An Efficient and Robust Committee Structure for Sharding Blockchain

Mengqian Zhang, Jichen Li, Zhaohua Chen, Hongyin Chen, and Xiaotie Deng, Fellow, IEEE

Abstract—Nowadays, sharding is deemed a promising way to save traditional blockchain protocols from their low scalability. However, such a technique also brings several potential risks and a huge communication burden. An improper design may give rise to an inconsistent state among different committees. Further, the communication burden arising from cross-shard transactions, unfortunately, reduces the system’s performance. In this article, we first summarize five essential issues that all sharding blockchain designers face. For each issue, we discuss its key challenge and propose our suggested solutions. In order to break the performance bottlenecks, we design a committee structure and propose a reputation mechanism for selecting leaders. The term reputation in our design reflects each node’s honest computation resources. In addition, we present a recovery procedure in case the leader is malicious. Theoretically, we prove that the system is robust under our design. Further simulation results also support this. In addition, the results show that selecting leaders by reputation can dramatically improve the system’s performance.

Index Terms—Sharding blockchain, committee design, reputation mechanism, recovery procedure

1 INTRODUCTION

With the characteristics of decentralization, immutability, and auditability, blockchain technology has been a game changer for various fields - finance [2], Internet of Things [3], supply chain [4], healthcare [5], and education [6], just to name a few. As the name suggests, blockchain is a chain of blocks recording a growing list of transactions. This single chain works as a public ledger, and each contained transaction needs to be verified in advance. For security, all nodes in the system keep a copy of the ledger and reach consensus on it according to the pre-agreed rules. For example, Bitcoin [7], which is the pioneer blockchain system, requires all consensus nodes (or say, miners) to generate new blocks and update the ledger via Proof of Work (PoW). In this way, Bitcoin allows trustless users to carry out electronic transactions directly without any centralized authority.

Unfortunately, traditional blockchain systems (e.g., Bitcoin [7] and Ethereum [8]) suffer from serious scalability problems. In the context of blockchains, a system is considered scalable if its throughput (transactions per second or tps for short) grows with the increase of participation. In these early trials, all nodes should reach a consensus on the system’s state for security. To this end, the block size cannot be too large, and the (expected) block interval cannot be too short. Otherwise, the end-to-end block transmission delay may cause nodes to hold different blocks and have different views of the latest state, thus threatening the system’s security. Under these limits, only a rather fixed amount of transactions can be included in a block over a period of time, regardless of the number of consensus nodes in the network. As a result, the classic design leads to quite a low throughput, which severely restricts the practicability of blockchain. A typical example of this weakness is that the tps of Bitcoin is 3 to 4 orders of magnitude lower than those for centralized payment systems such as Visa: Bitcoin can process only 3.3-7 transactions per second [9] while Visa handles more than 24,000 transactions per second [10]. Nowadays, with the emergence of Non-fungible Tokens (NFT) [11], Metaverse [12] and Web3 [13], increasing requirements for throughput further challenge blockchain.

To overcome this problem, a promising way is to borrow the idea of sharding from the database field. Sharding is a well-known technique for building the scale-out database. The main idea is to break up large tables into smaller chunks and spread them across multiple servers in the cluster. Each server is only responsible for its own part of the data. As a result, sharding enables the sharing of the overall workload and significantly increases processing performance. Inspired by this scheme, researchers in the area of blockchain apply sharding to improve blockchains’ scalability in recent years [14], [15]. Similarly, we can partition nodes into parallel committees and have each committee process a subset of transactions (also known as a shard). There is no overlap of members and tasks between any two committees. In sharding blockchains, each committee maintains a sub-chain instead of the whole chain as before. As a result, much fewer nodes are involved in consensus, which requires less bandwidth for each node. More importantly, the transaction processing rate can be proportional to the number
of committees rather than a constant. Therefore, higher processing capacity is achieved, and the system scales well.

Despite its benefits, applying sharding in blockchain also imposes a great deal of complexity and potential risks. With fewer consensus nodes, each committee is more likely to be attacked, especially in a permissionless environment where anyone is free to join and leave. Furthermore, sharding also leads to the emergence of cross-shard transactions, which means that when the node verifies a transaction, it needs the information stored by another committee for reference. Due to the partition, each committee independently maintains a subset of the overall state. This makes cross-shard transactions difficult to process. An improper design may cause the inconsistency of data, which is an enormous threat to the system. In addition, cross-shard transactions make it inevitable to carry out a large number of communications among different committees. By doing empirical studies, we show that cross-shard transactions take an overwhelming fraction in the sharding blockchain (see Figs. 1 and 2 in Section 3.1). These transactions unavoidably bring a heavy burden on communicating, which is the task of the leader in each committee. As a result, leaders’ capability (e.g., bandwidth, computational resources) becomes a major bottleneck of the whole system.

In this paper, we first identify five essential issues in sharding blockchain, which together compose a complete protocol. For each issue, we discuss its key challenges, briefly review existing approaches, and propose our suggested solution. Then we focus on the communication burdens incurred by the cross-shard transactions. For better performance, we design a reputation mechanism and propose to select leaders via reputation. Specifically, the reputation of a node reflects its honest computation resource. In each round, those nodes with higher reputations will be assigned as leaders. For the safety and liveness of the system, we introduce a unique referee committee and several monitors in each committee. Briefly, the referee committee acts as an arbitrator between opposing parties, while monitors take responsibility for supervising the leader’s behavior. Further, we propose a recovery procedure triggered when a malicious leader is detected. As a consequence, the malicious leader is evicted, and a new leader is re-selected immediately, ensuring the system is running properly.

Theoretically, we prove that selecting leaders via reputation and the committee configuration is secure with overwhelming probability. Also, under such designs, the transaction processing satisfies safety and liveness. Further simulation results support that the reputation mechanism could effectively help to filter out those nodes with abundant computational resources. By arranging these nodes as leaders, each committee could process more transactions and further improve the system’s performance. Also, the recovery procedure ensures the system’s robustness against the adversary’s corruption.

This paper is organized as follows. In Section 2, we discuss five key issues in sharding blockchain. Section 3 states the system model and elaborates on the problem we aim to solve. We propose our solution in Section 4, including the committee configuration, reputation mechanism, and recovery procedure. After that, we give the theoretical analysis of our design in Section 5 and conduct a series of evaluations in Section 6. Finally, we review related studies in Section 7 and conclude the paper in Section 8.

2  ISSUES IN SHARDING BLOCKCHAINS

When it comes to sharding blockchain protocols, there are five basic issues to consider. Like previous works, we will discuss these problems based on the UTXO (Unspent Transaction Output) model. A UTXO indicates the amount of digital currency that its owner can use later. Specifically, a transaction will take one or multiple UTXOs as inputs, and then output new UTXOs. Each UTXO can only be spent once.

2.1 Issue #1: What to Split?

The first issue is to determine what to partition. Before getting into the details, we first distinguish a pair of essential terms. In this context, when referring to the term nodes, we mean the participants who invest resources (hardware, electricity, etc.) and offer services (such as processing transactions) in blockchain systems, similar to the miner in Bitcoin. On the other hand, the participants who use services (e.g., generating transactions) are known as users.

In line with the original intention of the sharding technology, we answer the question as follows:

- **Nodes.** A sharding blockchain divides nodes into disjoint groups, each of which is known as a committee. All committees work concurrently, therefore increasing the transaction processing speed.
- **Pending transactions.** Instead of being kept in the mempool by each node, pending transactions are also partitioned into different groups, which we call shards. It is worth noting that there is a one-to-one mapping from shards to committees. Each shard of transactions is only stored and processed by one committee of nodes.
- **Blockchain history.** To alleviate the full storage burden on nodes, the heavy blockchain data is also split and maintained by separated committees in the form of a sub-blockchain with a lower volume.

