passes the verification. Normally, nodes with different identities will receive different rewards after successfully processing a certain transaction. The more important their role is, the more rewards they will obtain. For nodes with the same identity in shards, the number of rewards they receive has a certain relationship with their activity and efficiency in participating in transactions, as well as the probability of the transactions passing verification.

In addition, for those malicious nodes, if there are no malicious behaviors involved in the transaction processing, they will be rewarded according to the motivation mechanism of honest nodes. Otherwise, malicious nodes will be punished and be included in the warning list, and it will be hard for them to participate in the transaction processing for a certain period of time.

4. Experiment Analysis

This section is divided into three main parts: Section 4.1 verifies the security and liveness of the proposed algorithm based on three types of five attack models; Section 4.2 studies and analyzes the influence of the number of shards and the scale of shards on the effectiveness of the shards when the intra-shard consistency protocol is PBFT; and Section 4.3 analyzes the performance of the algorithm.

4.1. Security Analysis

This section mainly analyzes the security and liveness of the algorithm from three perspectives: the 1% attack, corruption attack, and Byzantine node attack. Here are brief introductions of these three attack models:

- **The 1% attack model.** With the system divided into different shards, the 1% attack is caused by the uneven computing power of the shards; that is, the malicious node only needs to control a small part of the nodes in the shard to cause the entire shard to be invalid, thereby threatening the security of the entire system.
- **Corruption attack model.** This mainly refers to the scenario where attackers destroy honest nodes. Most of the existing sharding blockchains mainly account for mild corruption attacks; that is, the attacker needs a certain period of time to complete the destruction of the nodes, and the target node remains honest during this time. In this paper, we mainly considered attacks on newly joined nodes and member nodes.
- **Byzantine node attack model.** This refers to the phenomenon whereby Byzantine nodes disrupt the consensus process by acting maliciously and forging information. In this paper, we considered the case where the master node and other nodes within the shard are Byzantine nodes separately.

Next, we analyzed the security of the system in detail, based on our proposal and these attack models.

4.1.1. The 1% Attack Model

Malicious nodes often achieve a 1% attack by controlling the shard with the least arithmetic power. Once any shard is controlled, the security of the whole system will be threatened.

In order to deal with this problem, we combined the jump consistent hash algorithm with the improved Anchorhash algorithm to accomplish the process of node assignment under the assumption that all nodes have the same arithmetic power. This method effectively reduces the probability of a 1% attack in two ways:

- **Initialization phase.** We first used the specificity of each node's entry time into the system and the node's own signature to generate a random number belonging to

each node, then we used this number to complete the assignment of the nodes. The randomness of these numbers ensures that nodes are assigned randomly. Thus, there is no over-concentration of computing power in one shard.

- **Remapping and changes in the number of shards.** We improved the Anchorhash algorithm to enable it to cope with the change in the number of shards, because it only uses historical data calls to respond to bucket changes. This is very dangerous for the blockchain, because once malicious nodes obtain these data, they can easily launch an attack. Therefore, we combined the concept of timestamps and the jump consistent hash algorithm to ensure the security of the system. Combining the randomness of time with the variability of the number of shards makes it difficult for malicious nodes to predict the upcoming addition or removal of shards from the system. The jump consistent hash algorithm not only ensures that, when one shard changes, all nodes that need to be remapped are evenly distributed to the remaining shards, but it also minimizes the number of remapping actions.

4.1.2. Corruption Attack Model

Here, we introduce the concept of node activity, having set up a shard reconfiguration method to solve the attacks on newly joined nodes and member nodes:

- **Node activity.** According to the greedy algorithm, all honest nodes aim to receive the highest reward, and all attackers aim to destroy the entire system. Therefore, the nodes with higher activity are often honest nodes. Based on this, we established the node activity index and used the nodes with higher activity to guide the newly added nodes, which can avoid corruption attacks to a certain extent.
- **Shard reconfiguration.** We divided all nodes into two categories according to their activity, and periodically remapped all inactive nodes in the system. By reconfiguring the shards, it becomes difficult for a malicious node to anticipate the composition of the next stage of the shard in advance, so that the probability of mild corruption attacks is reduced.

