Pyramid: A Layered Sharding Blockchain System

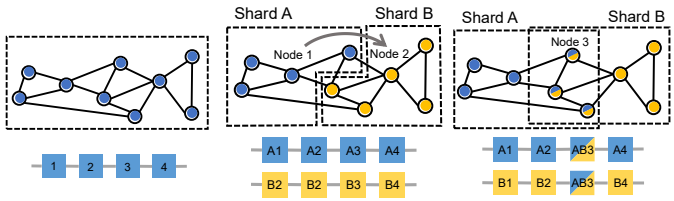
Zicong Hong[1,2], Song Guo[2], Peng Li[3], and Wuhui Chen*[1]
[1]School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China
[2]Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China
[3]School of Computer Science and Engineering, The University of Aizu, Aizuwakamatsu, Japan
zicong.hong@connect.polyu.hk, song.guo@polyu.edu.hk
pengli@u-aizu.ac.jp, chenwuh@mail.sysu.edu.cn
*Corresponding author: Wuhui Chen

***Abstract*—Sharding can significantly improve the blockchain scalability, by dividing nodes into small groups called shards that can handle transactions in parallel. However, all existing sharding systems adopt complete sharding, i.e., shards are isolated. It raises additional overhead to guarantee the atomicity and consistency of cross-shard transactions and seriously degrades the sharding performance. In this paper, we present Pyramid, the first layered sharding blockchain system, in which some shards can store the full records of multiple shards thus the cross-shard transactions can be processed and validated in these shards internally. When committing cross-shard transactions, to achieve consistency among the related shards, a layered sharding consensus based on the collaboration among several shards is presented. Compared with complete sharding in which each cross-shard transaction is split into multiple sub-transactions and cost multiple consensus rounds to commit, the layered sharding consensus can commit cross-shard transactions in one round. Furthermore, the security, scalability, and performance of layered sharding with different sharding structures are theoretically analyzed. Finally, we implement a prototype for Pyramid and its evaluation results illustrate that compared with the state-of-the-art complete sharding systems, Pyramid can improve the transaction throughput by 2.95 times in a system with 17 shards and 3500 nodes.**

I. Introduction

Blockchain draws tremendous attention from academia and industry [1], [2], since it can provide distributed ledgers with data transparency, integrity, and immutability to untrusted parties for pseudonymous online payment, cheap remittance, and various decentralized applications, such as auction houses [3] and marketplaces [4]. However, existing blockchain systems have poor scalability because their consensus protocols involve all nodes [5], [6]. Every node needs to verify and store all transactions and every consensus message needs to be broadcast in the whole blockchain network.

Sharding is one of the most promising technologies that can significantly improve the scalability of blockchain [7]–[12]. Its main idea is to divide nodes into small groups called *shards*, which can handle transactions in parallel. A comparison between non-sharding blockchain and sharding one is shown in Fig. 1(a) and Fig. 1(b). In the non-sharding system, all nodes maintain only one blockchain. In the sharding system, each shard can maintain a blockchain and run its own consensus protocol independently. Elastico [7] is the first sharding system that achieves sharding for transaction computation, but it still needs to broadcast each block to all shards and requires all nodes to store the entire ledger. Afterwards, Omniledger [8] and RapidChain [9] have been developed to further achieve sharding for computation, storage, and communication. Besides, recent research efforts include the sharding for smart contracts [11], [13], security improvement for sharding [10], and sharding for permissioned blockchain systems [12].



(a) Non-sharding (b) Complete sharding (c) Layered sharding

Fig. 1. Illustration for different blockchain systems

The sharding scheme adopted by existing work is considered as *complete sharding* as shown in Fig. 1(b), where shards are isolated completely and each node belongs to a single shard. The nodes in a shard are responsible for the consensus of this shard, including verification, storage, and communication. Although complete sharding can improve blockchain scalability, it raises additional challenges for cross-shard transactions, which account for a large portion of total transactions in the blockchain. According to statistics, more than 96% transactions are cross-shard in a sharding-based system [9]. Fig. 1(b) illustrates a cross-shard transaction sent by Node 1 in Shard A and received by Node 2 in Shard B. Existing sharding systems, e.g., RapidChain [9] and Monoxide [10], divide each cross-shard transaction into several sub-transactions that let associated shards handle them individually to guarantee the atomicity and consistency. Therefore, the system needs to process several sub-transactions per cross-shard transaction, which seriously degrades the sharding performance in terms of throughput and confirmation latency.

In this paper, we propose Pyramid, a novel *layered sharding* system, whose basic idea is to allow shards to overlap, rather than isolating them completely, so that some nodes can locate in more than one shard as illustrated in Fig. 1(c). For cross-shard transactions involving the different numbers of shards, nodes located in the overlap of these shards can verify and process them directly and efficiently, rather than splitting them

into a number of sub-transactions like the complete sharding. For example, in Fig. 1(c), Node 3 can verify the cross-shard transaction given in Fig. 1(b) and propose a block including it directly because Node 3 stores the records of both shards.

Although Pyramid is promising to increase system throughput by offering an efficient way to handle cross-shard transactions, its design faces several critical challenges. The first challenge is about consensus protocol design, which specifies how to generate a block and how to verify a block. Existing sharding systems adopt either Proof-of-Work (PoW) or Byzantine-based schemes for block proposal in each shard. Due to the complete sharing, the shards can verify their blocks independently. However, some shards in Pyramid are responsible for generating blocks of cross-shard transactions. The generated blocks need to be sent to corresponding shards for commitment, which makes the consensus protocol design more challenging. The second challenge is about shard construction, i.e., how many shards are needed and which nodes should be assigned to which shards. Different from traditional sharding schemes that all shards have the identical role, shards in Pyramid play different roles. Some shards are responsible for internal transactions, while others should handle cross-shard transactions. It is critical to study shard assignment for Pyramid, which determines system throughput and security level. The main contributions of this paper are summarized as follows:

- **Layered Sharding Architecture:** We present the formation process of layered sharding. Based on the characteristics of cross-shard transactions, we investigate the verification rules for cross-shard transactions and design a cross-shard structure for blocks. With our cross-shard block design, cross-shard transactions can be included into one single block for the consensus.
- **Layered Sharding Consensus:** We propose a layered sharding consensus protocol to commit cross-shard blocks in each shard depending on the collaboration among nodes in different shards. Compared with complete sharding in which cross-shard transactions are split into multiple sub-transactions and processed in multiple consensus rounds, our layered sharding consensus can commit each cross-shard transaction in one round thus improve the sharding performance.
- **Theoretical Analysis:** We give a theoretical analysis for Pyramid with different sharding structure in the aspect of security, scalability and performance and compare it with the non-sharding and complete sharding system considering the distribution of multi-step transactions.
- **System Implementation:** We develop a prototype for Pyramid and evaluate its performance by comparing it with two state-of-the-art complete sharding systems. The result illustrates that compared with complete sharding, Pyramid improves the transaction throughput up to 2.95 times in a system with 17 shards and 3500 nodes.

