Hindawi
Mobile Information Systems
Volume 2022, Article ID 9864217, 1 page
https://doi.org/10.1155/2022/9864217

Retraction

Retracted: A Comprehensive Survey on Sharding in Blockchains

Mobile Information Systems

Received 6 May 2022; Accepted 6 May 2022; Published 31 July 2022

Mobile Information Systems and the authors have retracted the article titled "A Comprehensive Survey on Sharding in Blockchains" [1], due to substantial similarities with a previously published article by different authors [2].

The authors agree to the retraction and the notice.

References

[1] J. Xi, S. Zou, X. Guoai et al., "A comprehensive survey on Sharding in Blockchains," *Mobile Information Systems*, vol. 2021, 22 pages, 2021.

[2] G. Yu, X. Wang, K. Yu, J. Wei Ni, A. Zhang, and R. P. Liu, "Survey: Sharding in Blockchains," *IEEE Access*, vol. 8, pp. 14155–14181, 2020.

Hindawi
Mobile Information Systems
Volume 2021, Article ID 5483243, 22 pages
https://doi.org/10.1155/2021/5483243

Review Article

A Comprehensive Survey on Sharding in Blockchains

Jinwen Xi,[1] **Shihong Zou**,[1] **Guoai Xu**,[1] **Yanhui Guo**,[1] **Yueming Lu**,[1] **Jiuyun Xu**,[2] **and Xuanwen Zhang**[3]

[1]*School of Cyberspace Security, Beijing University of Posts & Telecommunications, Beijing, China*
[2]*College of Computer Science and Technology, China University of Petroleum (East China), Tsingtao, China*
[3]*Comprehensive Research Center of Electronic Information Technology, the MIIT of China, Weihai 264 200, China*

Correspondence should be addressed to Shihong Zou; zoush@bupt.edu.cn

Received 16 August 2021; Revised 28 September 2021; Accepted 17 November 2021; Published 18 December 2021

Academic Editor: Francesco Gringoli

Blockchain technology has been widely used in many fields, such as smart cities, smart health care, and smart manufacturing, due to its anonymity, decentralization, and tamper resistance in peer-to-peer (P2P) networks. However, poor scalability has severely affected the widespread adoption of traditional blockchain technology in high-throughput and low-latency applications. Therefore, based on the three-layer architecture, this study presents a variety of solutions to improve the scalability of the blockchain. As the scale of the network expands, one of the most practical ways to achieve horizontal scalability is sharding, where the network is divided into multiple subnetworks to avoid repeated communication overhead, storage, and calculations. This study provides a systematic and comprehensive introduction to blockchain sharding, along with a detailed comparison and evaluation for primarily considered sharding mechanisms. We also provide the detailed calculations and then analyze the characteristics of existing solutions along with our insights.

1. Introduction

Since the publication of Bitcoin, the most famous peer-to-peer (P2P) payment system presented in 2008 [1], the concept of blockchain has spread across the world. The blockchain technology is well known for leading to an increased transparency, decentralization, and tamper resistance. In recent years, these benefits have motivated the wide application of blockchain to almost all industry segments, including cryptocurrencies, Internet of Things, supply chain finance, social welfare, government affairs, and artificial intelligence [2–4]. Blockchain (the technical logic architecture is shown in Figure 1, including seven layers) is regarded as another disruptive technology after cloud computing, the Internet of Things, and Big Data. It is highly concerned by governments, financial institutions, and technology companies in various countries.

However, the scalability issue of blockchain has always been the focus of the industry, which means whether the era of large-scale commercial usage of blockchain will really come. In particular, Bitcoin [1] can only process 7 transactions per second, Ethereum [5] can process 15 transactions per second, and EOS [6] can process up to hundreds of transactions per second. The number of transactions that can be processed per second, that is, throughput, is far from the actual transaction processing requirements. Improving transaction throughput has become a stuck neck problem for blockchain to be implemented in multiple scenarios. By contrast, Visa can process 1, 700 transactions per second. That is, the scalability issue of blockchain needs to be solved urgently.

In the past literature, scalability is not well defined. However, Vitalik Buterin, the co-founder of Ethereum [7], firstly described the well-known scalability trilemma, stating that trade-offs are inevitable between three significant properties of blockchain: decentralization, security, and scalability. Decentralization is the core and the intrinsic nature of blockchain, and security is an essential property, whereas scalability is the main challenge. We can only have two of either decentralization, security, or scalability at the same time (i.e., we can pick just one side of the triangle that is shown in Figure 2). Thus, trade-offs are almost inevitable.

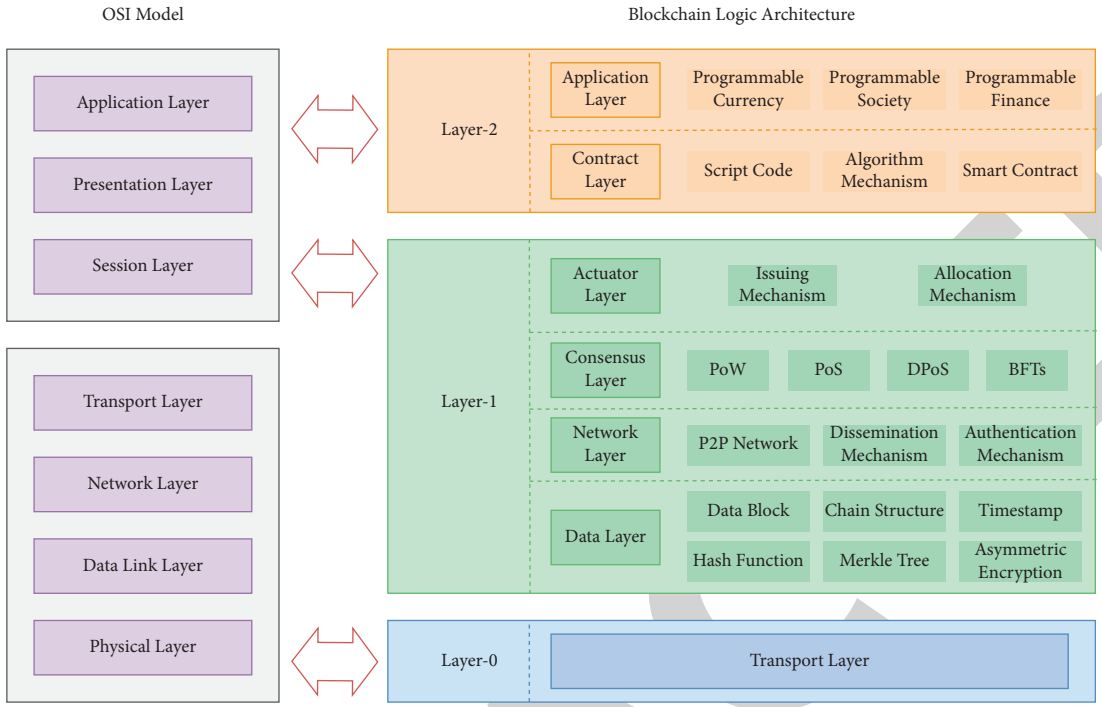

Figure 1: Blockchain technical logic architecture.