4.1.3. Byzantine Node Attack Model

We used the PBFT consensus algorithm as the intra-shard consistency and improved the traditional 2PC method to solve the problem of possible Byzantine node attacks. Since PBFT itself can withstand Byzantine node attacks to some extent, here, we will only discuss the case where the master node is a Byzantine node.

When the master node is a Byzantine node, the following two scenarios may occur. One option is that the master node receives information from other nodes in the shard and then deliberately does not send the information, causing the failure of the consensus of the whole system. This kind of behavior can be easily detected. The other option is that the master node modifies the received information privately, which in turn leads to the failure of the whole system. This kind of behavior cannot be easily detected. Thus, we added a "review phase" and "submit phase" to the traditional 2PC method to allow all nodes in the shard except for the master node to confirm the authenticity of the information once again.

In order to further test the effectiveness of the method, we analyzed the detection results of the algorithm, as described in this paper, and the traditional 2PC through 50 experiments, in which the leader node was malicious and the other nodes in the shard were malicious in 10 cases each. The experimental results are shown in Table 2.

It can be seen from the data in the table that the algorithm proposed in this paper exhibited a certain improvement in the total detection accuracy compared with the traditional 2PC algorithm. In particular, the existence of the audit phase greatly improved the accuracy of the algorithm in detecting malicious master nodes. However, although the algorithm proposed in this paper accurately detects the malicious behavior of all the other nodes, there are cases in which valid transactions are identified as invalid transactions. It is difficult to finally determine whether the transaction is valid, resulting in misjudgment.

Table 2. Security analysis of the cross-shard transaction processing phase.

Algorithms	Leader Node Malicious	Other Nodes Malicious	Leader Node Malicious Detection Accuracy	Other Nodes Malicious Detection Accuracy	Error	Total Accuracy
2PC	7	9	70%	90%	4	80%
Improved 2PC	9	11	90%	100%	2	90%

4.2. Shard Effectiveness Analysis

We assumed that the proportion of Byzantine nodes existing in the system is R, where $0 \leq R < 1/3$. We defined L as the number of nodes in a shard. It was stipulated that all nodes in the shard will be verified according to their own identity (Byzantine or non-Byzantine) in the consensus phase.

Let $X = \sum_{i=1}^{L} X_i$, where X_i represents the identity of the node as a Byzantine node, and X represents the total number of Byzantine nodes in the current shard. The literature [12] designates the probability distribution of X as a binomial distribution, as shown in (5), and considers that the probability of honesty or maliciousness in each node selection remains unchanged, but this is only applicable to an infinite number of nodes. In this case, there are certain limitations.

$$P(X < \left\lfloor \frac{L}{3} \right\rfloor) = 1 - P(X \geq \left\lceil \frac{L}{3} \right\rceil) = 1 - \sum_{X=\lceil \frac{L}{3} \rceil}^{L} C_L^x f^x (1-f)^{L-x} \tag{5}$$