Paper Organization. The remainder of this paper is structured as follows. Section II provides related work for the paper. Section III describes the system model and threat model. Afterwards, Section IV presents the layered sharding formation, cross-shard block design and layered sharding consensus. Section V gives the theoretical analysis for layered sharding. Section VI reports the evaluation results for our prototype. Section VII discusses shard overlapping in complete sharding, heterogeneous blockchain nodes and multi-step transactions. Finally, Section VIII concludes the paper.

II. Related Work

The researches around blockchain sharding experience three periods as follows.

A. Non-sharding System

Bitcoin [1] and Ethereum [2] are the two most famous non-sharding systems. In a non-sharding system, the untrusted nodes maintain a distributed ledger to guarantee a trustworthy and decentralized environment. However, the consensus of the non-sharding system involves all nodes in the system, which means each node needs to verify and store all transactions happened in the system and every consensus message needs to be broadcast in the whole network. It results in the poor scalability of the non-sharding system.

B. Partial Sharding System

ELASTICO [7] is the first public and decentralized sharding-based blockchain system in which every shard is responsible to validate a set of transactions and achieves consensus based on PBFT. Then, a final shard verifies all the transactions received from shards into a global block which will be then broadcast to and stored in all nodes in the system. Although ELASTICO achieves sharding for transaction computation, it does not achieve sharding for storage and communication in the system, thus ELASTICO is referred to as a partial sharding system. Since each node receives and stores complete information of the system, there do not exist cross-shard transactions in the system, but the nodes still suffer from heavy storage and bandwidth overhead.

C. Complete Sharding System

To further alleviate the overhead of nodes in the blockchain, a number of researches focus on complete sharding, i.e., sharding for transaction computation, storage and communication.

Omniledger [8] is the first sharding-based blockchain system achieving full sharding. The system adopts a client-driven mechanism to commit cross-shard transactions. Then, another client-driven sharding system named Chainspace [11] is presented to support sharding for generic smart contracts.

However, the client-driven mechanisms put extra burden on typically lightweight user nodes and are vulnerable to denial-of-service (DoS) attacks by malicious users. To further improve the performance, researchers propose several shard-driven mechanisms to process cross-shard transactions. For example, RapidChain [9] proposes a *transfer mechanism* in the unspent transaction output (UTXO)-based system. For each cross-shard transaction, the mechanism first transfers

all involved UTXOs to the same shard by sub-transactions. Then, the cross-shard transaction becomes an internal transaction and can be processed in single shard. Monoxide [10] proposes a *relay mechanism* in the account/balance-based system. Each cross-shard transaction is split into a number of sub-transactions including internal transactions and relay transactions. Each of internal transactions is corresponding to a related shard. A relay transaction is needed between every two consecutive internal transactions.

Although the above complete sharding systems can guarantee the atomicity and consistency of cross-shard transactions, their effort to commit cross-shard transactions is multiplied. In particular, each cross-shard transaction needs to be split into a number of sub-transactions, and the complete sharding system needs to validate and process all related sub-transactions to commit a cross-shard transaction. This seriously degrades the sharding performance in terms of throughput and confirmation latency.

III. System and Threat Model

A. System Model

In Pyramid, there are N nodes and S shards in the system. The shards can be divided into two kinds. One includes nodes responsible for handling only internal transactions. They are referred to as *i-shards*. Similar with the shards in complete sharding schemes, each i-shard independently verifies internal transactions of the shard. The other one includes nodes bridging multiple i-shards by dealing with cross-shard transactions related to the i-shard. They are referred to as *b-shards*. Note that b-shards can handle internal transactions like i-shards, but they also take the job of processing cross-shard transactions. For example in Fig. 2, there are three shards in the system, i.e., i-shard A, i-shard B, and b-shard C. The b-shard C takes the job for cross-shard transactions among i-shard A and B.

Pyramid adopts the account/balance model to represent the ledger state in which each node has a pair of account and balance. The nodes in Pyramid are connected by a partially synchronous peer-to-peer network [14], and messages sent by a node can reach any other nodes with optimistic, exponentially-increasing time-outs.

B. Threat Model

There are two kinds of nodes in Pyramid: *honest* and *malicious*. The honest nodes abide by all protocols in Pyramid while malicious nodes may collude with each other and violate the protocols in arbitrary manners, such as denial of service, or tampering, forgery and interception of messages. The fraction of malicious nodes is denoted as f in the system. In other words, fN nodes are controlled by a Byzantine adversary. Furthermore, similar to other sharding systems [7]–[9], we assume that the Byzantine adversary is slowly-adaptive, i.e., the set of malicious nodes and honest nodes are fixed during each epoch and can be changed only between epochs.

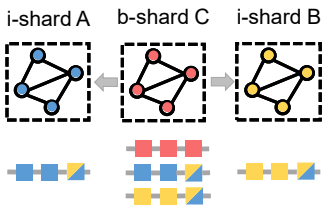


Fig. 2. Illustration for a layered sharding system for i-shard A, B and b-shard C

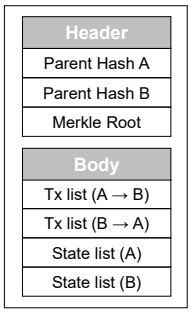


Fig. 3. Structure of a cross-shard block for i-shard A and B

IV. System Design

A. Layered Sharding Formation

1) Randomness Generation. The running of Pyramid proceeds in fixed time periods named *epochs*. At the beginning of each epoch, a randomness is generated via a public-verifiable, bias-resistant, unpredictable and available randomness generation method [15], e.g., the verifiable random function [16], verifable delay function [17], and trusted execution environment [12], similar to that of other sharding systems [8], [9]. It can be considered as a separated module in a sharding system and is orthogonal with our work, thus we do not discuss in detail.