2.2 Issue #2: How to Split?

As a subsequence, it is important to ponder the methods to partition the nodes, pending transactions, and blockchain history.

Under the existence of potential Byzantine attackers, nodes are expected to be distributed uniformly into committees. The uniform assignment keeps all committees under a safe Byzantine fraction and drives away the possibility for malicious nodes to gather in or even take over certain committees. To realize a uniformly random assignment, the key is to generate a trusted randomness distributively, which has long been a critical task in blockchain. Several protocols [16], [17] use an external cryptographic hash function, which takes an unpredictable and tamper-resistant value (e.g., the Merkle root of transactions in the previous block) as input, and the output of the function is the randomness. Other tries include Verifiable Random Functions (VRFs) [18] in Algorand [19], and Publicly Verifiable Secret Sharing (PVSS) schemes [20], [21] in Ouroboros [22] and OmnilLedger [23].

For the purposes of load balancing, the transaction is assigned to one committee based on its id, i.e., the hash
value of the transaction. Specific designs vary among different protocols. One method is to use the first $k$ bits of the transaction id to determine its shard, assuming that there are $2^k$ shards (and committees) in the system. For example, all transactions beginning with 010 in their hash belong to shard #2 (here $k = 3$). We can also determine the belonging shard of a transaction by calculating its id modulo $2^k$. To prevent unbalanced loading caused by deliberate users when generating transactions, some protocols add salt when calculating the hash of transactions [24].

We emphasize that, under any proper sharding configuration, each transaction should be precisely maintained by a unique committee to avoid consistency issues\textsuperscript{1}. Accordingly, the blockchain history is also separated, and each virtual UTXO is exactly kept by a single committee.

### 2.3 Issue #3: How to Deal With the Dynamic Membership?

As discussed previously, the sharding technique aims at solving the scalability issue for permissionless blockchains, in which nodes can join and leave freely. In this scenario, the attackers may execute sufficiently many joining and leaving operations to manipulate a certain committee. Thus, it is necessary to adjust the committee assignments periodically. A simple idea is to reshuffle all nodes at the beginning of each round. However, such an idea has several shortcomings. First, a full bootstrapping procedure will stop the whole system. All nodes halt until new committees are formed. Second, frequent reorganization brings expensive communication burdens. Specifically, nodes have to change connections with different sets of committee members. Also, they need to fetch data of the new ledger from other nodes.

To balance usability and performance, replacing only a subset of committee members is a better alternative. Such a solution deals with nodes’ joining and leaving smoothly and resists the slowly-adaptive Byzantine adversary. As a practice, RapidChain [25] borrows the idea of the Cuckoo rule [26], requiring that only a constant number of nodes are switched to other committees in each round. Nevertheless, this method leaves the adversary enough room to take over a certain committee and further cause damage to the whole system. More specifically, at some round $r$, the adversary may plan to corrupt all members of a committee in the next few rounds. Afterward, even if a constant number of members are switched out, the majority of this committee will be malicious.

In order to solve the dynamic membership issue properly, we propose an Expected Constant-Fraction Reshuffling (ECFR) scheme with parameter $\alpha$. In expectation, it requires a constant fraction of members inside a committee to be switched out in each round. Precisely, the $\alpha$-ECFR scheme is designed as follows. Let $\alpha \in (0, 1)$ be a constant. In each round, we independently mark each node with probability $\alpha$, and uniformly reassign all marked nodes to all committees, assuring that all committees have the same size.

We emphasize that the number of reshuffled nodes of $\alpha$-ECFR scheme lies between reshuffling all nodes and RapidChain’s solution. With little bias of notation, reshuffling all nodes is an extreme case of $\alpha$-ECFR scheme with $\alpha = 1$. Further, there exists a constant $\alpha$ enabling that with high probability, honest nodes take the majority in all committees. We prove this security property in Section 5.1.3.

### 2.4 Issue #4: How to Process Intra-Shard Transactions?

Based on the UTXO model, a transaction specifies one or more outputs of previous transactions as its input(s). For a transaction $tx$ charged by some committee $C$, if all its referenced transactions are also maintained in $C$, then $tx$ is known as an intra-shard transaction. As a result, committee members can verify the intra-shard transactions by themselves and reach a consensus afterward. There are a lot of options for the specific consensus algorithm. On the one hand, each committee can be regarded as a smaller blockchain system with fewer nodes and transactions. Thus, the typical blockchain consensus algorithms, such as Proof of Work (PoW) and Proof of Stake (PoS), can be applied here (e.g., [27], [28]). The discussion on these consensus algorithms is out of the scope of this paper. More details can be found in [29].

On the other hand, several protocols which are not suitable for large-scale blockchain systems find their place in this scenario. A typical example is the classic Byzantine Fault Tolerance consensus protocol. As the first practical solution, PBFT [30] correctly survives Byzantine faults in asynchronous networks, allowing a group of nodes to reach an agreement on some value. However, it only works in the setting where all participants are known to each other and the network size is small. Specifically, such a solution does not scale well because it creates $O(n^2)$ communication complexity, where $n$ is the number of nodes. This communication burden becomes unacceptable when the network size increases to, for example, 10,000. Fortunately, these problems are mitigated when applied to each committee in a sharding design. Here are two main reasons. (1) It is easier for nodes to know each other in a smaller committee, and (2) the size of a committee is small enough to enable BFT protocols to work well. As a result, BFT protocols have been widely used in the sharding blockchain systems to reach intra-committee consensus [23], [25], [31], [32], [33], [34].

### 2.5 Issue #5: How to Process Cross-Shard Transactions?

In comparison to an intra-shard transaction, for a transaction $tx$ charged by committee $C$, if $tx$ has a referenced input transaction which is not maintained by $C$, then $tx$ is called a cross-shard transaction. It is crucial to efficiently deal with cross-shard transactions without causing inconsistency among committees.

Existing studies give some solutions to this issue. In OmniLedger [23], before submitting a cross-shard transaction to its assigned committee $C$, the user who generates it needs to collect proof-of-acceptance from all referenced committees, whose members process the transaction and update the UTXOs state. RapidChain [25] transforms the cross-shard transaction into multiple intra-shard transactions with the same trading amount. CycLedger [1] transforms the cross-committee consensus into intra-committee consensus by having the leader of $C$ discuss the relevant inputs.

\textsuperscript{1} Or else, different committees may come up with conflicting results concerning the same transaction.
with its belonged committee, and comes up with a consistent result for all relevant committees. Such a solution resembles a two-phase commitment in a decentralized manner. Pyramid [35] proposes a layered sharding architecture that divides shards into i-shards and b-shards. The i-shards are only responsible for intra-shard transactions, while b-shards are particularly built for processing cross-shard transactions.

In conclusion, as a special outcome of sharding blockchain, the processing of cross-shard transactions requires close cooperation among related committees.

### 3 Problem and Threat Model

#### 3.1 Problem Statement

This paper focuses on the extra communication burdens in sharding blockchain, which are mainly brought from cross-shard transactions. Cross-shard transactions account for a large proportion of sharding blockchain. We conduct several empirical studies to support this idea. According to the data from Bitcoin Visuals\(^2\), which runs a fully-validating bitcoin node and provides extensive statistics over the network, we visualize the information on transaction inputs over the past ten years. Fig. 1 shows the percentages regarding the number of transactions’ inputs in Bitcoin. As can be seen, most transactions have two or three inputs. Furthermore, we randomly divide Bitcoin transactions into multiple shards with the transactions data from Blockchair\(^3\). As Fig. 2 shows, when splitting transactions into 20 shards, approximately 96% of them are cross-shard transactions. When the shard number increases to 100, such fraction ascends correspondingly to above 99%. Although some measures were taken to alleviate the situation [34], [36], cross-shard transactions are inevitable in sharding blockchain.