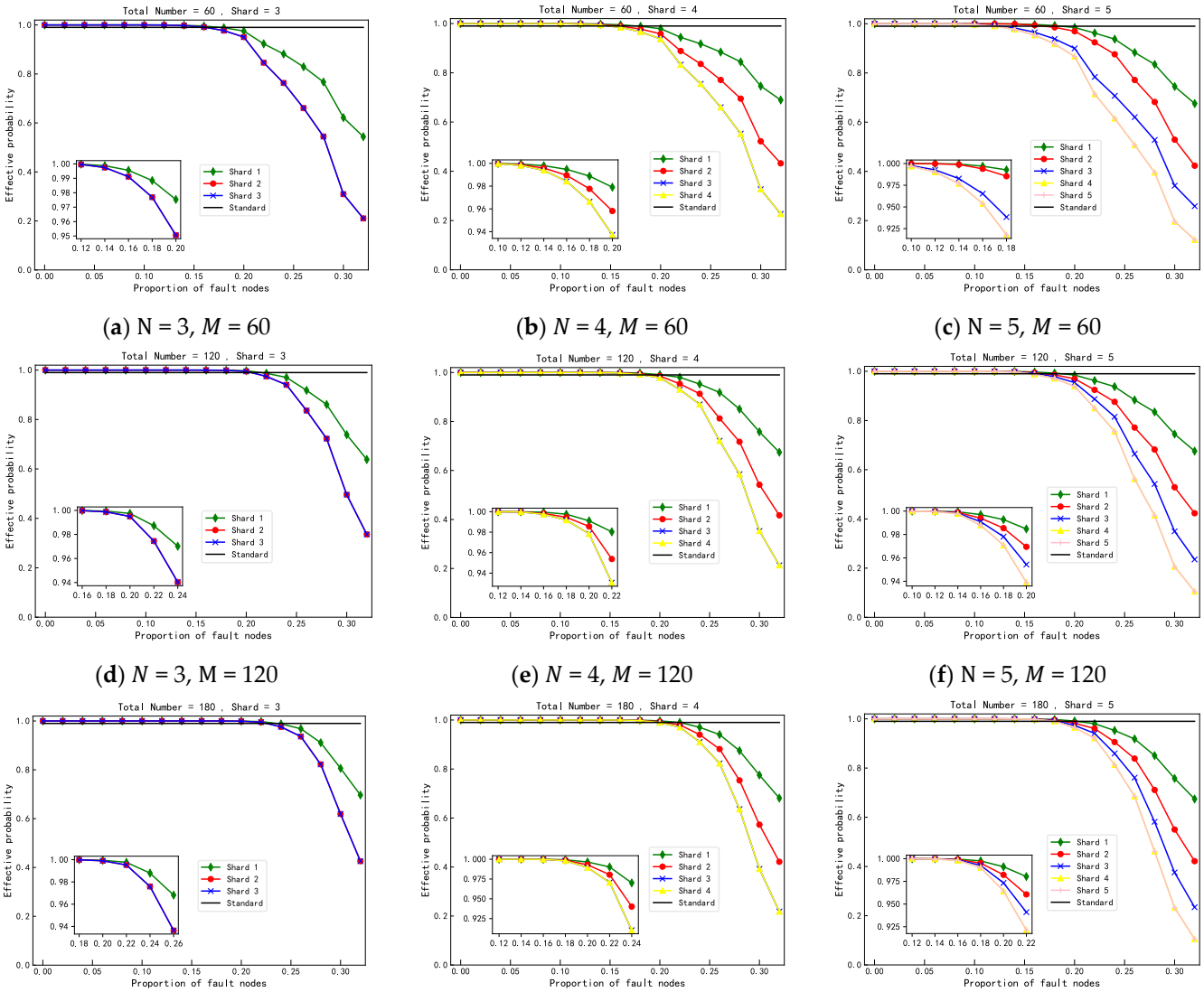
In fact, due to the limited total number of nodes, each node selection affects the ratio of honest nodes to malicious nodes in the system. Based on this, we adopted a hypergeometric distribution that is closer to the actual situation to calculate the probability distribution of X. This is shown in Formula (6), where U represents the total number of nodes in the system and quantifies the influence of the number of nodes in the shard and the number of shards on the effectiveness of the shard:

$$P(X < \left\lfloor \frac{L}{3} \right\rfloor) = 1 - P(X \geq \left\lceil \frac{L}{3} \right\rceil) = 1 - \sum_{x=\lceil \frac{L}{3} \rceil}^{L} \frac{C_{\lfloor UR \rfloor}^{x} C_{U-\lfloor UR \rfloor}^{L-x}}{C_U^L} \tag{6}$$

Based on the experimental results reported in the literature [12], we firstly set the number of shards $N = \{3, 4, 5\}$ and the total number of system nodes $M = \{60, 120, 180\}$, and then, based on the fault-tolerant performance of the PBFT algorithm, three experiments were performed. The experimental results are as follows:

Experiment 1: Effectiveness Analysis of a Single Shard.