2) Participation. For each node, before joining an epoch, a fresh Proof-of-Work (PoW) puzzle is generated based on its public key and the randomness of the epoch. To participate in an epoch, a node needs to solve its exclusive puzzle generated in the epoch. After solving the puzzle successfully, a node needs to append its solution into an identity blockchain to register its identity. The identity blockchain is a PoW-based blockchain which is specialized to record identities of nodes, the same as the identity blockchain in [6], [8], [9]. The difficulty of PoW puzzle in each epoch can be adjusted according to the number of identities in the previous epoch to keep the number of nodes stable. If there are more than N identities in the previous epoch, the difficulty will be increased in this epoch, otherwise it will be reduced.

3) Assignment. Each admitted node is assigned a shard ID randomly based on the identity of the node and the randomness generated in the epoch. The shard IDs are divided into two types, i.e., i-shard ID and b-shard ID, and each b-shard ID is corresponding to a number of i-shard IDs. The setting for shard IDs will be discussed in Section V that shows the setting bears on the security, scalability and performance of layered sharding. The nodes with i-shard IDs belong to the i-shards while the nodes with b-shard IDs belong to the b-shards. Note that the results of assignment for all nodes in the epoch are public and they can be computed based on the randomness in the epoch and the identity chain.

B. Cross-Shard Block Design

Transaction Definition. A transaction is a payment between two accounts, namely *sender* and *receiver*. If the sender and

receiver of a transaction are located in different shards, we call it a *cross-shard* transaction. A more general case about transactions involving more than two accounts and supporting smart contract is discussed in Section IV-D.

Validation Rule. The validity of each transaction can be divided into *source validity* and *result validity*. The source validity denotes that the state of the sender satisfies the condition for the transaction, i.e., the sender has sufficient money, and the result validity denotes that the state of the receiver accords with the running result of the transaction, i.e., the receiver receives proper money. Only the transactions with the above two validities are supposed valid. However, the two validities for a cross-shard transaction are separately verified by two shards. As described in Section II, in the complete sharding [10], the cross-shard transaction needs to be divided into three sub-transactions, i.e., two internal transactions and one relay transaction, to be processed in the system.

Block Design. In Pyramid, each shard has a blockchain. Nodes in each i-shard store one blockchain. For each b-shard, besides its own blockchain, nodes in it store multiple blockchains, the number of which equals the number of i-shards bridged by the b-shard. Different from complete sharding, in Pyramid, nodes in b-shard can verify both the source validity and result validity of cross-shard transactions. Thus, the nodes in a b-shard can propose blocks including valid cross-shard transactions among the related i-shards, for which we propose a new cross-shard block structure as follows.

Each cross-shard block is composed of a *header* and a *body*. The header includes the hashes of parent blocks in the related i-shards and the Merkle tree root of the body. The body includes transactions and states involving with the related shards. Fig. 3 illustrates a cross-shard block related to i-shard A and B. The body includes the cross-shard transactions between i-shard A and B, i.e., Tx list (A $\rightarrow$ B) and Tx list (B $\rightarrow$ A), and the states of accounts in i-shard A and B, i.e., State list (A) and State list (B).

Although a valid cross-shard block can be proposed by any node in a b-shard, to guarantee the consistency of states between the b-shard and its related i-shards, the cross-shard block needs to be agreed by the other nodes in the b-shard and its related i-shards depending on a layered sharding consensus in the next subsection.

C. Layered Sharding Consensus

In Pyramid, each epoch consists of a number of consensus rounds. In each consensus round, a leader will be randomly elected based on the randomness of the epoch. For each i-shard, its leader can propose a block including its internal transactions and the corresponding consensus process is similar with traditional complete sharding systems. For each b-shard, its leader can propose a block including its internal transactions or a cross-shard block associated with multiple i-shards. If the leader of a b-shard proposes a cross-shard block, the cross-shard block is committed via a layered sharding consensus as follows.

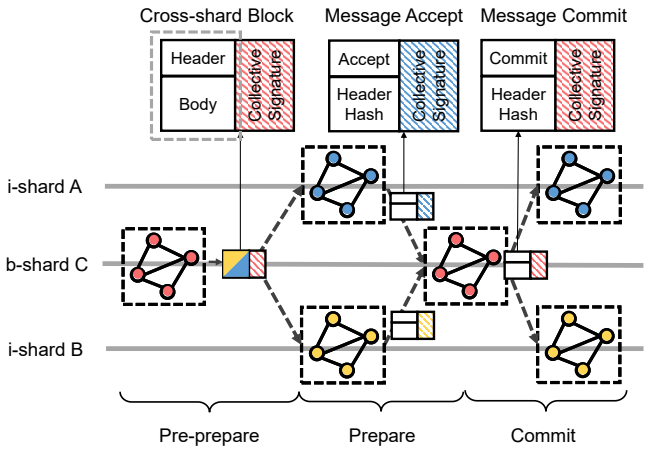


Fig. 4. Layered sharding consensus

Fig. 4 illustrates an example of committing a cross-shard block proposed by the leader in the b-shard. The procedure of committing the cross-shard block includes three phases:

1) Block Pre-prepare. As shown in Fig. 4, in this round, the leader of the b-shard proposes a cross-shard block related to i-shards A and B. According to Section IV-B, the nodes in the b-shard store blocks of both i-shards A and B, thus they can validate the transactions related to accounts in i-shard A and B. Therefore, the b-shard can treat the cross-shard transactions as its internal transactions.

The nodes in the b-shard achieve a consensus of the cross-shard block via a BFT protocol. For scalability, we adopt CoSi [18], a collective signing protocol that can efficiently scale to thousands of nodes, similar to the ByzCoinX in Omniledger [8]. This protocol can generate a multisignature co-signed by a decentralized group of nodes. A cross-shard block with a collective signature of a two-third super-majority of nodes attests that the b-shard agrees on it under a Byzantine environment. Besides, the collective proof constructed by echoes message using BFT in RapidChain [9] also works.

However, the cross-shard block cannot been committed yet, because the nodes in associated i-shards do not know the block and have not updated their local states according to the cross-shard transactions. Thus, the cross-shard block with the collective proof of the b-shard will be sent to the nodes of associated i-shards, i.e., i-shard A and B.

2) Block Prepare. After receiving the cross-shard block with the collective proof from the b-shard, nodes in each i-shard can verify the collective signature based on public keys recorded in the identity blockchain. Then, each related i-shard can achieve a consensus for the block via the collective signing too. After the consensus is successfully reached, the i-shard sends a message of *Accept*, including the hash of header and a collective signature to the b-shard. Note that the condition for message of *Reject* is about the conflict among blocks, which will be given in the Section IV-D.