The processing of a cross-shard transaction involves both intra-committee consensus and cross-committee consensus. As discussed before, most sharding blockchains adopt the classic BFT algorithm to reach consensus within a committee. In this process, leader of the committee plays a key role. Nodes in each committee are led by the leader and work together to process transactions. On the other hand, the cross-committee consensus also heavily relies on the interaction among leaders. Cross-shard transactions make it inevitable to carry out a large number of communications among different committees. This, unfortunately, puts a huge strain on the leader of each committee and makes the leader a bottleneck to the system’s efficiency. In this work, we focus on this key problem in sharding blockchain, aiming to improve the performance of sharding systems.

#### 3.2 Threat Model

We use a Public-Key Infrastructure (PKI) to give each node a public/secret key pair \((PK, SK)\), which enables the digital signature scheme in communication. We assume the existence of a probabilistic polynomial-time Adversary which controls a \(f < 1/3\) fraction of total nodes. Corrupted nodes may collude and act out arbitrary behaviors. The adversary can change the order of messages sent by non-faulty nodes under the restriction given in our network model. Others, known as honest nodes, always follow the protocol and do nothing outside the regulation. The blockchain system works in rounds. Everyone who wants to participate in the round \(t\) is required to solve a PoW puzzle offline before the new round starts. The puzzle is released at the end of round \(t - 1\). It is mainly to limit the number of Sybil identities created by the malicious nodes. At the same time, we suppose the adversary to be mildly-adaptive, such that after appointing a corruption set, the set of nodes become malicious after \(d \geq 1\) rounds. Also, we assume all nodes in the network have access to an external random oracle \(H\), which is collision-resistant.

### 4 Committee Structure and Protocol

In this section, we propose our solution to the communication bottlenecks in sharding blockchain. We present a novel committee structure (Section 4.1), which selects the nodes with more computational resources as the leader of each committee. It is achieved by introducing the concept of reputation based on the nodes’ historical behaviors (Section 4.2). A recovery procedure is also presented for safety and liveness (Section 4.3).

#### 4.1 System Structure

In this section, we show the overall committee structure in our design. Assume there are \(n\) active nodes in the system.
They will be divided into \( m + 1 \) committees, namely, \( m \) common committees and a referee committee \( C_R \).

### 4.1.1 Common Committees

Like the traditional design in sharding blockchain, the \( m \) common committees are responsible for processing transactions in parallel. Differently, each committee contains three types of nodes:

- **Leader.** The leader is required to process transactions and initiate the consensus within a committee. It also needs to handle cross-shard communications with other committees. For efficiency, the leader is decided according to the nodes’ reputation. Simply put, in each round, the top \( m \) available nodes with the highest reputation value are assigned to the leaders of the \( m \) common committees. We will describe the detailed mechanism in the next section.

- **Monitor.** A monitor is required to supervise its leader’s behavior and to respond appropriately when the leader acts maliciously (see Section 4.3). Besides, a monitor should process transactions and participate in the consensus process. Each common committee has \( \lambda \) monitors, which collaboratively are known as the monitor set. These nodes are selected uniformly at random. For safety, it is required that at least one node is honest in the monitor set. This can be achieved by adjusting \( \lambda \) to an appropriate value, like \( 40 \) in practice.

- **Ordinary member.** An ordinary member of the common committee should only process transactions and participate in the consensus process. The ordinary members are randomly assigned and self-organized. Specifically, the selected leaders and monitors will be broadcast and thus get noticed by the whole network. A node that is not in the notification list can figure out which committee it belongs to. In the distributed environment of blockchain, this can be done through cryptographic sorting [19].

### 4.1.2 Referee Committee

In the system, the unique referee committee acts as a mediator to arbitrate between opposing parties. When a certain leader is accused of behaving maliciously, it will handle the exception to keep the system running properly (see Section 4.3). Furthermore, the referee committee shall maintain the reputation value of all nodes in the system (see Section 4.2). The nodes in the referee committee are also selected randomly. It is worth noting that all these nodes are equal in the sense that there is neither leader nor monitors.

Fig. 3 demonstrates the hierarchical structure of the system. In our design, all nodes in a rectangle should be fully connected. More specifically, the nodes within a committee are well connected. Meanwhile, all leaders, monitors, and referee committee members are linked. Compared to other works [23], [25], [31], [37] which expect reliable connections among all honest nodes, our design requires far less amount of reliable connection channels. To avoid misunderstanding, we emphasize that the committee organization is not static. As discussed in Section 2.3, nodes within each committee will be reshuffled regularly. We will analyze the security of the above committee structure in Section 5.

### 4.2 Reputation Mechanism

In this section, we design a mechanism to calculate each node’s reputation. Here, reputation reflects the honest computational resources of a node. It helps to select the leaders of each committee.

Each node \( i \) in the system has a reputation \( r_i \in R \). This value is maintained by the referee committee. Its updating occurs after the committee reaches a consensus on some processed transactions. For clarity, we describe the design of the node’s reputation with the following scenario.

#### 4.2.1 Scenario

In a committee of \( c \) nodes, the leader broadcasts a set of transactions to be processed (\( TXList \) for short) to everyone in the committee. After receiving \( TXList \), each node makes a judgment on the validity of listed transactions. More specifically, the node votes \( \text{Yes} \) for those transactions it agrees, \( \text{No} \) for disagreed, and \( \text{Unknown} \) for the left. Afterward, each node forwards its voting list back to the leader. For the nodes who fail to reply within a certain predetermined time, they are regarded as voting \( \text{Unknown} \) on all transactions. This is to prevent malicious nodes from indefinitely delaying. Denote the voting lists of all nodes by \( VList \). Then the leader processes and integrates these opinions as follows. If a transaction has more than \( \frac{2}{3} \) \( \text{Yes} \), mark it as \( \text{valid} \) (+1) in \( TXList \); otherwise, mark as \( \text{invalid} \) (−1). At last, the leader runs the PBFT [30] within this committee to reach a consensus on both \( TXList \) and \( VList \).

#### 4.2.2 Reputation Design

After consensus on the voting results, the leader scores all members according to their votes and the final decision. Let +1, −1 and 0 represent \( \text{Yes} \), \( \text{No} \) and \( \text{Unknown} \) respectively. Recall that there are \( c \) nodes in the committee, and suppose the amount of transactions to be processed is \( D \) (i.e.,
Let \( VList \) contains \( c \) voting lists, and the size of each list is \( D \). Let the \( D \)-dimension vector \( v_i = \{v_{i,k} | k = 1, 2, \ldots, D\} \) denote the vote of node \( i \), where \( v_{i,k} \) is the node \( i \)'s opinion on the \( k^{th} \) transaction. Similarly, we use \( u = \{u_k | k = 1, 2, \ldots, D\} \) to denote the final results on the transactions’ validity. Then the score of node \( i \) in this consensus is designed as

\[
s_i = D \cdot \cos (v_i, u) + \text{bonus},
\]

\[
= D \cdot \frac{\sum_{k=1}^{D} v_{i,k} u_k}{\sqrt{\sum_{k=1}^{D} v_{i,k}^2} \sqrt{\sum_{k=1}^{D} u_k^2}} + \text{bonus}.
\]

(1)

As (1) shows, the score function contains two parts. In the first part, we use the cosine similarity between a node’s voting vector \( v \) and the resulting vector \( u \) to measure its voting accuracy. The first part ensures that one’s score is positively associated with the number of transactions processed in this voting and its accuracy. The second part is an extra bonus to award those nodes whose votes are totally the same with the final result, i.e., whose accuracy \( \cos (v, u) = 1 \). It is worth noting that the bonus is different for the leader and other members. Specifically, the bonus is designed as follows:

\[
\text{bonus}_i = \begin{cases} 
\sigma \cdot D - \omega / D, & \cos (v_i, u) = 1, \ i \text{ is a leader;} \\
\sigma \cdot D, & \cos (v_i, u) = 1, \ i \text{ is a member;} \\
0, & \cos (v_i, u) < 1.
\end{cases}
\]

(2)

where \( D \) is the number of transactions; \( \sigma, \omega > 0 \) are two predetermined parameters. There are two key points in the design of the bonus function:

- A perfect non-leader member gains more bonus than the leader. It is based on the consideration that the member may have more honest computational resources, yet limited by the leader who sets an upper line on the number of processed transactions of the committee. Therefore, a higher bonus improves the possibility for the member to stand out.