The literature [30] pointed out that the validity of each shard in a sharding blockchain system must be greater than 99%. Based on this, we conducted a comprehensive analysis of nine possible situations and compared the results with the threshold of 99%. The experimental results are shown in Figure 9.



(**a**) N = 3, M = 60 (**b**) N = 4, M = 60 (**c**) N = 5, M = 60

(**d**) N = 3, M = 120 (**e**) N = 4, M = 120 (**f**) N = 5, M = 120

(**g**) N = 3, M = 180 (**h**) N = 4, M = 180 (**i**) N = 5, M = 180

Figure 9. N, M impact on individual shard validity. Among which N represents the number of shards and M represents the total number of the nodes.

It can be seen from a single graph that, in the same situation, the effectiveness of the sharding is related to the speed of the completion of the node allocation in the sharding. The slower the completion speed of the sharding is, the more easily it is affected by the proportion of Byzantine nodes in the system. Through a horizontal and vertical comparison of experimental results, we can observe that the effectiveness of each shard is related to the number of nodes in the system, the number of shards, and the Byzantine ratio in the system. The more nodes there are in a shard, the more the shard can tolerate a higher proportion of Byzantine nodes in the system. On this basis, we also conducted additional comparative experiments for cases where the total number of nodes and the number of shards in the system are different, but the number of nodes in the shards is the same. There were also some differences in the effectiveness.

Table 3 lists the specific values for which all shards are valid in the critical cases in the nine graphs above.

Table 3. Critical values of fragmentation effectiveness under different conditions.

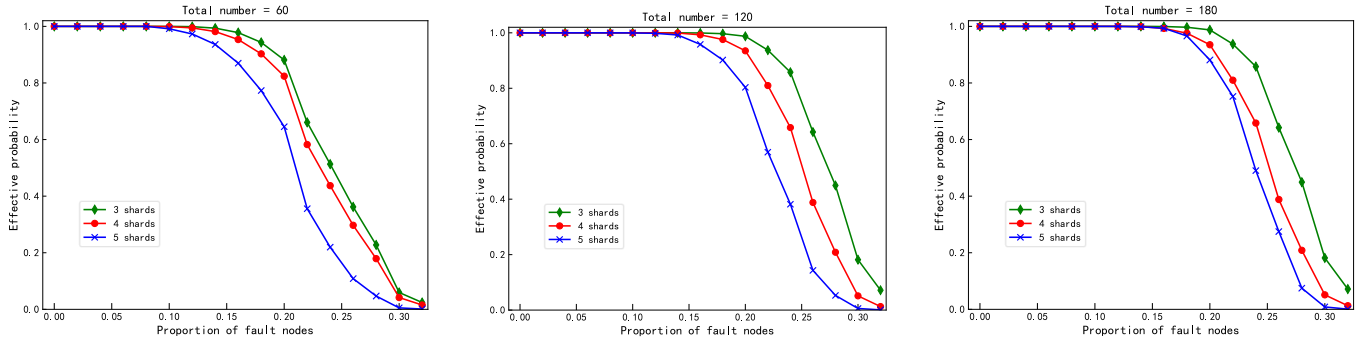
Numbers	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
N	3	3	3	4	4	4	5	5	5
M	60	120	180	60	120	180	60	120	180
P_{min}	0.9911	0.9949	0.9928	0.9938	0.9917	0.9980	0.9903	0.9977	0.9901
R	0.16	0.2	0.22	0.14	0.18	0.18	0.12	0.14	0.18

Experiment 2: Analysis of the overall effectiveness of the system based on the scale of shards.

Experiment 3: Overall effectiveness analysis of the total number of nodes in the system These three figures respectively describe the probability that the entire system is effective when the total number of nodes is the same and the number of shards is different.Among which (a) The total number of nodes is 60; (b) The total number of nodes is 120; (c) The total number of nodes is 180.