3) Block Commit. After the pre-prepare phase, nodes in the b-shard initialize a counter with the number of its related i-shards. When a node in the b-shard receives a valid message

of *Accept* from an associated i-shard, it will decrease its local counter and broadcast the message to other nodes in the b-shard. When the counter equals to 0, nodes in the b-shard can ensure that the cross-shard block can be committed in this round. As in PBFT, block prepare phase is insufficient to ensure that the block will be committed [6], thus an additional collective signing is needed to guarantee that the cross-shard block will be committed and a message of *Commit* including the hash of header and a collective signature will be also sent to the related i-shards.

D. Design Refinements

General Cases. The above description is based on a payment between two accounts, i.e., a transaction with single step. However, the multi-step transactions, i.e., transactions involving many interactions among accounts, are common in current systems. (The figures are provided in Section VII-C.) A multi-step transaction is a sequence of interactions among accounts. Each interaction involves two accounts and its function can be the money transfer, creation and function-call of smart contracts, etc [2]. The multi-step transaction can be a cross-shard transaction involving several i-shards and be committed by the b-shard bridging these related i-shards using the layered sharding consensus in the same way as described above.

Parallel Commitment. In the above design, a cross-shard block occupies the consensus round of the b-shard and its related i-shards, which hinders the commitment of internal transactions. However, internal transactions that do not conflict with the cross-shard block, i.e., involving different accounts, can be committed in each shard at the same round safely.

Therefore, we do a little improvement for Pyramid to identify conflicting blocks and commit non-conflicting blocks in each i-shard in parallel. In the prepare phase, based on the transaction list in the received blocks, each i-shard can identify the conflicting blocks. Next, based on the randomness of the epoch, each i-shard accepts one from all conflicting blocks randomly and rejects the other blocks. Thus, among all conflicting blocks, only one block can have message *Accept* and the other blocks only have message *Reject*, which stops the conflicting blocks to be committed and commits the non-conflicting blocks at the same round.

Storage Optimization. According to Section IV-B, a cross-shard block includes the account states of all its associated i-shards. After the cross-shard block is committed via the layered sharding consensus, each related i-shard needs to store the block, leading to storage redundancy of account states. To reduce the storage overhead, nodes in each related i-shard can ignore the state list of other i-shards in cross-shard blocks. For example, for nodes in i-shard A, the State list (B) in the cross-shard block is redundant thus they can ignore the State list (B).

Relay Mechanism. In the above design, each leader can propose cross-shard blocks related to at most its related i-shards. It raises the problem that to process all cross-shard transactions, every possible b-shard should exist, which is impossible because there are a limited number of nodes in the system. Thus, we develop Pyramid based on a relay mechanism for cross-shard transactions originating in [10].

The main idea of the mechanism is referred to Section II-C. In brief, it can divide a cross-shard transaction into a number of internal transactions. Similarly, in Pyramid, if each cross-shard transaction can be divided into several cross-shard transactions, each of which involves parts of related i-shards, the above problem can be solved. To adopt it in Pyramid, we take some minor modifications compatible with our above design. First, after a block is committed successfully, both its header and the collective signature will be broadcast to and stored in all nodes in the system. The body includes an additional list named *outbound transaction list*. The list consists of transactions whose senders belong to the related i-shard of the body but receivers do not belong to any related i-shards of the block. Thus, these outbound transaction can pass the source validity in this round. For example, the outbound transaction list of the block of Fig. 3 can include transaction whose senders are in i-shard A and receivers are in any i-shard except i-shard A and B, and these transaction cannot pass the result validity in this round because of the limitation of the leader. Although they are not completely committed in this round, the leader or any other nodes can send them and their corresponding Merkle tree path to the next step i-shards. Then, in the following consensus rounds, other leaders in i-shards or b-shards can use the Merkle tree path and the block header as a proof to continue the verification for these transactions.

V. Analysis

In this section, we give a theoretical analysis for a layered sharding system with S shards and compare it with a complete sharding system with S shards and a non-sharding system in the aspect of security, scalability and performance.

First of all, we define the layer distribution as $\mathbf{d} = \{d_1, d_2, \cdots, d_{S-1}\}$, where d_1 denotes the number of i-shards, other d_i denotes the number of b-shards bridging i shards and $\sum_{1 \le i \le S-1} d_i = S$. The layer distribution denotes the sharding structure for Pyramid. Note that a b-shard in the layered sharding can bridge $S-1$ shards at most. The complete sharding can be considered as a particular case for the layered sharding where all nodes are located in the i-shards, $d_1 = S$.

A. Security Analysis

For the security, we prove the *safety* and *liveness* property of layered sharding consensus. Similar to other consensus protocols [9], [12], [14], the safety indicates honest nodes agree on the same valid block in each round and the liveness indicates the block finality, i.e., every block proposed by the leader in each round will eventually be committed or aborted.

Theorem 1. *The layered sharding consensus achieves safety if there are no more than $v < \frac{1}{3}$ fraction of malicious nodes in each shard.*

Proof. Given no more than $v < \frac{1}{3}$ malicious nodes in each shard, the intra-shard consensus can guarantee the cross-shard block proposed by the b-shard is valid. Then, a message

along with a collective signature is honest because honest nodes are the super-majority, i.e., more than two-thirds, of the shard. Meanwhile, the message cannot be modified and forged because the collective signature can be used to detect forgery and tampering. Therefore, the communication among shards can safely proceed if there are no more than $v < \frac{1}{3}$ fraction of malicious nodes in the involved shards, which can guarantee that all related shards can receive the valid cross-shard block. The prepare phase and commit phase in the consensus similar to the two-phase commit protocol in other distributed systems [11], [12]. The prepare phase aims to reach the tentative agreement of commitment for cross-shard transactions and the commit phase aims to perform the actual commit of the transactions among the related shards. Thus, honest nodes in all related shards including i-shards and b-shards agree on the same valid cross-shard block in each round, i.e., the consensus achieves safety. □

Theorem 2. *The layered sharding consensus achieves liveness if there are no more than $v < \frac{1}{3}$ fraction of malicious nodes in each shard.*