- The bonus difference between a perfect non-leader member and the leader (i.e., \( \omega / D \)) gets smaller with the rise in the number of processed transactions. This is because a larger number of processed transactions implies the better ability of a leader. Therefore, it is reasonable to lessen the possibility that the well-performed leader gets replaced.

After calculating the score of all nodes in the committee, the leader assembles them into a \( \text{ScoreList} = \{s_1, \ldots, s_c\} \). It broadcasts \( \text{ScoreList} \) and \( VList \) to all members and waits for the consensus. In this process, each non-faulty member should sign on the \( \text{ScoreList} \). Afterward, the leader sends the agreement to the referee committee, together with the relevant certification. Consequently, the referee committee updates the reputation of each node \( i \) as \( r_i + s_i \), where \( r_i \) is the node \( i \)'s reputation before this voting.

4.2.3 Reputation’s Application

As shown in (1), if a node has more honest computational resources, it can process more transactions with higher accuracy, thus winning a higher score. Therefore, a node’s reputation, or the accumulated score across rounds, is a good reflection of its honest computational resources.

In our design, first, reputation offers a reference for the leader selection. At the end of any round, \( m \) nodes with the highest reputation are selected as leaders of the next round. These nodes are randomly assigned to \( m \) common committees. As the reputation reflects the honest computational resources one node has contributed, such a method enhances the performance and throughput of the system.

Second, the profit of nodes is also determined by their reputation. Considering that one’s reputation may be negative, we first map the reputation \( r \) to a positive number using a monotone function \( g(\cdot) \), as (3) shows.

\[
g(r) = \begin{cases} 
 e^r, & r \leq 0; \\
 1 + \ln(r + 1), & r > 0.
\end{cases}
\]

(3)

Rewards are then distributed proportionally to the mapped value \( g(r) \). Such a scheme ensures that whoever works more gets more, thus providing enough incentive for nodes to work honestly and as hard as they can.

We also give a punishment mechanism for the misbehavior. For a leader who violates the protocol, its reputation will be decreased to the cube root. Combining the punishment with (3), the mapped value, which is closely related to the node’s revenue, will reduce to approximately one-third of the original mapped value. Therefore, the higher the reputation a leader has, the stronger the punishment it will suffer when behaving maliciously.

4.3 Recovery Procedure

As shown above, the leader of each committee is selected according to reputation. Such a scheme enhances the efficiency of the sharding blockchain by placing resourceful nodes in high-load positions. Nevertheless, a malicious node with many computational resources may be elected as the leader by pretending to be honest in early epochs and accumulating a high reputation. Therefore, we propose the following recovery procedure to ensure that malicious leaders will not affect the safety and liveness of the system.

4.3.1 Semi-Commitment Exchanging

This scheme is to prevent a malicious leader from cheating on the member list and manipulating the consensus results. For example, a malicious leader may ignore the opinions of some honest members and temporarily make some fake nodes during the process of consensus, thus collecting more than \( c/2 \) Yes on an invalid transaction.

The semi-commitment of a committee is the hash of the member list. In the context of cryptography, a commitment scheme has both hiding and binding property. Here, we only require the computational-binding property of a commitment scheme. That is where the name “semi-commitment” comes from. The semi-commitment exchanging scheme runs as follows.

4. The hiding property of the commitment is not necessary here. For a malicious leader, even if it figures out the members of another committee in the semi-commitment exchanging phase, it can do nothing under our threat model as the adversary cannot get control of a trusty node immediately.
Once the committees are formed, the leader should list its members $S = \{PK_1, PK_2, \ldots, PK_c\}$ and compute the committee’s semi-commitment $\textit{SEMI\_COM} = H(S)$. Then it broadcasts both $S$ and $\textit{SEMI\_COM}$ to everyone in the referee committee. To prevent any means of cheating, the leader also delivers both to its monitor set. After receiving the semi-commitments from all committees, the referee committee runs an agreement process inside to check (i) the member in any list has provided a valid PoW solution before the round starts; (ii) all semi-commitments are valid. Afterward, they transmit the set of valid semi-commitments to all leaders and monitors. For the invalid ones, the referee committee will expel the corresponding leaders by invoking the leader re-selection procedure (introduced later). The monitor should verify whether the semi-commitment received from the referee committee matches with the member list $S$ received from its leader. Once detecting any mismatch, an honest monitor reports the circumstance to the referee committee to evict the current leader (to be discussed later).

We show the phase in Algorithm 1. For brevity, the digital signatures and the verifying process executed by monitors are omitted.

**Algorithm 1. Semi-Commitment Exchanging**

Ensure: Each committee gets the semi-commitment of other committees.

{$C_k$} denotes the referee committee. The subscript $k$ is used to denote the $k$-th committee. $C_{k,M}$ denotes the monitor set of the $k$-th committee.

For leader $l_k$:

1: $S_k \leftarrow \{PK_{k,1}, PK_{k,2}, \ldots, PK_{k,c}\}$
2: $H_k \leftarrow H(S_k)$
3: $\textit{COMList} \leftarrow \emptyset$
4: $\textit{ConfList} \leftarrow \emptyset$
5: for $rm \in C_R$ do
6: \hspace{1em} $\textit{SEND} (rm \mid \textit{SEMI\_COM}, H_k, S_k, k)$
7: \hspace{1em} end for
8: for $pm \in C_{k,M}$ do
9: \hspace{1em} $\textit{SEND} (pm \mid \textit{SEMI\_COM}, H_k, S_k)$
10: \hspace{1em} end for
11: while $\textit{DELIVER} (rm \mid \textit{SEMI\_COM}, H_j, j, \textit{Sig})$ do
12: \hspace{1em} $\textit{ConfList}[j][H_j] \leftarrow \textit{ConfList}[j][H_j] + 1$
13: \hspace{1em} if $\textit{ConfList}[j][H_j] \geq |C_R|/2$ then
14: \hspace{2em} $\textit{COMList}[j] \leftarrow H_j$
15: \hspace{1em} end if
16: \hspace{1em} end while
17: for $rm \in C_R$ do
18: \hspace{1em} while $\textit{DELIVER} (l_k \mid \textit{SEMI\_COM}, H_k, S_k, k)$ do
19: \hspace{2em} $\textit{SigList} \leftarrow \textit{CONSENSUS} (\textit{SEMI\_COM}, k, l_k, H_k)$
20: \hspace{2em} for each leader $l$ do
21: \hspace{3em} $\textit{SEND} (l \mid \textit{SEMI\_COM}, H_k, l, \textit{SigList})$
22: \hspace{2em} end for
23: \hspace{1em} end while

4.3.2 Leader Re-Selection

This procedure is to evict a malicious leader and re-select a new leader. Specifically, it is invoked when an honest monitor notices that its leader is malicious or any referee committee member notices that some leader is malicious.