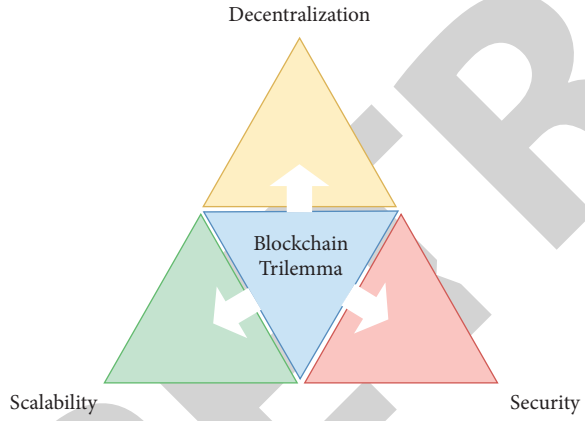

Figure 2: Blockchain trilemma.

That is, it is challenging to improve scalability without sacrificing decentralization and security in a blockchain-based system. There are several solutions proposed in the literature to solve the scalability issue. In this study, we classify these solutions into three categories in Table 1 according to the blockchain technical logic architecture: Layer 0 solutions (e.g., BDN [8] and bloXroute [9]), Layer 1 solutions (e.g., Segregated Witness [10], DAG [11–13], sharding [14–17], and consensus [18–20]), and Layer 2 solutions (e.g., state channels [21], side chain [22], cross-chain [23], and off-chain computation [24]). Layer 0 solutions are designed for improving the scalability by changing the underlying data transmission protocol of blockchain; Layer 1 solutions are designed for improving the scalability by changing the basic blockchain protocol such as block data structure, consensus algorithms, and incentive measures; and Layer 2 solutions are designed for improving the scalability through the off-chain methods at the application layer.

However, these proposed solutions cannot make significant performance gains without sacrificing decentralization or security, or both. In particular, the scalability solution that preserves both decentralization and security is sharding, which creates multiple groups (called shards) of validators and lets them process transactions in parallel [25]. Thus, we focus on sharding in this study, which is considered one of the most promising solutions to the scalability issue.

New blockchain-based sharding solutions have been proposed in recent studies [10, 25–28]. Surveys of blockchain scalability [29–34] that only focused on vertical scaling and reducing overhead have been gradually taking blockchain sharding into account. However, none of them can systematically introduce the characteristics and restrictions of existing sharding solutions, as well as the challenges and future trends.

1.1. Summary of Main Contributions. The main contributions of this study can be summarized as follows:

(1) This study is the first to provide a comprehensive introduction of blockchain scalability mechanisms ranging from technical logic architecture to various scalability schemes. Sharding is considered as the scalability solution that preserves both decentralization and security.

(2) Then, we survey and compare shard-based blockchain mechanisms, giving our insights to analyze the features and restrictions of existing mechanisms,

Table 1: Description of scalability solutions.

No.	Category	Layer	Solution	Description
1	Scaling on-chain	Data layer solutions	Expand block	Increasing block size
2			Segregated Witness	Isolating digital signatures and increasing the block size in disguise
3			Directed acyclic graph (DAG)	Changing chain structure to net structure
4		Network layer solutions	Sharding	Parallel processing of transactions in shards to improve processing efficiency
5		Consensus layer solutions	BFT consensus	Reaching consensus through voting
6			Non-BFT consensus	Reducing block verification propagation time and consensus formation time by reducing the complexity of the consensus algorithm and reducing the number of propagation nodes
7			Hybrid consensus	Combining multiple consensus algorithms
8	Scaling off-chain	Layer 2 solutions	State channels	Transferring part of the transaction to the state channel, which can be opened and closed at any time
9			Side chain	Transferring transaction to the side chain and processed by the root chain only when fraud occurs
10			Cross-chain	Realizing transactions off-chain from a cross-chain perspective
11			Off-chain computation	Executing complex transactions off-chain, and the result is returned to the chain
12	Scaling on Layer 0	Layer 0 solutions	BDN/bloXroute	Improving the scalability of the blockchain by modifying and optimizing the underlying protocol, without affecting the existing blockchain

including intrashard consensus protocol security, cross-shard transaction atomicity, and general improvements.

(3) Finally, we propose a comprehensive comparison of all the considered sharding mechanisms and obtain the theoretical upper bound of throughput through a large number of calculations and simulation experiments.

1.2. Paper Organization. The remainder of this study is organized as follows. We review the existing related works in Section 2 and then present the overview of sharding technology in Section 3. We provide a systematic survey of sharding solutions in Section 4, upon which the comprehensive comparison and evaluation are presented in Section 5. Finally, we conclude our survey in Section 6.

2. Related Work

We have investigated the existing studies [30, 33, 35–37] on the scalability of the blockchain and discussed them in conjunction with our work. Note that previous studies have mainly emphasized the development trend of future blockchain scalability and tried to scale the blockchain network. Besides, some researchers introduce new ideas to enhance blockchain scalability. These solutions include but are not limited to block expansion, Segregated Witness, DAG, scalable consensus algorithms, and multichain architecture.

The previous surveys [18, 29, 31, 32, 34, 38, 39] include discussions of the above solutions, but little research on sharding. Sharding technology is not an innovative concept. It has been operating in traditional databases long before the advent of blockchain technology and is mainly used to optimize large commercial databases. The concept is to cut the data in the database into many data fragments and then allocate these fragments to different servers for storage. In this way, there will be no server overload problem due to a large number of data access requests in a short period of time. Sharding has been considered the most practical solution for horizontally scaling blockchain systems so far. Therefore, many researchers have proposed their sharding mechanisms; at the same time, some surveys have been published to summarize these solutions and propose new benchmarks. However, all of these studies only analyze and evaluate one or two sharding mechanisms, and the introduction and evaluation of sharding are vague.