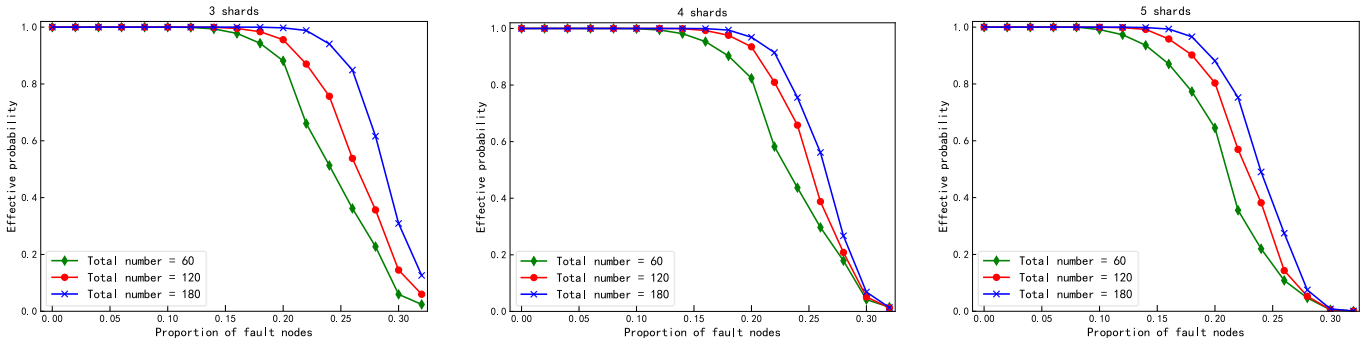
Based on Figures 10 and 11, we can reach the following conclusions:

1. Under certain circumstances, the effectiveness of the entire system increases with the increase in the number of nodes in the shard. When a certain limit is reached, the effectiveness of the system will not further increase. The main reason for this is that, when the proportion of Byzantine nodes in the system is acceptable, and if the number of nodes in the shard reaches a certain condition, the effectiveness of each shard almost approaches the maximum value. In this case, further increasing the number of nodes in the shard may not optimize the effectiveness of the overall system. However, it may lead to an exponential increase in the complexity of the intra-shard communication, which is not conducive to the efficiency of the entire system in processing transactions.
2. Under the conditions set out in this paper, when the proportion of Byzantine nodes in the system is below 0.2, in most cases, the effectiveness of the system is basically guaranteed. When it exceeds 0.2, the effectiveness will drop rapidly, even if the Byzantine proportion of the entire system is satisfied. We can also see that, even if the ratio does not exceed 1/3, the effectiveness of the entire system is close to 0. Thus, differently from the traditional blockchain system, in the sharding blockchain system there are stricter requirements for the proportion of Byzantine nodes, which cannot only be satisfied with the theoretical upper limit.



(**a**) M = 60 (**b**) M = 120 (**c**) M = 180

Figure 10. Analysis of the overall effectiveness of the system based on the scale of shards.



(**a**) $N = 3$ (**b**) $N = 4$ (**c**) $N = 5$

Figure 11. Overall effectiveness analysis of the total number of nodes in the system. These three figures respectively describe the probability that the entire system is effective when the number of shards is the same and the total number of nodes is different. Among which (**a**) The number of shard is 3; (**b**) The number of shard is 4; (**c**) The number of shard is 5.

It can be seen from the above analysis that, in a sharding blockchain system where the intra-shard consensus protocol is PBFT, regardless of the scale of the entire system, the primary goal of sharding is to maximize the ratio of sharding to the Byzantine nodes in the system in order to ensure the validity of all shards. In addition, in the process of determining the size of each shard, it is necessary to fully consider the communication complexity of the PBFT on-chip consensus protocol, and in order to ensure the effective completion of the transaction processing, the efficiency of the transaction processing should be improved as much as possible.