If a monitor wants to accuse its leader, it would first collect valid evidence and send it to the referee committee. Here, we say the evidence is valid if and only if it can support the dishonest behavior of the leader. Taking the aforementioned semi-commitment exchanging as an example, the evidence is the member list $S$ and the semi-commitment $\textit{SEMI\_COM}$ where $\textit{SEMI\_COM} \neq H(S)$. It is worth noting that both have been signed by the leader.

When receiving a valid prosecution from the monitor, the referee committee disqualifies the malicious leader from working in this round. To keep the committee running properly, the monitor who discovered the malicious leader is appointed to be the new leader. On the other hand, if the evidence from the monitor is invalid, any other behavior of the monitor would be disregarded in the current round to prevent possible DDoS attacks. Further, if the referee committee detects a leader’s misbehavior, it will randomly pick one node from the influenced committee to work as the new leader. Afterward, the new leader needs to make a new semi-commitment of the committee via the semi-commitment exchanging scheme. Then the referee committee informs the new semi-commitment and the new leader’s address to all other leaders so that cross-shard communication can continue safely.

5 THEORETICAL ANALYSIS

In this section, we provide a comprehensive discussion on the security of our design, showing that (1) the committee configuration is secure with overwhelming probability, and (2) based on our design, the transaction processing in the system satisfies safety and liveness.

5.1 Security on Committee Configuration

5.1.1 Monitor Set

Recall that monitors of a committee are responsible for supervising the leader’s behavior. We say a monitor set is secure when at least one node in the set is honest. As no more than $1/3$ of nodes are faulty in the system, when the number of monitors is set to 40, the probability that a monitor set is insecure at most:

$$\left(\frac{1}{3}\right)^{40} < 8 \times 10^{-20}.$$

Associated with union bound, when the number of committees is 20, the probability that there exists an insecure monitor set is no more than $2 \times 10^{-18}$.

5.1.2 Single Committee

We say a committee is secure when more than half of nodes are honest. Recall that committees are formed uniformly except for leaders. Let $X$ denote the number of malicious nodes in a committee, and $c$ be the expected committee size. We consider the tail bound of hypergeometric distribution, which gives the following result:

$$\Pr \left[ X \geq \frac{c}{2} \right] = \sum_{x=\lceil c/2 \rceil}^{c} \frac{\binom{n}{x} \binom{n-x}{c-x}}{\binom{2n}{c}} \leq e^{-D_{\frac{c}{2}} f c},$$

(4)
where $D(\cdot | \cdot)$ is the Kullback-Leibler divergence. Here $t < \frac{n}{2}$ and $f < \frac{1}{3} + \frac{1}{2}$, thus,

$$\Pr \left[ X \geq \frac{c}{2} \right] \leq e^{-\frac{n}{2}}.$$  \hfill (5)

When the expected committee size is $c = \Theta(\log^2 n)$, we derive that the probability that a committee is insecure is less than $n^{-\log n}$, which is negligible of $n$.

Fig. 4 visualizes (5). It shows the probability of an insecure committee under varying committee size (i.e., $c$). The population of the whole network is 4,000, and the amount of malicious nodes is 1,333, which is less than one-third of the size of the network. Particularly, when $c = 240$, the error probability for a single committee is less than $2.8 \times 10^{-5}$.

### 5.1.3 All Committees

For the simplicity of expression, we say a sharding blockchain is secure if all committees in any round are secure. First, following the analysis of the previous section and applying union bound, the error probability in the initialization round is no more than $6 \times 10^{-7}$ when the number of committees is less than 20. Second, as we have discussed in Section 2.3, nodes in the committee are reshuffled at the beginning of each round. We suggest randomly picking $\alpha$ percentage of nodes in each committee and uniformly reassigning them to one of the committees, which we call the $\alpha$-ECFR scheme. The following theorem shows that after reshuffling with $\alpha$-ECFR scheme, the system is still secure with high probability.

**Theorem 1.** Assume that the adversary controls $f < 1/3$ fraction of nodes and the corruption requires $d$ ($d \geq 1$) rounds to take effect. Then, there exists a constant $\alpha$, such that under the $\alpha$-ECFR scheme, any round is secure with high probability, given that all previous rounds are secure.

**Proof.** To simplify the proof, we suppose that the adversary specifies a set of nodes to corrupt by the end of round $r$ ($r > 1$) (before it knows the committee configuration of round $r + 1$), and nodes in the set become malicious at the start of round $r + d$. To prove the theorem, we show that no matter how the adversary chooses the set, nodes will be sufficiently reshuffled after $d$ rounds so that all committees have an honest majority, with high probability.

Further, in this proof, we call a node black if it is ever marked in $d$ rounds of reshuffling, and white otherwise. \hfill \Box

**Lemma 1.** For any constant $\beta \in (0, 1)$, there is a constant $\alpha \in (0, 1)$, such that after $d$ rounds of $\alpha$-ECFR, with high probability, for all committees, at most $\beta$ fraction of nodes remain white.

**Proof.** Consider any committee $C$ with size $c = O(\log^2 n)$. Let $X_1, \ldots, X_r, \ldots, X_d$ denote the number of newly-marked nodes in $C$ in round $r + 1$, $\ldots$, $r + i$, $\ldots$, $r + d$. Let $\beta := (1 - \alpha^2)^d$, and $\gamma_i := (1 - (1 - \alpha^2)^i), 1 \leq i \leq d$. (Therefore, $\gamma_d = 1 - \beta$). We define the following $d$ events:

$$E_i := \{ X_1 + \cdots + X_i > \gamma_i \cdot c \}, \quad 1 \leq i \leq d.$$  

First, notice that $E[X_1] = \alpha \cdot c$. By Chernoff-Hoeffding bound, we have

$$\Pr[\neg E_1] = \Pr[X_1 \leq \alpha^2 \cdot c] \leq \exp \left\{ -\frac{1}{2} (1 - \alpha^2) \alpha \cdot c \right\}.$$  

For any $1 \leq i < d$ (when $d \geq 2$), We now give an upper bound on $\Pr[\neg E_{i+1}]$.

$$\Pr[\neg E_{i+1}] = \Pr[\neg E_i \cap \neg E_{i+1}] + \Pr[\neg E_i \cap -E_{i+1}] \leq \Pr[\neg E_i] + \sum_{t \geq \gamma_i \cdot c} \Pr[\neg E_{i+1} | X_i + \cdots + X_t = t] \leq \Pr[\neg E_i] + \max_{t \geq \gamma_i \cdot c} \Pr[X_i + \cdots + X_t \leq \gamma_i \cdot c \cdot t | X_i + \cdots + X_t = t].$$

An important property is that $\gamma_{i+1} - \alpha \leq (1 - \alpha) \gamma_i$, which implies that $\gamma_{i+1} \cdot c \cdot t \leq \alpha \cdot (c - t)$ for any $t > \gamma_i \cdot c$. As $E[X_{i+1}] = \alpha \cdot (c - X_i - \cdots - X_t)$, we can apply Chernoff-Hoeffding bound again:

$$\Pr[\neg E_{i+1}] = \Pr[X_{i+1} \leq \gamma_i \cdot c \cdot t | X_i + \cdots + X_t = t] \leq \max_{t \geq \gamma_i \cdot c} \exp \left\{ -\frac{1}{2} \left( 1 - \gamma_{i+1} \cdot c + (1 - \alpha) \cdot t \right)^2 \right\} \frac{\alpha \cdot (c - t)}{\gamma_i \cdot c} \leq \exp \left\{ -\frac{1}{2} \frac{(1 - \alpha)^2 (1 - 2\alpha) \gamma_i^2 c}{\alpha \cdot (c - \gamma_i)} \right\}.$$  

Therefore, as $c = \Theta(\log^2 n)$, we have

$$\Pr[\neg E_d] = \Pr[\neg E_1] + \sum_{i=1}^{d-1} \Pr[\neg E_{i+1}] = \Theta(\log^2 n).$$
Note that $\mathcal{E}_d$ is the event that at most $\beta$ fraction of nodes in committee $C$ remain white. By a union bound on all $m = \Theta(n/\log^2 n)$ committees, the lemma is proved. \hfill $\Box$

**Lemma 2.** Conditioning on the committee configuration of round $r$ and the number of black nodes, the identity of these black nodes and their belongings at round $r + d$ is uniform.