Some documents put their focus on sharding. Bez et al. [40] propose a scalable three-dimensional architecture and emphasize that horizontal scalability can only be improved by isolating data and consensus. However, its introduction to the Ethereum 2.0 sharding scheme is vague and [3] has the same problem. The separation of the consensus layer from the blockchain ledger data is inconsistent with reality. To benchmark sharding mechanisms, Manshaei et al. [41] propose an analytic model based on game theory. Furthermore, the Byzantine fault tolerance (BFT) algorithm [41, 42] is used to reach consensus between multiple committees in sharding, which is outdated as Ref. [43] proposes a sharding mechanism based on Nakamoto (Monoxide). In Ref. [15, 44], there is no unified comparison between Nakamoto-based sharding mechanisms and BFT-based sharding mechanisms.

To the best of our knowledge, there is currently no comprehensive overview of sharding. This article introduces the fundamental concepts related to various sharding mechanisms, comprehensively compares the more typical sharding mechanisms, and provides new guidance ideas for researchers based on our insights. Besides, we provide a comparison, based on the insights and calculation, among the considered sharding mechanisms. The result is characterized in Tables 2 and 3.

Table 2: A comprehensive comparison of considered blockchain scalability solutions.

Parameter		Monoxide	Elastico	OmniLedger	RapidChain	Chainspace	Ethereum 2.0	TEEChain
Network model (S: synchronization; PS: partial synchronization; AS: asynchronization)		PS	PS	PS	Intra S Total PS	PS	PS	PS
Randomness	Existence	No	Yes	Yes	Yes	Unknown	Yes	Yes
	Use	N/A	(1) The seed of PoW puzzle for the next epoch; (2) select the leader during intrashard consensus	(1) Select the leader and the subgroup allocation; (2) epoch reconfiguration; (3) trust-but-verify transaction validation scheme	(1) The seed of PoW puzzle for the next epoch; (2) select the leader during intrashard consensus; (3) decentralized bootstrapping; (4) epoch reconfiguration	Unknown	(1) Select the proposer and attesters of each shard; (2) select the validators responsible for checkpointing from the global pool	(1) Decentralized bootstrapping; (2) epoch reconfiguration
Participants	Node allocation	Address-based one-off allocation	PoW-based allocation	R-based allocation	PoW-based allocation	Object-based one-off allocation	R-based allocation	R-based allocation
	Safe epoch reconfiguration	N/A	Unsafe	Yes, 2/3 at a given time	Yes, swapping out a constant number of node	N/A	Yes	Yes, t/n
Transaction data structure		Account-based	UTXO-based	UTXO-based	UTXO-based	Object-driven, contract-sharded	Account-based	General workload
Global blockchain		No	Yes	Yes	Yes	No	Yes	Yes
Intrashard consensus algorithm		PoW-based Chu-ko-nu mining	PBFT	ByzCoinX	50% BFT	Mod start + PBFT	BFT-based PoS	TEE + PBFT
Security model	Threat model	Arbitrarily, uncoordinated majority	Arbitrarily, slowly adaptive	Arbitrarily, slowly adaptive	Arbitrarily, slowly adaptive	Arbitrarily, uncoordinated majority	Arbitrarily, uncoordinated majority	Arbitrarily, uncoordinated majority
	FT Intrashard	1/2	1/3	1/3	1/2	1/3	1/3	1/2
	Global	1/2	1/4	1/4	1/3	1/4	1/3	1/2
Cross-shard Tx	Support or no	Support	No	Support	Support	Support	Support	Support
	Method (S: synchronization; AS: asynchronization)	AS, lock-free	N/A	S, lock, or unlock	S, lock, or unlock	S, lock, or unlock	S, lock, or unlock	S, 2PC, and 2PL
Complexity	Communication	Mixed: $O(m + n \log_2 n)$ Identical: $O(m + n)$	$O(m^2 + n)$	$O(\log_2 m + n)$	$O(m^2 + m \log_2 n)$	$O(m^2 + n)$	$O(m^2 + n)$	$O(2^{-l}N^2)$
	Storage	$O(\lvert S\rvert) \sim O(\lvert S\rvert + n\lvert S_h\rvert)$	$O(n\lvert S\rvert)$	$O(\lvert S\rvert)$	$O(\lvert S\rvert + \lvert S_r\rvert)$	$O(\lvert S\rvert + \lvert S_{nh}\rvert)$	$O(\lvert S\rvert + n\lvert H\rvert + \lvert S_g\rvert) \sim O(n\lvert S\rvert + \lvert S_g\rvert)$	$O(\lvert S\rvert)$

Table 3: A comparison among the discussed sharding mechanisms.

Parameter		Monoxide	Elastico	OmniLedger	RapidChain	Chainspace	Ethereum 2.0	TEEChain
Shard settings	Number of shards (n)	$2^{10} \sim 2^{18}$	$<10^2$	$<2^6$	$<2^8$	$<10^2$	$<2^9$	$<2^5$
	Shard size (m)	$10^2 \sim 10^4$	$<10^2$	$2^2 \sim 2^{10}$	$2^2 \sim 2^8$	$<10^2$	$<10^2$	$<2^6$
	Epoch (e) length	N/A	10 min	24 hours	24 hours	Unknown	1 week	24 hours
Latency	Transaction confirmation	24 s	<800 s	90 s	60 s	3 s	7 s	5.9 ~ 15 s
	Epoch reconfiguration	N/A	N/A	1000s	200 ~ 300 s	Unknown	Unknown	80 s
Improving factor ($\mathcal{N}$)		$n/2$	n	$1 \sim n/2$	$n/2$	$1 \sim n/2$	$n/3$	$n/2$
Throughput		1.024 ~ 2.048 Mtps	38.4 ktps	24 ktps	128 ktps	<400 tps	86 ktps	16 ktps

3. Preliminaries of Sharding

3.1. Overview of Sharding Technology. In the traditional blockchain network, transactions must be confirmed by each node to ensure security, which is one of the main reasons why it is difficult to increase the transaction speed. In addition, most blockchains are single-chain structures, and all miners compete for the accounting rights of the next block, and most of the blockchains have a fixed average time for generating blocks. For example, Bitcoin generates a block every 10 minutes. Though more and more miners join the ranks of mining, the blockchain will automatically increase the mining difficulty to guarantee that the block generation speed is fixed at a block every 10 minutes. The increase in computing power cannot improve the transaction speed; this is why the blockchain is not scalable.

Sharding is first proposed by Ref. [45] and is commonly used in distributed databases and cloud infrastructure. Based on the pioneering proposals [14, 46], sharding technology is integrated with permissioned and permissionless blockchain. The way to apply the sharding technology to the blockchain network divides the blockchain network into several subnetworks (or shards). Each subnetwork will contain a part of the nodes. The data and transactions stored in the network will be randomly assigned to each shard. In this way, each node only needs to process a small part of the work and transactions in different shards can be processed in parallel, so the transaction speed of the network can be improved (as shown in Figure 3). Sharding is one of the most practical solutions to achieve a scaling blockchain system, where the calculation, storage, and processing can be conducted in parallel. It is possible that the capacity and throughput are linearly proportional to the number of shards or that of participating nodes.