4.3. Performance Analysis

In this section, we first compare our algorithm with the classic sharding consensus algorithm ELASTICO. The ELASTICO algorithm is mainly composed of five parts: committee node confirmation, shard configuration, intra-shard consensus, epoch random number generation, and shard reconfiguration. The comparison mainly includes three aspects: the committee node confirmation, shard configuration, and cross-shard communication:

- **Committee node confirmation.** ELASTICO uses PoW scheme to confirm the committee nodes by mining, which causes the system to spend too much unnecessary time waiting for the nodes solving the PoW scheme to fill up the committee. In contrast, the algorithm proposed in this paper is to confirm the member nodes by the positivity of the nodes, which saves time to some extent.
- **Shard configuration.** ELASTICO reduces the communication complexity of the shard composition through a predefined directory committee. Its communication complexity is proportional to the directory committee. The proposed algorithm uses the jump consistency hash function and Anchorhash algorithm. The experimental results are shown in Figure 12. With the proposed algorithm, the time complexity exists between the jump consistency hash function and Anchorhash algorithm and is slightly higher than ELASTICO, because this method reduces the amount of storage of historical data and memory usage by distinguishing the node types and takes advantage of the low time complexity of the jump consistency hash algorithm in remapping to achieve the average mapping of all nodes. In general, the algorithm ensures the uniformity of the arithmetic power across all shards and the security of the whole system at the expense of some communication complexity.
- **Cross-shard communication**. The ELASTICO protocol divides the set of transactions into subsets of transactions that do not intersect with one another, avoiding a great deal of cross-shard communication. Once cross-shard communication is involved, the performance of the ELASTICO algorithm will drop dramatically. We set up a more

comprehensive cross-shard communication process that fully considers Byzantine attacks, ensuring the effectiveness of the system to a large extent.

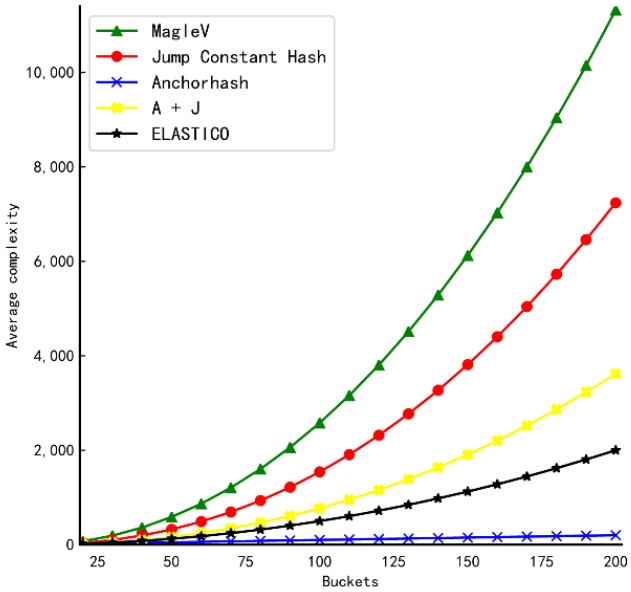


Figure 12. Average time complexity.

Then, we compared our algorithm with the traditional sharding consensus algorithms and identified three advantages of our algorithm:

- It further limited the number of Byzantine nodes allowed in the system. This can provide a standard for other sharding consensus algorithms. It can also reduce the resource consumption caused by consensus failure, to a certain extent.
- It can handle the situation where the leader node in the shard is a malicious node. Thus, it can cope better with the possible monopoly phenomenon and disputes of interest in various applications.
- The addition of the signature and node activity better ensures the security of the algorithm. At the same time, it also provides new ideas for the further application of sharding consensus algorithms in the field of privacy protection.

5. Conclusions

In this paper, we proposed a new and effective sharding blockchain consensus mechanism based on the problems of existing sharding technologies. In regard to the dynamic management of the number of sharding nodes, we proposed a method based on the signature Anchorhash and jump consistency hash function, which realizes the minimum remapping and reduces the average time complexity to ensure the system's security. In terms of cross-shard transaction processing, a relatively complete cross-shard transaction scheme was designed. This scheme not only considers the malicious attacks that may be launched by the internal nodes of the shard, but also pays attention to the possible malicious behavior of the leader in the shard. In the design of the incentive mechanism, the differences between the incentive mechanisms of different shards and different nodes were fully considered, a method which is more reasonable and effective than the traditional incentive mechanisms. In addition, we used hypergeometric distribution to calculate the effective probability of each shard and the system under different conditions. We not only analyzed the influence of factors such as the number of shards and the total number of system nodes on the effectiveness of the shards, but also further restricted the proportion of Byzantine nodes that are actually allowed in the shard block connection system, which was reduced from the traditional 1/3 to 0.2.