Proof. According to the system model in Section III-A, because the nodes are connected by a partially synchronous network and each shard has no more than $v < \frac{1}{3}$ malicious nodes, the BFT protocol adopted as the intra-shard consensus of each shard can achieve liveness. According to Theorem 1, each shard agrees on the same block in each round. Therefore, no malicious nodes can block the consensus indefinitely and each block will be eventually be committed or aborted, i.e., the protocol achieves liveness. □

Therefore, the system failure can be defined as the event that there exists a shard with no less than $1/3$ fraction of malicious nodes in the system. In Pyramid, the number of nodes in each shard is $n = N/S$. According to the cumulative hypergeometric distribution function same as [9], [12], the upper bound of the probability for the system failure in each epoch $Pr[Failure]$ can be computed by

$$Pr[Failure] < S \sum_{i=\lfloor n/3 \rfloor}^{n} \frac{\binom{fN}{i}\binom{N-fN}{n-i}}{\binom{N}{n}}. \quad (1)$$

By adjusting the number of nodes N and number of shards S according to (1), i.e., to satisfy that the system fails in each epoch with low probability, i.e., $Pr[Failure] \leq 2^{-\lambda}$ where λ is the security parameter in our system, Theorem 1 and 2 can be achieved, i.e., the layered sharding consensus achieves safety and liveness.

B. Scalability Analysis

For the scalability of a blockchain system, we define scalability parameter ω as the average complexity for the bandwidth, storage, and computation in one node in the system. For example, a b-shard bridging i i-shards needs to receive, store, and validate the blocks of i i-shards besides its own task, thus its bandwidth, storage, and computation complexity is $\frac{i+1}{S}$. Therefore, the scalability parameter of a layered sharding system can be defined as

$$\omega = \frac{1}{S}\cdot\frac{d_1}{S} + \frac{3}{S}\cdot\frac{d_2}{S} + \cdots + \frac{d_{S-1}}{S} = \frac{d_1 + \sum_{2\leq i\leq S-1}(i+1)d_i}{S^2}. \quad (2)$$

According to (2), the scalability parameter of complete sharding is $\omega^{complete} = \frac{1}{S}$ and that of non-sharding is $\omega^{non} = 1$. The scalability parameter of layered sharding is between that of complete sharding and non-sharing blockchain and increases as the b-shards bridging more i-shards increase. In other words, when there are more b-shards bridging more i-shards in the system, the average complexity for the bandwidth, storage, and computation will increase.

C. Performance Analysis

For the performance of a blockchain system, we analyze the transaction throughput and confirmation latency of the system. The transaction throughput denotes the number of transactions processed by the system per second and the confirmation latency denotes the delay between the time that a transaction starts to be processed until the transaction is committed.

As discussed in Section II, the performance of a sharding system can be affected by cross-shard transactions, thus we need to consider the distribution of cross-shard transactions in the system when evaluating its performance. We first define the distribution of transaction step $\alpha = \{\alpha_1, \alpha_2, \cdots, \alpha_A\}$ in which α_k is the percentage of transactions involving k accounts and A is the largest possible number of accounts involving by a transaction. The distribution α can be collected from a real blockchain system. Next, the distribution α can be converted into another distribution $\beta = \{\beta_1, \beta_2, \cdots, \beta_S\}$ in which β_1 is the percentage of internal transactions and other β_k is the percentage of cross-shard transactions involving k shards via a simple sampling simulation.

For each shard, because of the communication complexity of the intra-shard consensus, the latency of a consensus round will be influenced by the number of nodes in the shard. Thus, we define the latency of a consensus round in a shard with n nodes as a decreasing function $f(n)$ [19], the specific form of which depends on the type of intra-shard consensus and the propagation method in the network. We define the maximum number of transactions processed in a shard in each consensus round as K.

According to Section II-C, in a complete sharding system, each cross-shard transaction involving k shards needs to be divided into k sub-transactions at least. Thus, the maximum transaction throughput of the complete sharding system with S shards is $TPS^{complete} = \frac{SK}{f(\frac{N}{S})}(\beta_1 + \frac{\beta_2}{2} + \cdots + \frac{\beta_S}{S})$. In a non-shard system, there do not exist cross-shard transactions thus the transaction throughput of a non-sharding system is $TPS^{non} = \frac{K}{f(N)}$. It is obvious that if there are more cross-shard transactions involving more shards in the system, the transaction throughput of complete sharding systems deteriorates while that of non-sharding systems remains unchanged.

According to Section IV, the nodes in a b-shard bridging i shards can process cross-shard transactions involving i shards

at most. Furthermore, for a cross-shard transaction involving more than i shards, the nodes in the b-shard can process i sub-transactions using the relay mechanism. Therefore, the maximum transaction throughput of a layered sharding system is $TPS^{layer} = \frac{SK}{f(\frac{N}{S})} \sum_{1 \le i \le S} \frac{d_i}{S} (\sum_{1 \le j \le i} \beta_j + \sum_{i+1 \le j \le S} \frac{i\beta_j}{j})$. When there are more b-shards bridging more i-shards in the layered sharding system, the transaction throughput of the system will increase.

In the complete sharding system, each of the sub-transactions for a cross-shard transaction needs to be committed in a round one by one, thus the expected confirmation latency is $CONF^{complete} = f(\frac{N}{S}) \sum_{1 \le i \le S} i\beta_i$. In the non-sharding system, the expected confirmation latency is $CONF^{non} = f(N)$. In the layered sharding system, the cross-shard transaction can be committed in a round if it is committed by a b-shard bridging its related shards, thus the expected confirmation latency is $CONF^{layer} = f(\frac{N}{S})$.

VI. Evaluation

A. Implementation

We implement a prototype of Pyramid in Go[1] for performance evaluation. For comparison, we also implement two complete sharding prototypes. According to Section IV-C, since the intra-shard consensus in the layered sharding can be substituted by any other BFT consensus, to ensure the result will not be affected by the difference in intra-shard consensus, we adopt the ByzCoinX proposed in Omniledger [8] for the intra-shard consensus for these two complete sharding prototypes. The difference between two prototypes is the cross-shard transaction processing. The first one uses the relay mechanism in Monoxide [10] and the second one uses the transfer mechanism in RapidChain [9]. Their main ideas are referred to in Section II-C.

B. Setup

Similar to most running blockchain testbeds, the bandwidth of all connections between nodes are set to 20 Mbps and the links are with a latency of 100 ms in our testbed. In a consensus round, each node can verify up to 4096 transactions, each of which is 512 bytes. We generate a transaction data set in which the average step of transactions is set as 6. Furthermore, based on the data provided by XBlock [20], we can collect a historical transaction set of Ethereum. The security parameter λ is set as 17, which means the failure probability needs to smaller than $2^{-17} \approx 7.6 \cdot 10^{-6}$, i.e., one failure in about 359 years for one-day epochs.