3.2. Categories of Sharding. In sharding technology, the increase in blockchain computing power means an increase in the number of shards. That is to say, for the independent transactions (inputs and outputs are in the same shard), the more computing power invested in the network, the more transactions that can be processed in parallel at the same time, and the transaction speed of the entire network will increase linearly. The sharding technology used on the blockchain can be divided into three categories: network sharding, transaction sharding, and state sharding. Calculation sharding is considered to run through the entire sharding scheme [14].

3.2.1. Network Sharding. Because of the strong connectivity between blocks, that is to say, when a block is added, miners need to communicate with each other to confirm the validity of the new block, although this can ensure the accuracy of transactions, but also lead to the shortcomings of non-scalability in the blockchain structure. Therefore, the first step of sharding is to shard the blockchain network, on the premise of minimizing communication with each other. Each shard handles on-chain transactions. The miners are randomly grouped, and the work is assigned to each group of miners for independent verification.

There are two main schemes for network sharding to ensure the security of on-chain transactions:

(i) Random allocation: after the miners are grouped, the first problem of network sharding that needs to be solved is how to resist low-cost attacks. Assuming that under PoW, an attacker needs to master 51% of the computing power for paralyzing the network, and as sharding progresses, the attacker may only need to have 1% of the entire network's computing power to have the opportunity to paralyze one or even two of shards, but this 1% of the computing power does not work at all under the original PoW. Establishing randomness is an effective way to prevent attackers. The method of establishing randomness in the blockchain field is mainly to use verifiable random function (VRF) [47], which supports randomly selecting nodes to form shards. Such a random sampling method can prevent malicious nodes from controlling a single shard.

(ii) Consensus protocols of sharding: after solving the problem of grouping miners, we have to face the grouped miners and reach a consensus during the verification process. The consensus algorithms can choose PoW [48], PoS [49], PBFT [50], and others. In other words, network sharding is the mining rule. To avoid centralization while sharding, developers need to strike a good balance between security and efficiency, for example, the number of shards in the network and the number of nodes in each shard.

3.2.2. Transaction Sharding. Transaction sharding solves which transactions will be allocated to which shards. Then, the difference in the blockchain ledger model will have an impact on transaction sharding.

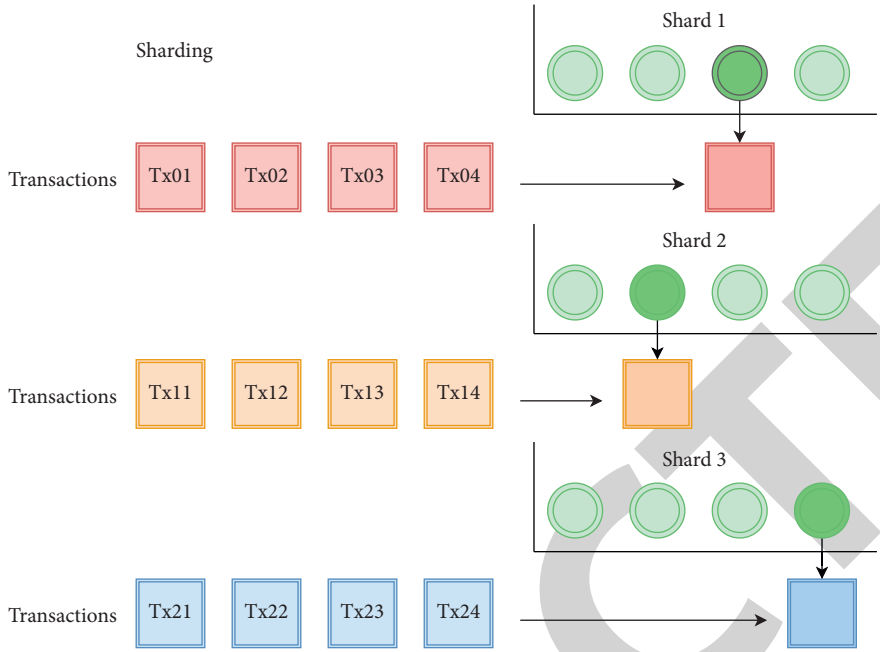

Figure 3: Sharding workflow.

(i) UTXO-based ledger model: UTXO means unspent transaction output, such as Bitcoin, and transaction records are composed of multiple inputs and multiple outputs [51]. There is no concept of an account and balance. It is impossible to split transactions according to addresses to avoid the double-spending problem effectively. The intuitive way of sharding is to divide transactions according to the last few bits of the transaction hash value. For example, if the last bit of the hash value is 0, the transaction will be allocated to shard 1; otherwise, it will be allocated to shard 2 (assuming there are two shards). However, if the initiator issues multiple transactions with the same input but different outputs within a certain period of time, if no measures are taken, different transactions may be allocated to different shards for processing and verification, leading to double-spending attacks. To solve the double-spending problem, communication between shards is necessary to ensure that the same amount is not repeatedly spent, but this violates the concept of independent verification of shards. Therefore, the UTXO ledger model is inherently difficult to achieve real sharding.

(ii) Account-based ledger model: therefore, most blockchains using sharding technology are based on the ledger account model [52] like Ethereum. On the basis of the account, each transaction will contain the sender's address and balance. The transaction can be partitioned according to the sender's address to ensure that multiple transactions from the same account are processed in the same shard, effectively detecting double-spending transactions. However, if when a cross-shard transaction is involved, cross-shard communication is required to verify the transaction validity. The difference from UTXO is that it only needs to be carried out between two shards rather than all shards.

3.2.3. State Sharding. State sharding can be understood as assigning blockchain data to different shards, thereby reducing the storage burden of nodes. Compared with the other two sharding mechanisms, state sharding is the most difficult. At present, three challenges need to be solved to implement state sharding:

(i) The benefit balance of cross-shard transaction communication: transactions will be processed according to the address allocation in different shards. When cross-shard transactions are involved, communication between shards is required to ensure the transaction validity. Fragmentation interaction can easily lead to a decrease in overall network efficiency;

(ii) The balance between the dynamic reconfiguration of shards and the update of node status: the nodes in the blockchain network will increase over time, and if the nodes are not redistributed for a long time, it will lead to an excessive centralization of the transaction state and reduce the flexibility when being attacked. Therefore, the network needs to reconfigure the network nodes after every epoch, the transaction must be processed after the segment completes the network synchronization, which will cause some delay problems, and even cause the segmentation to be paralyzed; and

(iii) The balance between network-wide data backup and centralization risk: if certain specific shards are

attacked and their verification becomes invalid, since the shards do not replicate the entire state of the system, the network cannot rely on verification of the transactions in this shard. Therefore, it is necessary to maintain data archives or node backups to help the system repair fault and restore unavailable data.