**Proof.** The lemma holds obviously according to the definition of ECFR scheme. \hfill $\Box$

Now we come back to the main theorem. Let $Y$ be the number of black nodes, and $Z$ be the number of black nodes that are to be corrupted at round $r + d$. By Lemma 1, with high probability, $Y \geq (1 - \beta) \cdot n$.

We simply disregard the negligible failure probability. Due to Lemma 2, $Z$ follows the hypergeometric distribution $\mathcal{H}(f \cdot n, n, Y)$. Therefore, we have

$$\Pr[Z \leq f \cdot (1 + \beta) \cdot Y] \leq \max_{y \geq (1 - \beta) \cdot n} \Pr[Z \leq f \cdot (1 + \beta) \cdot y | Y = y] \leq \exp\left\{-D(f \cdot (1 + \beta) \cdot y) \cdot (1 - \beta) \cdot n\right\},$$

according to the tail bound of hypergeometric distribution.

Now suppose $Z \leq f \cdot (1 + \beta) \cdot Y$, which happens with high probability according to the previous inequality. For a committee $C$, let $Y_C$ be the number of black nodes in $C$ at round $r + d$, and $Z_C$ be the number of black nodes in $C$ that are malicious at round $r + d$. Again, $Z_C \sim \mathcal{H}(Y, Y, Y_C)$, which leads to

$$\Pr[Z_C \geq f \cdot (1 + \beta)^2 \cdot Y_C] \leq \max_{y \geq (1 - \beta) \cdot C} \Pr[Z_C \geq f \cdot (1 + \beta)^2 \cdot y_C | Y_C = y_C] \leq \exp\left\{-D\left(f \cdot (1 + \beta)^2 \cdot y \cdot (1 - \beta) \cdot C\right)\right\},$$

which is negligible in $n$. Subsequently, we assume $Z_C \leq f \cdot (1 + \beta)^2 \cdot Y_C$. Let $M_C$ be the number of malicious nodes in committee $C$ at round $r + d$. Above all, with probability $1 - O(n^{-\log n})$, we have

$$M_C \leq \beta \cdot C + (1 - \beta) \cdot f \cdot (1 + \beta)^2 \cdot C = \left(\beta + f \cdot (1 + \beta)^2 \cdot (1 - \beta)\right) \cdot C.$$

When $f < 1/3$, with appropriate small $\beta$ (e.g., $\beta = 1/8$), we have $\beta + f \cdot (1 + \beta)^2 \cdot (1 - \beta) < 1/2$. Applying an union bound on all $m = \Theta(n/\log^2 n)$ committees, we obtain the theorem.

To sum up, our proposed committee structure is secure.

### 5.2 Safety and Liveness on Transaction Processing

Safety and liveness are two classes of essential properties in sharding blockchain systems, with the following implication respectively:

- **Safety.** Each committee will never propose a block with invalid transactions.
- **Liveness.** By the end of each round, each committee will propose a non-empty valid block.

In this section, we show that our design satisfies both two properties.

#### Claim 1. In cross-shard communication, a malicious leader cannot deceive a trustful leader by forging a member list of its committee as long as the referee committee has an honest majority.

**Proof.** The process of semi-commitment exchanging is under the supervision of its monitor set. Note that with high probability, the monitor set has at least one honest node. Therefore, each leader cannot lie, and the semi-commitment exactly corresponds to the true member list.

Owing to the hash function’s collision-resistance property, a committee’s semi-commitment satisfies the computational binding property. After the semi-commitment is released, only with negligible probability a probabilistic polynomial-time malicious leader can forge a false member list that corresponds to the same semi-commitment. Therefore, the leader cannot provide acceptable results by falsifying the committee’s signature. \hfill $\Box$

#### Claim 2. A malicious leader is always detected and thus evicted via the leader re-selection procedure, as long as the referee committee has an honest majority.

**Proof.** According to the discussion in Section 5.1, with high probability, there is at least one honest node in the monitor set, and the referee committee has an honest majority. Therefore, as a leader’s action is always monitored by the monitor set during the execution of the system, any irregular behavior from the leader will be detected and the valid evidence can be provided by an honest monitor. At the same time, as the leader itself signs the evidence, a malicious leader can never deny the charges. \hfill $\Box$

#### Claim 3. A trustful leader will never be framed up by a faulty monitor, as long as the referee committee has an honest majority.

**Proof.** We mention that an evidence is valid if and only if it can derive a leader’s malicious behavior. The corresponding information should be signed by the leader. A malicious monitor, however, cannot counterfeit an evidence with the leader’s signature. Therefore, a trustful leader will never be unjustly accused. \hfill $\Box$

As discussed above, any leader can never behave badly, such as tampering with cross-shard transactions, proposing blocks with invalid transactions, and so on. Otherwise, it will be evicted until an honest node becomes the leader. As a consequence, a non-empty valid block will be proposed by the end of each round. Thus, we claim that our design has both safety property and liveness property.

In conclusion, considering both the committee configuration and the transaction processing, our design is robust.

### 6 System Evaluation

In this section, we conduct a series of simulations to evaluate the performance and robustness of our design. First, we describe the simulation settings. Then we show the simulation results and give the corresponding analysis.

#### 6.1 Setup

This section describe the general simulation setup.
6.1.1 Implementation

In our simulations, 20,000 nodes are divided into \( m \) committees. By default, \( m \) is set to 50. To better imitate the realistic condition, we crawl 208,936 transactions from the Bitcoin network and ignore those with more than 12 inputs (which barely happen in today’s network, see Fig. 1). These transactions are sent to the system sequentially. For each transaction \( tx \), it is randomly assigned to a committee, the ID of which is \( tx_{\text{hash}} \mod m \). Due to the high throughput of sharding blockchain, these transactions are utilized repeatedly. For convenience, we discretize the time into rounds with a fixed length. Each simulation runs for 1,000 rounds.

We assume that the honest computational resources of nodes follow a beta distribution. The cost of processing a transaction is assumed to be linear with the number of its inputs, which is realistic in sharding blockchain. Besides processing transactions, leaders suffer from extra resource consumption on decision-making, communication, and reputation updating, while other nodes also carry the burden of communication in reaching consensus. Thus, we assume that for leaders and other members, the fraction of computation resources specially used to process transactions is 70% and 90%, respectively. Further, for the two parameters \( \sigma \) and \( \omega \) in the bonus function (see (2)), we set \( \sigma = 0.01 \) and \( \omega = 0.1 \).

6.1.2 Evaluation Metrics

We evaluate the effect of our design using the following metrics.

- **Leaders’ Capacity**: The leaders’ capacity is defined as the average computation resources of all leaders in one round. This metric is used to reflect whether the reputation mechanism helps to select the nodes with more computation resources as leaders. As illustrated in Section 3.1, our key problem is to improve the performance of processing cross-shard transactions. Since leaders suffer the most communication burden, we hope leaders have higher computation resources.