Author Contributions: Conceptualization, R.C. and L.W.; investigation R.Z.; methodology, R.C., L.W. and C.P.; supervision, L.W.; validation, L.W. and R.Z.; writing—original draft preparation, R.C.; writing—review and editing, L.W. and C.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. Available online: http://www.bitcoin.org/bitcoin.pdf (accessed on 14 June 2022).
2. Bonneau, J.; Miller, A.; Clark, J.; Narayanan, A.; Kroll, J.A.; Felten, E.W. SoK: Research Perspectives and Challenges for Bitcoin and Cryptocurrencies. In Proceedings of the 2015 IEEE Symposium on Security and Privacy, San Jose, CA, USA, 17–21 May 2015; pp. 104–121. [CrossRef]
3. Miličević, K.; Omrčen, L.; Kohler, M.; Lukić, I. Trust Model Concept for IoT Blockchain Applications as Part of the Digital Transformation of Metrology. *Sensors* **2022**, *22*, 4708. [CrossRef]
4. Gai, K.; Guo, J.; Zhu, L.; Yu, S. Blockchain Meets Cloud Computing: A Survey. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 2009–2030. [CrossRef]
5. Xie, J.; Tang, H.; Huang, T.; Yu, F.R.; Xie, R.; Liu, J.; Liu, Y. A Survey of Blockchain Technology Applied to Smart Cities: Research Issues and Challenges. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 2794–2830. [CrossRef]
6. Korpela, K.; Hallikas, J.; Dahlberg, T. Digital supply chain transformation toward blockchain integration. In Proceedings of the 50th Hawaii International Conference on System Sciences, Hilton Waikoloa Village, HI, USA, 4–7 January 2017.
7. Neudecker, T.; Hartenstein, H. Network Layer Aspects of Permissionless Blockchains. *IEEE Commun. Surv. Tutor.* **2018**, *21*, 838–857. [CrossRef]
8. Fusen, W.; Zhihuai, L.; Na, T. Multiple rounds of PBFT verification scheme to improve scale and validity of sharding. *Comput. Eng. Appl.* **2020**, *56*, 102–108.
9. Corbett, J.C.; Dean, J.; Epstein, M.; Fikes, A.; Frost, C.; Furman, J.J.; Ghemawat, S.; Gubarev, A.; Heiser, C.; Hochschild, P.; et al. Spanner: Google's globally distributed database. *ACM Trans. Comput. Syst.* **2013**, *31*, 1–22. [CrossRef]
10. Liu, Y.; Liu, J.; Salles, M.A.V.; Zhang, Z.; Li, T.; Hu, B.; Henglein, F.; Lu, R. Building Blocks of Sharding Blockchain Systems: Concepts, Approaches, and Open Problems. *arXiv* **2021**, arXiv:2102.13364.
11. Hu, B.; Zhang, Z.; Liu, J.; Liu, Y.; Yin, J.; Lu, R.; Lin, X. A comprehensive survey on smart contract construction and execution: Paradigms, tools, and systems. *Gene Expr. Patterns* **2021**, *2*, 100179. [CrossRef] [PubMed]
12. Wang, M.; Wu, Q.; Qin, B.; Wang, Q.; Liu, J.; Guan, Z. Lightweight and Manageable Digital Evidence Preservation System on Bitcoin. *J. Comput. Sci. Technol.* **2018**, *33*, 568–586. [CrossRef]
13. Zahnentferner, J. Chimeric ledgers: Translating and unifying UTXO-based and account-based cryptocurrencies. *Cryptol. ePrint Arch.* 2018; *preprint*.
14. Sel, D.; Zhang, K.; Jacobsen, H.A. Towards solving the data availability problem for sharded ethereum. In Proceedings of the 2nd Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers, Rennes, France, 10–14 December 2018.
15. Luu, L.; Narayanan, V.; Zheng, C.; Baweja, K.; Gilbert, S.; Saxena, P. A secure sharding protocol for open blockchains. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016.
16. Kokoris-Kogias, E.; Jovanovic, P.; Gasser, L.; Gailly, N.; Syta, E.; Ford, B. Omniledger: A secure, scale-out, decentralized ledger via sharding. In Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 20–24 May 2018; pp. 583–598.
17. Al-Bassam, M.; Sonnino, A.; Bano, S.; Hrycyszyn, D.; Danezis, G. Chainspace: A Sharded Smart Contracts Platform. *arXiv* **2018**, arXiv:1708.03778. [CrossRef]
18. Zamani, M.; Movahedi, M.; Raykova, M. Rapidchain: Scaling blockchain via full sharding. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, ON, Canada, 15–19 October 2018.
19. Dang, H.; Dinh, T.T.A.; Loghin, D.; Chang, E.C.; Lin, Q.; Ooi, B.C. Towards scaling blockchain systems via sharding. In Proceedings of the 2019 International Conference on Management of Data, Amsterdam, The Netherlands, 30 June–5 July 2019.
20. Kwak, J.-Y.; Yim, J.; Ko, N.-S.; Kim, S.-M. The design of hierarchical consensus mechanism based on service-zone sharding. *IEEE Trans. Eng. Manag.* **2020**, *67*, 1387–1403. [CrossRef]
21. Abuidris, Y.; Kumar, R.; Yang, T.; Onginjo, J. Secure large-scale E-voting system based on blockchain contract using a hybrid consensus model combined with sharding. *Etri J.* **2021**, *43*, 357–370. [CrossRef]
22. Liu, Y.Z.; Liu, J.W.; Zhang, Z.Y.; Xu, T.G.; Yu, H. Overview on Blockchain consensus mechanisms. *J. Cryptol. Res.* **2019**, *6*, 395–432.
23. Castro, M.; Liskov, B. Practical byzantine fault tolerance and proactive recovery. *OsDI* **1999**, *99*, 173–186. [CrossRef]
24. Lamping, J.; Veach, E. A fast, minimal memory, consistent hash algorithm. *arXiv* **2014**, arXiv:1406.2294.