4. A Systematic Survey on Sharding

New challenges have been proposed in sharding, i.e., the intrashard consensus protocol security, cross-shard transaction atomicity, and general improvements regarding the reconfiguration, latency, and storage. It is significant for all sharding mechanisms to resolve these issues.

(i) Intrashard consensus protocol security: it is to ensure that the consensus protocol on transactions within a shard is kept away from Nakamoto-based 50% and BFT-based 1% attacks [40]. Note that BFT-based 1% attack is considered as a consensus-attacking strategy in a sharding network. Not as difficult as attacking the entire network, an attacker can easily control a single shard by breaking the intrashard consensus process [16].

(ii) Cross-shard transaction atomicity: as the sharding network expands, transactions between shards are inevitable. Cross-shard transaction verification and atomicity guarantee are important to the entire sharding system [17, 53, 54].

(iii) General improvements: due to the intrashard consensus security and cross-shard transaction atomicity, this study focuses on the improvement factor that optimizes the global throughput multiple of each considered sharding mechanism. Furthermore, we also consider the additional latency and overhead, and other new problems from the proposed sharding solutions. Therefore, we discuss some general improvements adopted in the considered schemes.

We take these three aspects as the survey methodology and focus on sharding in a permissionless blockchain network, which is based on the typically published papers and other research references, such as Monoxide [43], Elastico [14], OmniLedger [28], RapidChain [27], Chainspace [55], Ethereum 2.0 [56], and TEEChain [26]. The used notations of necessary parameters are shown in Table 4.

4.1. Intrashard Consensus Protocol Security. Sharding technology improves throughput at the expense of security, making blockchain systems more vulnerable to attacks.

In the traditional PoW consensus mechanism, due to the total mining power P, it is difficult for a single entity to control the whole network by dominating more than 50% network mining power. After sharding the network (assuming n shards), although throughput increases, the network mining power is thus distributed into the different shards. A single shard is more vulnerable to be attacked than the entire network, because ideally a malicious node would only need to control about $P/n \times 50\%$ mining power to manipulate the entire shard. As the scale of sharding increases, the situation will continue to deteriorate, which is why the PoW consensus mechanism is not suitable for the intrashard consensus.

Different from PoW-based consensus mechanisms, the BFT-based consensus mechanism is considered to be more effective in dealing with the abovementioned attacks. However, such design also introduces other problems as follows:

(i) Randomness generation: generating an unpredictable and unbiasable randomness is important for the sharding scheme independently in permissionless blockchains, which can be used for (1) validator allocation and (2) leader selection at the beginning of each epoch, meanwhile (3) determining the transmission route of a cross-shard transaction, etc. In this way, a strictly selected randomness is effectively used to prevent attackers from biasing the node allocation and controlling the leader election in a shard.

(ii) Shard size: the shard size is proportional to the strength of attack resistance for the BFT-based consensus algorithms. To improve security, the shard size should be increased as much as possible. However, an excessively high shard size will seriously affect the transaction throughput, resulting in greater overhead due to complex communication. We use the binomial distribution function to describe as follows:

$$F(x; m, p) = P[X \le x] = \sum_{i=0}^{x} \binom{m}{i} p^i (1-p)^{m-i},$$
$$\widehat{F}(x; m, p) = 1 - F(x; m, p), \tag{1}$$

where X represents an event that a malicious miner is randomly picked; m denotes the shard size; x indicates the number of malicious members in a shard; and p denotes the global fault tolerance. Note that $F(x; m, p)$ is suggested to be larger than 99% [57], while it requires $m > 143$, which will significantly reduce the efficiency of traditional BFT-based consensus algorithm.

Next, we discuss and compare the considered intrashard consensus mechanisms, including Monoxide, Elastico, OmniLedger, RapidChain, Chainspace, Ethereum 2.0, and TEEChain.

4.1.1. Monoxide: PoW-Based Chu-Ko-Nu Mining. Monoxide is the first to combine Chu-ko-nu mining with PoW-based algorithm for achieving intrashard consensus and solving the problem of computing power being diluted after partitioning, which makes attacking a single shard as difficult as attacking the entire system, where a Merkle Patricia tree (MPT) [5] is used to store all proposed blocks from multiple shards.

Table 4: Notation definition.

No.	Notation	Definition
1	P	Total amount of mining power
2	$\mathscr{B}$	Block
3	$\mathscr{H}$	Hash function
4	H	Block header
5	Sig	Signature
6	Tx	Transaction
7	k	Mining factor of shards
8	t	Total number of validators
9	n	Number of shards
10	m	Size of each shard
11	$\mathscr{R}$	Random seed
12	e	Epoch
13	$\mathscr{T}$	Block period
14	h	Block height
15	$\mathscr{N}$	Improving factor
16	S_h	Blockchain with a block height of h within a single shard
17	$\widehat{S}_h$	Header chain with a block height of h within a single shard

Chu-ko-nu mining is inspired by merged mining [58], which applies the PoW-based consensus and adopts a parent-child chain to share mining power. The parent chain with total mining power can protect these child chains with relatively smaller mining power from being attacked. Furthermore, an MPT root is involved to store all proposed blocks from different shards. Furthermore, the factor of k can be used to amplify the mining power; when $k \longrightarrow n$, the effective mining power that contributed by shards can achieve Pk/n≃P to address 1% attack.

In particular, the consensus algorithm in Monoxide is described as follows:

$$\mathscr{H}(\varphi \| \mathscr{H}(H \| \mathrm{MPT}_M)) \leq \epsilon, \tag{2}$$

where $\mathscr{H}$ represents the hash function; φ indicates the random nonce; H denotes the block header content; MPT_M represents the MPT root that aggregated by $[\mathscr{B}_0, \mathscr{B}_1, \ldots, \mathscr{B}_{n-1}]$ of each involved shard; and ϵ denotes the PoW target difficulty. φ, identities, and relay transactions are involved in each block.

Chu-ko-nu mining algorithm allows miners in any shard to generate multiple blocks when successfully solving the hash puzzle, but at most one block in each shard. In this way, the computing power of miners in each shard is amplified, so that the same level of computing power is required to attack a single shard as attacking the entire system.