C. Throughput

We measure the transaction throughput in transactions per second (TPS) for complete sharding and layered sharding with varying shard number S and scalability parameter ω and the result is illustrated in Fig. 5. The shard number S is set to 5, 7, 9, 11, 13, 15, 17, 20 and the scalability parameter ω is set to 0.2, 0.3, 0.4, 0.5. The layer distribution $\mathbf{d}$ is set based on the scalability parameter. The fraction of malicious nodes f is set as 12.5%. To guarantee the safety and liveness with high probability, according to (1), we adjust the number of shards when the number of nodes changes. As shown in Table I, all cases of the experiment in Fig. 5 satisfy $Pr[Failure] < 2^{-17}$. Fig. 5 illustrates that the TPS of layered sharding increases $1.5 \sim 2$ times when doubling the shard number, and the TPS increases by about 50% compared with two complete sharding prototypes for every 0.1 growth of scalability parameter ω.

[1]https://golang.org/

TABLE I
Failure probability for varying shard number and node number when percentage of malicious node $f = 12.5\%$

Shard number S, Node number N			
Failure probability $Pr[Failure]$			
5, 500	7, 1000	9, 1500	11, 2000
$1.5 \cdot 10^{-9}$	$8.4 \cdot 10^{-12}$	$5.1 \cdot 10^{-13}$	$1.4 \cdot 10^{-13}$
13, 2500	15, 3000	17, 3500	20, 4000
$2.8 \cdot 10^{-14}$	$3.9 \cdot 10^{-14}$	$1.7 \cdot 10^{-14}$	$1.1 \cdot 10^{-13}$

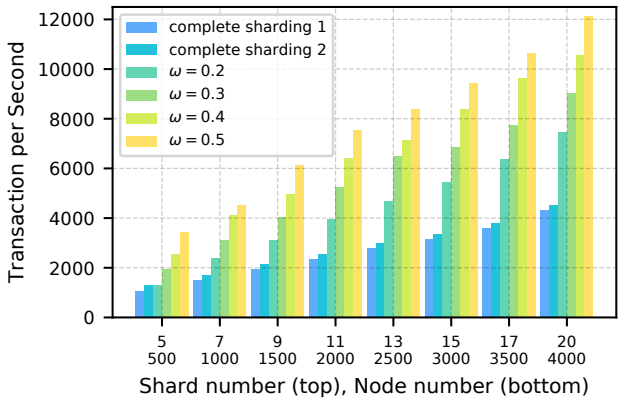


Fig. 5. Transaction throughput for layered sharding with varying shard number S and scalability parameter ω

We also evaluate the transaction throughput and failure probability influenced by the varying fraction of malicious nodes in Pyramid with shard number $S = 17$ and the scalability parameter $\omega = 0.3$. In our implementation, a malicious node can work when it is elected as a leader, and it generates and broadcasts the wrong block to interfere with the normal running of consensus. Figure 6 illustrates that as the fraction of malicious nodes increases, the TPS of Pyramid decreases in the approximate proportion. It is because the layered sharding consensus can guarantee that the blocks proposed by malicious nodes will be detected and aborted. Furthermore, as shown in the figure, the failure probability increases when there are more malicious nodes in the system and the percentage of the malicious node needs to be less than 19% to satisfy $Pr[Failure] < 2^{-17}$.

The above experiments are based on the transaction set in which the average step of transactions is 6. However, in different systems, the average step of transactions is different. Furthermore, the proportion of multi-step transactions is increasing over time because of the popularity of smart contracts

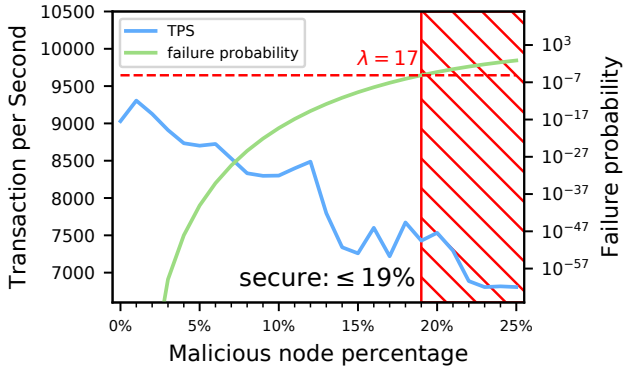


Fig. 6. Transaction throughput and failure probability for varying malicious node fraction with shard number $S = 17$ and scalability parameter $\omega = 0.3$

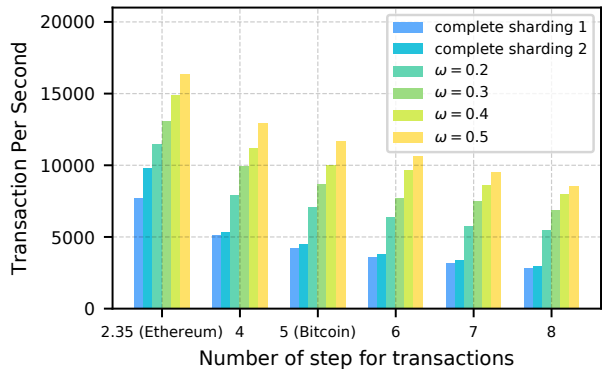


Fig. 7. Transaction throughput for transactions with different number of steps with shard number $S = 17$

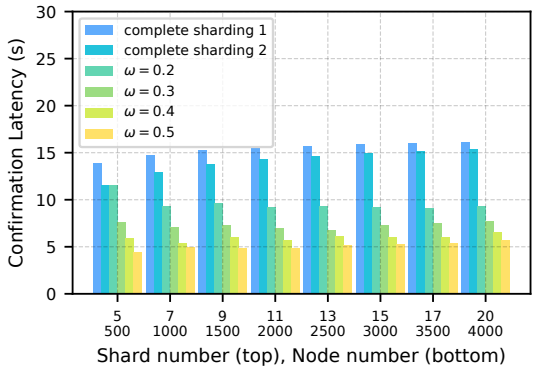


Fig. 8. Confirmation latency for layered sharding with varying shard number S and scalability parameter ω