25. Eisenbud, D.E.; Yi, C.; Contavalli, C.; Smith, C.; Kononov, R.; Mann-Hielscher, E.; Cilingiroglu, A.; Cheyney, B. Maglev: A fast and reliable software network load balancer. In Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16), Santa Clara, CA, USA, 16–18 March 2016.
26. Mendelson, G.; Vargaftik, S.; Barabash, K.; Lorenz, D.H.; Keslassy, I.; Orda, A. AnchorHash: A Scalable Consistent Hash. *IEEE/ACM Trans. Netw.* **2020**, *29*, 517–528. [CrossRef]
27. Pan, J.; Huang, D.C. Blockchain dynamic sharding model based on jump hash and asynchronous consensus group. *Comput. Sci.* **2020**, *47*, 273–280.
28. Pass, R.; Shi, E. Hybrid consensus: Efficient consensus in the permissionless model. *Cryptol. ePrint Arch.* 2016; *preprint*.
29. Bugday, A.; Ozsoy, A.; Sever, H. Securing Blockchain Shards By Using Learning Based Reputation and Verifiable Random Functions. In Proceedings of the 2019 International Symposium on Networks, Computers and Communications, Istanbul, Turkey, 18–20 June 2019; pp. 1–4. [CrossRef]
30. Kogias, E.K.; Jovanovic, P.; Gailly, N.; Khoffi, I.; Gasser, L.; Ford, B. Enhancing bitcoin security and performance with strong consistency via collective signing. In Proceedings of the 25th Usenix Security Symposium (Usenix Security 16), Austin, TX, USA, 10–12 August 2016.