Monoxide requires most miners in each shard to conduct Chu-ko-nu mining (i.e., $k = n$) to meet Pk/n≃P. Because scattered mining power may not guarantee the security of a single shard, on the other hand, it could result in overconcentration of mining power and generate additional overhead.

4.1.2. Elastico: BFT-Based. As shown in Figure 4, Elastico is the first to implement an intrashard BFT-based consensus protocol. In particular, Elastico allocates validators to different shards according to the least significant bits generated by the random solutions at the beginning of each epoch and then runs a classical Byzantine agreement protocol for intrashard consensus. Members agree and sign on one valid data block, and valid blocks have $2m/3 + 1$ signatures.

(1) Randomness Generation: A Decentralized Commit-And-Xor Scheme. This scheme is proposed to generate the randomness used in the mining process, which allows different random numbers picked by nodes to achieve unbiased and verifiably random. Each final committee member picks a random seed R_i and broadcasts the hash value $\mathscr{H}(R_i)$ to other committee members, and these members reach an agreement on a single set of hash values S and broadcast it to the entire network. Note that S must contain at least $2m/3$ signatures (m denotes the size of the final committee). Each final member broadcasts the selected R_i, and other nodes in the network will receive at least $2m/3$ and at most $3m/2$ $\{R_i - \mathscr{H}(R_i)\}$ pairs from the final committee members and discard the unmatched $\{R_i - \mathscr{H}(R_i)\}$ pairs. The final randomness can be finalized by randomly picking $m/2 + 1$ pairs from the remaining matched $\{R_i - \mathscr{H}(R_i)\}$ pairs for an XOR operation.

(2) Consensus Algorithm: Restrictions of PBFT in Sharding. According to the experimental results in Ref. [59], the transaction failure probability of Elastico is close to 8% with $F(x; m, p) = F(6; 16, 0.25)$. When the network scale is expanded to 100 shards ($m = 100$), it still incurs a 2.76% failure probability, which may be the bottleneck [28] to hinder Elastico from being practically applied. On the other hand, OmniLedger and RapidChain claim greatly improve this problem.

4.1.3. OmniLedger: BFT-Based. As shown in Figure 5, to resolve the bottleneck of Elastico, OmniLedger combines verifiable random function (VRF) [60] and RandHound [61] to generate an unbiased and unpredictable randomness, which is used to allocate members and select the leader. On the basis of ByzCoin [57], OmniLedger proposes ByzCoinX with larger shard size to resolve the BFT-based 1% attack in sharding.

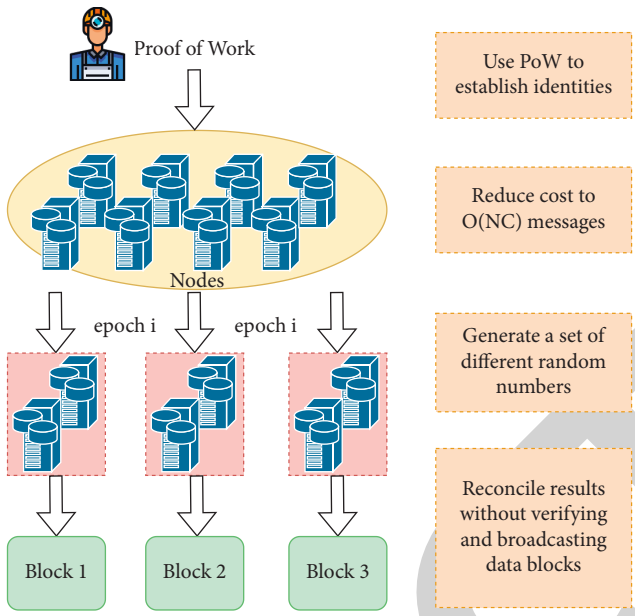

Figure 4: Node allocation of Elastico.

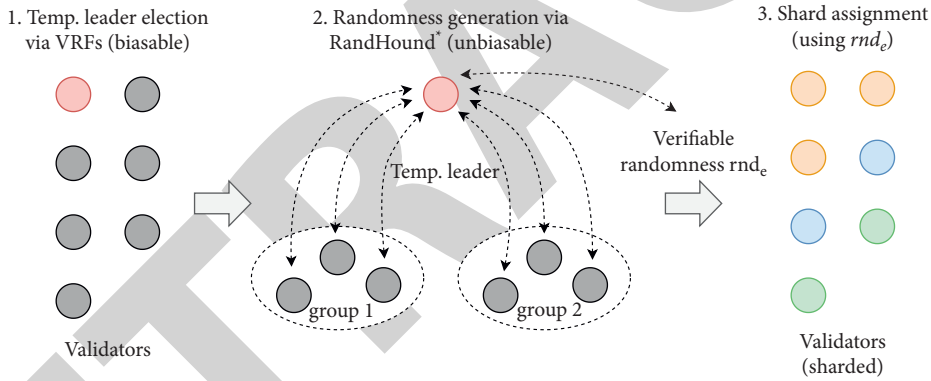

Figure 5: Shard validator assignment of OmniLedger.

(1) Randomness Generation: Combination of VRF and RandHound. OmniLedger combines RandHound [61] to propose a scalable bias-resistant distributed randomness generator to maintain the system availability and robustness, in which the publicly VSS (PVSS) [62] is applied to share the secret offline in advance. Furthermore, ByzCoinX introduces the Schnorr Signature [63] to reduce the communication complexity.

In particular, a VRF-based algorithm [60] is used for leader selection among participating validators randomly as follows:

$$\mathcal{R}_{e,\text{view},i}, \text{proof}_{e,\text{view},i} = \text{VRF}(\text{config}_e \| \text{view}, \text{sk}_i), \quad (3)$$

where config_e denotes the predefined settings of the genesis block; view denotes a view in Δ; $\mathcal{R}_{e,\text{view},i}$ represents the decided randomness; and $\text{proof}_{e,\text{view},i}$ represents the specific proof with epoch e and view for validator i.

(2) Consensus Algorithm: ByzCoinX. ByzCoin [57] proposes a hybrid scalable consensus protocol based on PoW and BFT, which implements a tree structure with signature collection to improve scalability. Compared with ByzCoin, ByzCoinX has lower latency and more robust fault tolerance for sharding. In particular, ByzCoinX implements a fixed three-depth shallow tree with an increased branching factor, generates the randomness at the beginning of current epoch and uses it to select the group leader, and then allocates the remaining validators to each group. Each group leader is responsible for managing a fixed number of members and forming a subtree according to the members. In addition, the group leader will maintain their roles until the next view occurs, thereby eliminating the shifting window and the difference between key blocks and micro-blocks. After aggregating more than 2/3 signatures from subtrees or leaves, the leader sends the signatures to the root node. Similarly, if at least 2/3 signatures are received, the decision in the root node can be finalized.