- **Throughput**: The throughput is the number of processed transactions in one round. This metric is used to measure the system’s processing ability.

6.2 Efficiency

In the simulation, we compare our design with the state-of-the-art RapidChain protocol [25] in the non-adversarial model to evaluate the effects of the reputation mechanism. Specifically,

- **Our Solution**: Select nodes with the highest reputation as leaders.
- **RapidChain**: The default random placement strategy in RapidChain.

Fig. 5a shows the average computation resources of all leaders (i.e., leaders’ capacity) in each round. As can be seen, for RapidChain, the leaders’ capacity almost stabilizes at a small value of approximately 1. In contrast, under our leader selection scheme via reputation, the leaders’ capacity rises rapidly and goes beyond 1.6 after 100 rounds. Therefore, our solution is helpful in picking out resource-rich nodes.

Fig. 5b presents for each round the average throughput of the system across all previous rounds (including the current round). Specifically, it turns out the average throughput of our protocol is approximately 1.4x higher than RapidChain after 100 rounds. The similarity between Figs. 5a and 5b reflects the positive correlation between the average throughput and leaders’ capacity. For a more comprehensive comparison, we further evaluate the average throughput per round under different numbers of shards. As Fig. 5c shows, with the increase of shards’ number, the average throughput of both solutions improve. This result demonstrates the advantage of the sharding technique. It is worth noting that in this process, our solution always has a significant advantage over RapidChain and the growth rate of our solution is greater as well. These results provide compelling evidence that selecting leaders via reputation significantly helps the system process more transactions and improve the throughput.

6.3 Robustness

This set of simulations evaluates the robustness of our solution against corruption. We consider the adversarial model that attackers regularly corrupt leaders of shards. Specifically, every 15 rounds, attackers would start to corrupt all leaders of shards, and the corruption takes effect after 10 rounds. In this kind of attack, we compare the following two strategies.

- **With Reselection**: Once a leader is corrupted, reselect a new leader to enable the committee to work well in this round.
- **Without Reselection**: When a leader is corrupt, its committee process no transactions in this round.
Fig. 6a compares the two strategies regarding the leader’s capacity over time. As we can see, influenced by the adversary’s corruption, the average computational resources of leaders suffer a decrease in both two cases. The case without the leader reselection procedure, however, decreases much more quickly. A similar situation also arises with the average throughput evaluation, as Fig. 6b shows. It indicates that the system could effectively counter attacks with the leader reselection procedure. We also consider varying shard numbers in Fig. 6c. We can find that with the leader reselection procedure, the system always have higher throughput under different numbers of shards. These results support that the leader reselection procedure would significantly enhance the robustness of a sharding system.

Overall, the simulation indicates that our design is both efficient and robust. Specifically, the reputation mechanism helps to pick out those nodes with more computation resources. Assigning them as leaders therefore dramatically improves the system’s throughput. Additionally, our leader reselection procedure successfully enhances the robustness against adversary’s corruption.

7 RELATED WORK

7.1 Sharding Blockchain Protocols

Elastico [31] is the first sharding-based protocol for public blockchains, which can tolerate up to a fraction of 1/4 of malicious parties. Unfortunately, it has a weak safety guarantee as the randomness in each epoch of the protocol can be biased by the adversary. OmniLedger [23] also allows the adversary to take control of at most 25% of the validators, as well as assuming the adversary to be mildly-adaptive. Nevertheless, its underlying consensus protocol ByzCoin has several security and performance issues [38], [39]. RapidChain [25] and CycLedger [1] further enhance the efficiency of sharding-based blockchain protocols on a large scale. To maintain liveness concerning malicious committee leaders, [33] involves a reference committee with an honest majority. It uses two-phase commit (2PC) and two-phase locking (2PL) protocols to coordinate all cross-shard transactions. SSChain [27] introduces a root chain to process cross-shard transactions. These two designs cause a large reduction in the parallelism brought by sharding, as the reference committee/root chain needs to process cross-shard transactions sequentially.

Compared to intra-shard transactions, the cross-shard transaction requires a longer confirmation time, extra communication, and computation cost. In terms of this issue, Optchain [36] provides a transaction assignment algorithm to reduce the number of cross-shard transactions, while keeping load balance among shards. It mainly uses a PageRank-like algorithm to assign linked transactions to the same shard. This algorithm is client-driven, requiring users to query information and do some computations. BrokerChain [34] focuses on the account-based sharding blockchain and achieves this by the state-graph partition and account segmentation mechanism. Specifically, a partition shard is devised to adaptively divide account states into multiple shards, thus reducing the number of cross-shard transactions and ensuring workload balance.

7.2 Reputation in Blockchain

The reputation mechanism is an important method in blockchains to provide incentives and enhance robustness.

Some works explicitly embed the reputation scheme into the consensus process. [40] introduces a reputation scheme into the consensus mechanism of consortium energy blockchains. In their work, reputation is a comprehensive measure of a node’s hardware contribution to the system. [41] proposes a reputation-based Byzantine Fault-Tolerance algorithm for consortium blockchain. In this design, reputation is used to score the behavior of consensus nodes. Nodes with higher reputations have a higher weight in the voting process. The paper [42] proposes a protocol for Industrial Internet of Things in which a node’s reputation will affect its mining difficulty. [43] designs a reputation-based consensus protocol, where the reputation of a node is based on its asset, transaction activity, and consensus participation. [44] also proposes a consensus protocol, in which a node’s reputation is used for jury selection and the block is created by the voting of juries.

Some works utilize the reputation system to enhance the performance of blockchain. [45] studies how to reduce verification workloads through a reputation mechanism in permissioned blockchains. [46] proposes a distributed reputation system for energy trading. The given system could provide a higher blockchain performance and bring a fairer market by reducing price discrimination and being friendlier to buyers/sellers with more contribution to the market. [47] proposes a reputation scheme for a blockchain-based cooperative multi-domain DDoS mitigation system. In this design, reputation reflects the contribution of each node and is managed through smart contracts. Rewards are distributed according to reputation to incentivize nodes to behave honestly. [48] proposes a reputation module for blockchain-based supply chain
applications to ensure the on-chain data which is provided by supply chain entities are trusted from the source.

The concept of reputation is also used to deal with the attacks in the blockchain. For example, to protect mining pools against possible block withholding attacks and distributed denial of service attacks, [49] propose to use the probability interval to define the level of miners’ reputation according to the manager’s evaluations. Only miners who meet the requirements of the reputation threshold have the right to join the pool and participate in mining. Likewise, [50] proposes a blockchain called PoolCoin, which allows pool managers to accept trusted miners in their mining pools. Moreover, miners can also evaluate pool managers and decide to join a pool based on their ranking. [51] proposes a blockchain system named ReputoCoin which can resist flash attacks (a.k.a. bribery attacks), where an attacker can possess more than 50 percent computing power of the entire network temporarily by renting mining capacity. It also uses a reputation-based voting mechanism for consensus. [52] proposes a reputation mechanism to prevent the spread of spam transactions in the blockchain. Regarding the unreliable and inactive behavior in blockchain networks, [53] proposes a reputation-based relay protocol to accelerate the broadcast of transactions.

8 Conclusion
In this work, we identify five basic issues in sharding blockchain. Specifically, we analyze the challenges involved and present our suggested solutions. In order to overcome the performance bottlenecks caused by cross-shard transactions, we introduce the concept of reputation and propose a reputation mechanism. By scoring each node according to its historical behavior, the term of reputation helps to locate those nodes with more honest computational resources. By assigning them to high-workload positions, the reputation mechanism enhances the system’s capability to process transactions. In addition, we introduce a semi-commitment scheme and a recovery procedure. They together enable users to detect and evict malicious leaders, thus trading safely in the system. Theoretical analysis and simulation results confirm that our design can significantly improve the system performance and robustness, without sacrificing any safety and liveness.