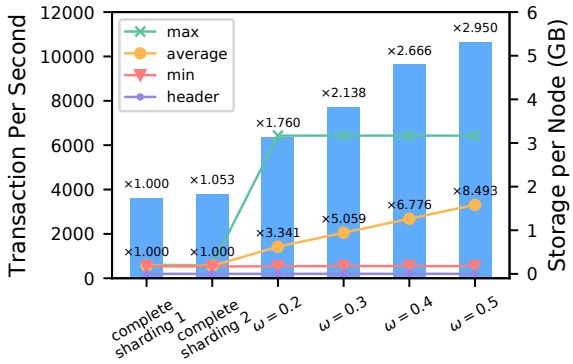


Fig. 9. Storage overhead per node after processing 1 million transactions for layered sharding with varying scalability parameter ω

which can produce transactions with more complex logic. We evaluate the performance of layered sharding when there are different numbers of steps for transactions in the system with 17 shards and 12.5% malicious nodes and the result is illustrated in Fig. 7. As shown in Fig. 7, the TPS decreases with the increase of the number of steps. Furthermore, the effect of the throughput boost of layered sharding is better when the transactions have more number of steps. Compared with complete sharding prototype 1, layered sharding with $\omega = 0.5$ achieves 2.12 times for transactions with 2.35 step and achieves 3.03 times for transactions with 8 steps.

D. Confirmation Latency

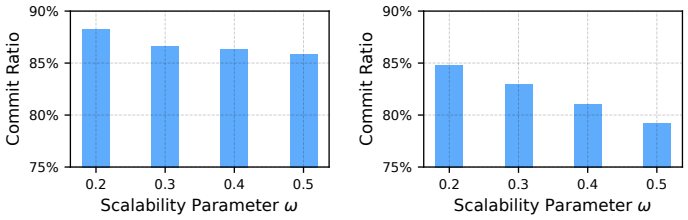
We then study the confirmation latency of transactions in complete sharding system and layered sharding system. As shown in Fig. 8, the latency increases as the number of shards increases and decreases as the scalability parameter increases. For the number of shards, it is because more shards in the system result in more cross-shard transactions. According to Section V-C, cross-shard transactions need more consensus round to be committed, thus the average confirmation latency of transactions increases. For the scalability parameter, it is because a higher scalability parameter denotes more b-shards in the system and a b-shard can commit cross-shard transactions related to the shards it bridges in one consensus round, which greatly shorten the confirmation latency.

E. Storage Overhead

We evaluate the storage overhead per node after processing 1 million transactions for different scalability parameter ω when there are 17 shards and 3500 nodes and show the evaluation result in Fig. 9. First, we can observe that as the scalability parameter ω increases, both the transaction throughput and average storage overhead increase. In particular, compared with complete sharding prototype 1, layered sharding with $\omega = 0.5$ can improve the transaction throughput to 2.95 times when introducing 8.49 times average storage overhead. Note that the main bottleneck in most sharding-based blockchain systems is the transaction throughput rather than storage overhead currently because the storage overhead can be solved by state-compaction process such as checkpoint mechanism [8], [21]. Second, we can observe that the maximum storage per node is unchanged in layered sharding, which is because the maximum storage mainly depends on the maximum number of i-shards bridged by a b-shard in the system. Finally, compared with the entire block, the storage of headers can be ignored.

F. Commit Ratio

According to Section IV-D, in each round, there may exist conflicting blocks that are proposed by different leaders but involve the state of the same accounts. Only one block can be committed among the conflict blocks. To study the block



(a) Shard number $S = 5$ (b) Shard number $S = 20$

Fig. 10. Commit ratio for varying shard number and scalability parameter ω

conflict in layered sharding, we measure the commit ratio for varying shard numbers and scalability parameters in Pyramid. The commit ratio denotes the proportion of committed blocks. Fig. 10 illustrates that as the scalability parameter increases, the commit ratio in the system decreases. It is because there are more b-shards in the system, which causes conflict probability higher and more blocks to be aborted. Furthermore, we can see that the decrease in the commit ratio for shard number $S = 5$ is less than that for shard number $S = 20$. It is because there are fewer accounts in a shard when the shard number increases, which causes a higher conflict probability.

VII. Discussion

A. Comparison with Shard Overlapping in Complete Sharding

In the sharding formation of RapidChain [9], shards can be sampled with or without replacement from the total population of nodes. If the complete sharding samples shards with replacement, nodes can be located in multiple shards, however, which is different from the layered sharding. In the complete sharding, although the nodes in multiple shards also store blocks of multiple shards, they only propose blocks involving one single shard in each consensus round, which will not improve the performance.

B. Heterogeneous Blockchain Node

In most sharding systems, the blockchain nodes are assumed to be homogeneous for computation, storage, and communication, which keeps a major bottleneck to the performance of sharding and violates the practice. Sharding complicates the structure of the blockchain system and makes it hierarchical. The computation, storage, and communication capacity of a node can influence its ability to process transactions, store historical data and relay message in the consensus of a sharding system. Therefore, it is important to consider the heterogeneity of blockchain nodes for improving the scalability of a sharding system. Furthermore, in practice, the nodes can be kinds of IoT devices or edge/cloud servers [22]. For example, in a blockchain smart grid, vehicles, roadside units, and gas/washing stations can be the blockchain nodes [23]. In a blockchain smart home, smartphones, personal computers, and other smart devices can be blockchain nodes [24]. Heterogeneous blockchain nodes with different capacity can be assigned to different layers in the layered sharding. For example, nodes with higher capacity of computation, storage, and communication can be assigned to the higher layer in our layered sharding system with higher probability, which will be considered in our future work.

C. Multi-step Transactions

For the account/balance model, there are a number of operations raising the interactions among smart contracts, including the creation of new contracts, function-call of and data-return from other existing contracts [2], [25]. According to the data provided by XBlock [20], we conduct an analysis for Ethereum from July 30th, 2015 to July 6th, 2019 and find more than 15% smart contract related transactions are composed by more than 1 steps and the average number of accounts in a transaction is 3.35. Furthermore, because of the popularity of more complex smart contracts, the proportion of multi-step transactions will be increasing over time. For the UTXO model, the Pay-to-Script-Hash popularizes multi-step transactions [26]. Based on Bitcoin data, the average number of inputs and outputs in a transaction is 2 and 3, respectively [27]. According to Section VI-C, the layered sharding can boost the transaction throughput more effectively when transactions have more steps, therefore we believe the layered sharding can make more sense in the future.