However, as the number of validators increases and exceeds the threshold, the tree branching keeps increasing,

and ByzCoin will outperform ByzCoinX in terms of latency. Furthermore, ByzCoinX improves the scalability by achieving an approximate 1.5% failure probability with $F(x; m, p) = F(50; 145, 0.25)$ and 1% communication overhead with $F(340; 1000, 0.3)$ by sacrificing transaction efficiency in large-scale networks. In addition, the combination of VRF and RandHound relies on the randomness initiated by the third party in the genesis block. There will be a fallback to inefficient solutions in an asynchronous network, which limits the scalability that RandHound can guarantee.

4.1.4. RapidChain: BFT-Based. To minimize the failure probability without sacrificing transaction efficiency, RapidChain [27] generates unbiased randomness by implementing the distributed random generation (DRG) protocol based on VSS [64]. RapidChain introduces a random graph to eliminate the third-party randomness limitation problem, which can guarantee a certain proportion (around 50%) of malicious validators [27].

(1) Randomness Generation: VSS-Based DRG Protocol. For fairness, RapidChain only introduces a basic VSS that shares scheme into DRG protocol to (re)allocate the participating validators to different shards.

(2) Consensus Algorithm: 50% *BFT with Pipelining.* In addition, RapidChain claims to increase the fault tolerance of the intrashard consensus up to 50% with a smaller shard size [65].

In particular, in each round of the consensus phase, each committee uses the epoch randomness to select a leader who collects transactions from the clients and packs them into blocks $\mathscr{B}_i$. The leader uses the information dispersal algorithm gossip (IDA-Gossip) protocol to broadcast $\mathscr{B}_i$ and creates a block header H_i containing the Merkle tree and the number of rounds. The committee members initiate a consensus protocol for the header H_i. In particular, the leader adds a proposed tag to the header H_i and broadcasts H_i through the gossip network. After receiving H_i, each node responds to and adds an echo tag to H_i, which is also broadcasted through the gossip network. If nodes see a different version of H_i, it means that the leader is malicious and they can reject it by gossiping a message with tag pending. On the contrary, if nodes receive more than $m/2 + 1$ echoes and there is only one version of H_i, it means that the committee has reached a consensus on block $\mathscr{B}_i$, and they accept the header gossiping and add an accept tag and a proof to H_i.

In addition, RapidChain proposes the pipeline processing to re-propose the pending blocks along with the new block for improving throughput. As shown in Figure 6, H_i is proposed in the iteration i, and if a new proposed block is replied with $m/2 + 1$ votes, it can be considered secure.

According to (1), the 50% BFT in RapidChain achieves a 1.5% failure probability with $F(x; m, p) = F(18; 35, 0.33)$ and 1% communication overhead with $F(50; 100, 0.40)$.

RapidChain prevents the BFT-based 1% attack and achieves 50% fault tolerance in intrashard consensus, differing from ByzCoinX in OmniLedger, which is suitable for small-sized shards. However, it is conceivable that the scheduled scheme that defines the timeout has not been proven to be sufficiently synchronized to run the 50% BFT of the pipeline.

4.1.5. Chainspace: BFT-Based. Chainspace implements the optimal PBFT in the proposed S-BAC protocol, called ModSMaRt [66]. However, it ignores the issue of 1% attack for scale PBFT and only decouples the communication and consensus process and at the same time reduces the consensus overhead by introducing a Validated and Provable Consensus protocol. However, Chainspace does not improve the high failure probability in intrashard consensus. Furthermore, the detail of randomness generator and leader selection is not provided.

4.1.6. Ethereum 2.0: BFT-Based PoS. Since 2014, Ethereum, as the second largest decentralized blockchain platform after Bitcoin, is the first to develop smart contracts to conduct transactions automatically, which is considered to be a Turing complete programming language (Blockchain 2.0).

With the gradually increasing transactions, the demand for high throughput increases. Shasper that sharding based on Casper FFG is proposed [67] to support Ethereum mainnet (Ethereum 1.0, the PoW-based single-chain architecture) to migrate to the new architecture (Ethereum 2.0, as shown in Figure 7) securely and stably. In particular, we mainly discuss Shasper on testnet and the intrashard consensus protocol, as well as the cross-shard transaction atomicity, because the other subprotocols have not yet been finalized.

(1) Randomness Generation: Combination of RANDAO and VDF. For generating randomness, Shasper combines RANDAO [68] and verifiable delay function (VDF) [69]. *RANDAO* is implemented through smart contracts on the beacon chain, which mainly contains three functions:

(i) Commit(): if all participating validators have deposited at least 32ETH, they will select a seed $\mathscr{R}$ and run a VDF as follows:

$$\mathrm{VDF}(\mathscr{R}_i) = \mathscr{H}(\mathscr{H}(\mathscr{H}(\ldots\mathscr{H}(\mathscr{R}_i)))), \tag{4}$$

where the iteration number of Hash() exceeds 10,000. As much, some malicious manipulation can be significantly prevented.

(ii) Reveal(): the local seed $\mathscr{R}$ of validators can be revealed through smart contract and be verified that whether it matches up with the commitment:

$$\mathscr{H}^{-1}\left(\mathscr{H}^{-1}\left(\ldots\mathscr{H}^{-1}\left(\mathrm{VDF}(\mathscr{R}_i)\right)\right)\right). \tag{5}$$

(iii) Generate(): a randomness is generated by merging all $\mathscr{R}_i$, and the validators that fail to reveal their own seeds in time will be punished.

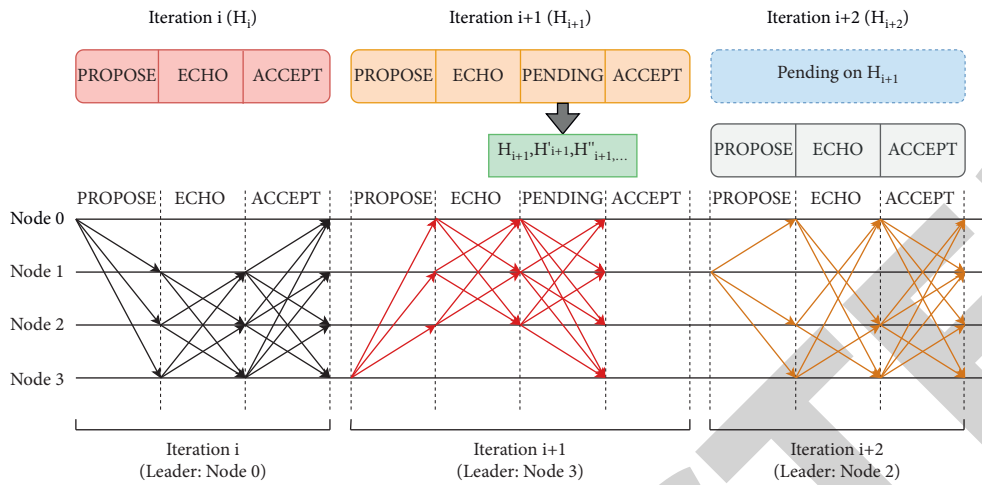

Figure 6: A synchronous BFT-based consensus protocol implemented by RapidChain.