Acknowledgments
M. Zhang, J. Li, Z. Chen, and H. Chen contributed equally to this work. A preliminary version of this article appears as [1].

References
[1] M. Zhang, J. Li, Z. Chen, H. Chen, and X. Deng, “CycLedger: A scalable and secure parallel protocol for distributed ledger via sharding,” in Proc. IEEE Int. Parallel Distrib. Process. Symp., 2020, pp. 358–367.
[2] P. Treleaven, R. G. Brown, and D. Yang, “Blockchain technology in finance,” Computer, vol. 50, no. 9, pp. 14–17, 2017.
[3] X. Wang et al., “Survey on blockchain for Internet of Things,” Comput. Commun., vol. 136, pp. 10–29, 2019.
[4] S. Saberi, M. Kouhizadeh, J. Sarkis, and L. Shen, “Blockchain technology and its relationships to sustainable supply chain management,” Int. J. Prod. Res., vol. 57, no. 7, pp. 2117–2135, 2019.
[5] T. McChin, K.-K. R. Choo, C. Z. Liu, and D. He, “Blockchain in healthcare applications: Research challenges and opportunities,” J. Netw. Comput. Appl., vol. 135, pp. 62–75, 2019.
[6] G. Chen, B. Xu, M. Lu, and N.-S. Chen, “Exploring blockchain technology and its potential applications for education,” Smart Learn. Environ., vol. 5, no. 1, pp. 1–10, 2018.
[7] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” Decentralized Bus. Rev., p. 21260, 2008.
[8] G. Wood et al., “Ethereum: A secure decentralised generalised transaction ledger,” Ethereum Project Yellow Paper, vol. 151, no. 0, pp. 1–32, 2014.
[9] K. Croman et al., “On scaling decentralized blockchains - (A position paper),” in Proc. Int. Conf. Financial Cryptogr. Data Secur., 2016, pp. 106–125.
[10] Visa, “Visa’s transaction per second.” Accessed: Nov. 6, 2022. [Online]. Available: https://usa.visa.com/run-your-business/small-business-tools/retail.html
[11] H. Chen, Y. Cheng, X. Deng, W. Huang, and L. Rong, “ABSNT: Securitization and repurchase scheme for non-fungible tokens based on game theoretical analysis,” 2022, arXiv:2202.02199.
[12] J. L. Lee et al., “Trust-based consensus: A complete survey on technological singularity, virtual ecosystem, and research agenda,” 2021, arXiv:2110.05352.
[13] J. Potts and E. Rennie, “Web3 and the creative industries: How blockchain are reshaping business models,” in A Research Agenda for Creative Industries. Cheltenham, U.K.: Edward Elgar Publishing, 2019, pp. 93–111.
[14] J. F. R. K. Yu, T. Huang, R. Xie, J. Liu, and Y. Liu, “A survey on the scalability of blockchain systems,” IEEE Netw., vol. 33, no. 5, pp. 166–173, Sep./Oct. 2019.
[15] G. Wang, Z. J. Shi, M. Nixon, and S. Han, “SoK: Sharding on blockchain,” in Proc. 1st ACM Conf. Adv. Financial Technol., 2019, pp. 41–61.
[16] I. Bentov, R. Pass, and E. Shi, “Snow White: Provably secure proofs of stake,” IACR Cryptol. ePrint Arch., vol. 2016, 2016, Art. no. 919.
[17] B. David, P. Gazi, A. Kiayias, and A. Russell, “Ouroboros Praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain,” in Proc. 37th Annu. Int. Conf. Theory Appl. Cryptographic Techn. Adv. Cryptol., 2018, pp. 66–98.
[18] S. Micali, M. O. Rabin, and S. P. Vadhan, “Verifiable random functions,” in Proc. IEEE 40th Annu. Symp. Found. Comput. Sci., 1999, pp. 120–131.
[19] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, “Algorand: Scaling byzantine agreements for cryptocurrencies,” in Proc. 26th Symp. Oper. Syst. Princ., 2017, pp. 51–68.
[20] B. Schoenmakers, “A simple publicly verifiable secret sharing scheme and its application to electronic,” in Proc. 9th Annu. Int. Conf. Confl. Res., 1999, pp. 148–164.
[21] I. Casidu and B. David, “SCRAPe: Scalable randomness attested by public entities,” in 15th Int. Conf. Appl. Cryptogr. Netw. Secur., 2017, pp. 537–556.
[22] A. Kiayias, A. Russell, B. David, and R. Oliynykov, “Ouroboros: A provably secure proof-of-stake blockchain protocol,” in Proc. 37th Annu. Int. Conf. Cryptol., 2017, pp. 357–388.
[23] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailey, E. Syta, and B. Ford, “OmniLedger: A secure, scalable, decentralized ledger via sharding,” in Proc. IEEE Symp. Secur. Privacy, 2018, pp. 583–598.
[24] A. Manuskin, M. Mirkin, and I. Eyal, “Ostraka: Secure blockchain scaling by node sharding,” in Proc. IEEE Eur. Symp. Secur. Privacy Workshop Workshops, 2020, pp. 397–406.
[25] M. Žamani, M. Movahedi, and M. Raykova, “RapidChain: Scaling blockchain via full sharding,” in Proc. ACM SIGSAC Conf. Comput. Commun. Secur., 2018, pp. 931–948.
[26] B. Awerbuch and C. Scheideler, “Towards a scalable and robust DHT,” Theory Comput. Syst., vol. 45, no. 2, pp. 234–260, 2009.
[27] H. Chen and Y. Wang, “S3Chain: A full sharding protocol for public blockchain without data migration overhead,” Pervasive Mobile Comput., vol. 59, 2019, Art. no. 101055.
[28] D. R. Lee, Y. Jang, and H. Kim, “Poster: A proof-of-stake (PoS) blockchain protocol using fair and dynamic sharding management,” in Proc. ACM SIGSAC Conf. Comput. Commun. Secur., 2019, pp. 2553–2555.
[29] M. S. Ferdous, M. J. M. Chowdhury, M. A. Hoque, and A. Colman, “Blockchain consensus algorithms: A survey,” 2020, arXiv:2001.07091.
M. Zhang et al., “Accelerating transactions relay in blockchain networks via reputation,” in Proc. IEEE/ACM 29th Int. Symp. Qual. Serv., 2021, pp. 1–10.

Mengqian Zhang received the BE degree in computer science from the Ocean University of China, in 2018. She is currently working toward the doctor’s degree with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. Her current research interests include Blockchain and algorithmic game theory.

Jichen Li received the BS degree in computer science from the School of EEECS, Peking University. He is currently working toward the PhD degree with Peking University. His current research interests focus on Blockchain and mechanism design.

Zhaohua Chen received the BS degree in computer science from the School of EEECS, Peking University. He is currently working toward the PhD degree with Peking University. His current research interests include Blockchain and mechanism design.

Hongyin Chen received the BS degree from the School of EEECS, Peking University, in 2020. He is currently working toward the doctor’s degree with the Center on Frontiers of Computing Studies, Peking University. His current research interests include Blockchain and mechanism design.

Xiaotie Deng (Fellow, IEEE) received the BSc degree from Tsinghua University, the MSC degree from the Chinese Academy of Sciences, and the PhD degree from Stanford University, in 1989. He is currently a chair professor with Peking University. He is currently working toward the doctor’s degree with Peking University. His current research interests include Blockchain and mechanism design.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.