VIII. Conclusion

We present Pyramid, the first layered sharding blockchain system. At its core, Pyramid allows some shards storing the state of multiple other shards to form layered sharding structure thus handles cross-shard transactions efficiently. It exploits shards to work as a bridge among other shards to verify and process cross-shard transactions so that their atomicity and consistency can be guaranteed with lower overhead. In order to commit the cross-shard transactions safely and efficiently in the layered sharding, Pyramid propose a new structure of cross-shard blocks, and accordingly provides a layered sharding consensus that can scale well as the number of related shards for cross-shard blocks. Finally, the evaluation result illustrates that layered sharding provides improvements in many aspects. In particular, compared with complete sharding, layered sharding can improve the transaction throughput up to 2.95 times in a system with 17 shards and 3500 nodes. In the future work, we plan to study the dynamic layered sharding and the transactions placement method for layered sharding.

Acknowledgment

The work described in this paper was supported by the National Key Research and Development Plan (2018YFB1003800), the National Natural Science Foundation of China (61872310, 61802450), JSPS Grants-in-Aid for Scientific Research grant number JP19K20258, Hong Kong RGC Research Impact Fund (RIF) with the Project No. R5060-19 and R5034-18, General Research Fund (GRF) with the Project No. 152221/19E, Collaborative Research Fund (CRF) with the Project No. C5026-18G, the Natural Science Foundation of Guangdong (2018A030313005), the Program for Guangdong Introducing Innovative and Entrepreneurial Teams (2017ZT07X355), and the Pearl River Talent Recruitment Program (No. 2019QN01X130).

REFERENCES

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[2] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, 2014.

[3] Auctionity, "The world's largest blockchain auction house for crypto collectibles," https://www.auctionity.com/.

[4] OpenSea, "A peer-to-peer marketplace for rare digital items and crypto collectibles," https://opensea.io/.

[5] I. Eyal, A. E. Gencer, E. G. Sirer, and R. V. Renesse, "Bitcoin-ng: A scalable blockchain protocol," in *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*. USENIX Association, Mar. 2016. [Online]. Available: https://www.usenix.org/conference/nsdi16/technical-sessions/presentation/eyal

[6] E. K. Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford, "Enhancing bitcoin security and performance with strong consistency via collective signing," in *25th USENIX Security Symposium (USENIX Security 16)*. Austin, TX: USENIX Association, Aug. 2016, pp. 279–296. [Online]. Available: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/kogias

[7] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS 16)*. ACM, 2016.

[8] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "Omniledger: A secure, scale-out, decentralized ledger via sharding," in *2018 IEEE Symposium on Security and Privacy (SP)*, 2018.

[9] M. Zamani, M. Movahedi, and M. Raykova, "Rapidchain: Scaling blockchain via full sharding," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS 18)*. ACM, 2018. [Online]. Available: http://doi.acm.org/10.1145/3243734.3243853

[10] J. Wang and H. Wang, "Monoxide: Scale out blockchains with asynchronous consensus zones," in *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*. USENIX Association, 2019. [Online]. Available: https://www.usenix.org/conference/nsdi19/presentation/wang-jiaping

[11] M. Al-Bassam, A. Sonnino, S. Bano, D. Hrycyszyn, and G. Danezis, "Chainspace: A sharded smart contracts platform," *CoRR*, vol. abs/1708.03778, 2017. [Online]. Available: http://arxiv.org/abs/1708.03778

[12] H. Dang, T. T. A. Dinh, D. Loghin, E.-C. Chang, Q. Lin, and B. C. Ooi, "Towards scaling blockchain systems via sharding," in *Proceedings of the 2019 International Conference on Management of Data*, ser. SIGMOD '19. ACM, 2019.

[13] Y. Tao, B. Li, J. Jiang, H. C. Ng, and B. L. C. Wang, "On sharding open blockchains with smart contracts," in *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, 2020.

[14] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *Proceedings of the Third Symposium on Operating Systems Design and Implementation (OSDI 99)*. USENIX Association, 1999.

[15] G. Wang, Z. J. Shi, M. Nixon, and S. Han, "Sok: Sharding on blockchain," in *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*. ACM, 2019.

[16] S. Micali, M. Rabin, and S. Vadhan, "Verifiable random functions," in *40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)*. IEEE, 1999, pp. 120–130.

[17] D. Boneh, J. Bonneau, B. Bünz, and B. Fisch, "Verifiable delay functions," in *Annual International Cryptology Conference*. Springer, 2018, pp. 757–788.

[18] E. Syta, I. Tamas, D. Visher, D. I. Wolinsky, P. Jovanovic, L. Gasser, N. Gailly, I. Khoffi, and B. Ford, "Keeping authorities "honest or bust" with decentralized witness cosigning," in *2016 IEEE Symposium on Security and Privacy (SP)*, May 2016, pp. 526–545.

[19] Z. Ni, W. Wang, D. I. Kim, P. Wang, and D. Niyato, "Evolutionary game for consensus provision in permissionless blockchain networks with shards," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, May 2019, pp. 1–6.

[20] P. Zheng, Z. Zheng, and H.-n. Dai, "Xblock-eth: Extracting and exploring blockchain data from ethereum." *Working Report*, 2019.

[21] G. Avarikioti, E. Kokoris-Kogias, and R. Wattenhofer, "Divide and scale: Formalization of distributed ledger sharding protocols," 2019.

[22] W. Chen, Z. Zhang, Z. Hong, C. Chen, J. Wu, S. Maharjan, Z. Zheng, and Y. Zhang, "Cooperative and distributed computation offloading for blockchain-empowered industrial internet of things," *IEEE Internet of Things Journal*, Oct 2019.

[23] T. Jiang, H. Fang, and H. Wang, "Blockchain-based internet of vehicles: Distributed network architecture and performance analysis," *IEEE Internet of Things Journal*, June 2019.

[24] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram, "Blockchain for iot security and privacy: The case study of a smart home," in *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, March 2017.

[25] T. Chen, Y. Zhu, Z. Li, J. Chen, X. Li, X. Luo, X. Lin, and X. Zhange, "Understanding ethereum via graph analysis," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, 2018.

[26] A. M. Antonopoulos, *Mastering Bitcoin: Programming the open blockchain*. O'Reilly Media, Inc., 2017.

[27] B. Visuals, "Average number of inputs and outputs of a transaction in bitcoin," https://https://bitcoinvisuals.com/.