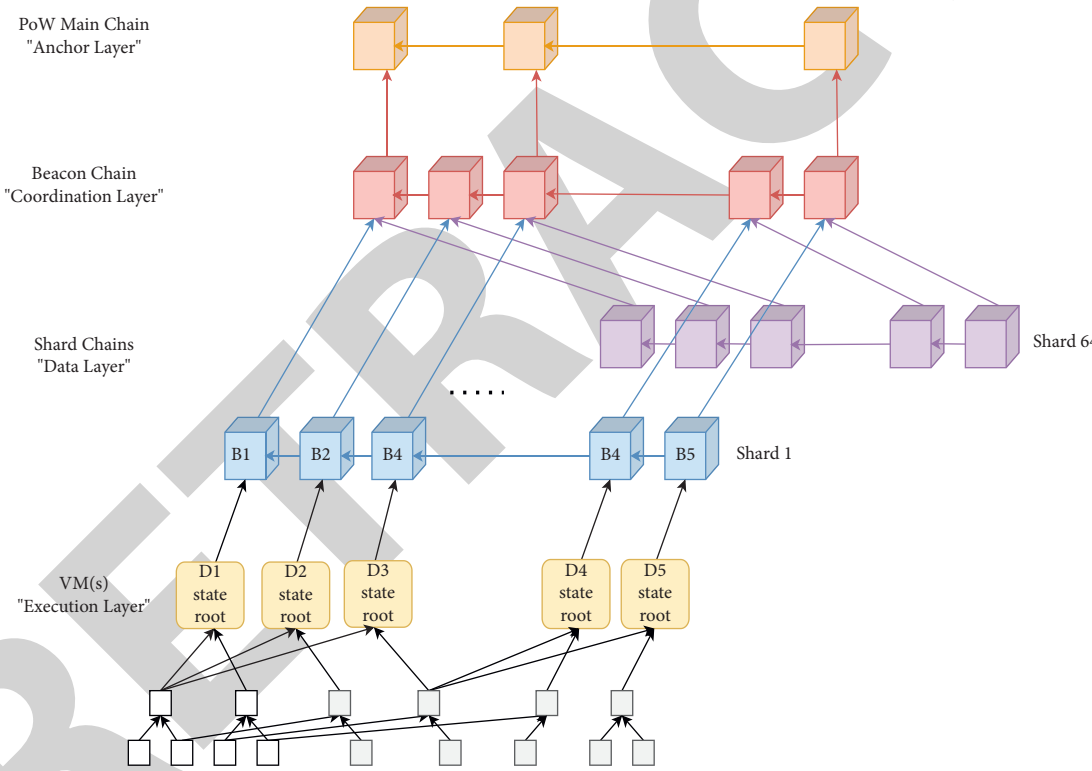

Figure 7: Ethereum 2.0.

However, n times $\mathcal{H}(\cdot)$ in VDF incurs a significant computation overhead of $O(n)$. Furthermore, the overall security is reduced due to the gas limit and malicious seed.

(2) Consensus Algorithm: Solving the Intrashard Consensus in a Global Way. To address 1% attack, Shasper applies the BFT-based algorithm to sharding, which decouples consensus process and node allocation. If a node wants to become a validator, it must deposit 32 ETH in an on-chain official contract. Having received the deposit, the beacon chain registers this node as a valid validator, which manages the validator pool. The validator needs to bet on the block in each round, while evaluating which block the other nodes will bet. If right, it can get the margin back plus the block bonus and gas fees for intrablock transactions. Otherwise, it can only get the margin back. Any malicious behaviors will be severely punished by the beacon chain. Due to the possible economic loss, the validator will choose to bet on the most favourable result. When the number of validators is enough, the betting algorithm ensures that the final result distribution tends to converge. Most of the validators will choose a high probability winning block as the eventual consistency result.

Considering that the previous shift of ETH miner from PoW node to PoS verification node may be lower than expected, Shasper is a hybrid mining scheme that overlaying the existing PoW with PoS, providing more flexible allocation for intrashard consensus. In particular, it remains the PoW mining until the PoS protocol is formally stabilized, but verification of PoS nodes will be triggered when every 50 blocks are generated, which allows some members of the community to participate in the testing of the new PoS chain, until the community makes a smooth transition to a pure PoS mainnet.

4.1.7. TEEChain: PBFT-Based in TEE. At present, most of the existing methods are to scale blockchain from the aspects of algorithms, protocols (software), etc. Later, some researchers improve the throughput and security from the perspective of hardware. The typical one is TEEChain [26] that is shown in Figure 8, which leverages the trusted execution environment (TEE) that combines with PBFT to eliminate equivocation.

(1) Randomness Generation: Combination of TEE and VDF. TEEChain works in different epochs (e.g., an epoch of 24 h), and each committee is dynamically shuffled at the beginning of each epoch. For preventing the attacker from selectively discarding the TEE output, leading to bias in random number generation, the TEE enclave can only be called once in each epoch. At the beginning of each epoch e, TEEChain performs the following steps to generate a random seed $\mathcal{R}$ that is uniform across the network:

(i) Generate two random numbers q and rnd

(ii) If q is zero, the node generates a signing certificate that contains $\langle e, \text{rnd}\rangle$ and broadcasts it to all other nodes

(iii) After waiting the time of Δ, lock the smallest collected rnd

(iv) Use the minimum rnd as the random seed $\mathcal{R}$ allocated by the committees for the current epoch

Once each node has the same rnd, it is possible to determine which committee a node belongs to. If the value of q that generated by all nodes is not equal to zero, that is, the certificate cannot be generated. Then, the epoch e is increased and the above steps are re-executed.

To ensure security, it is possible to tolerate up to $(m-1)/2$ nonequivocating Byzantine failures. Without equivocation, existing BFT protocols can achieve higher fault tolerance with the same number of nodes. In particular, the log stored in TEE can help prevent the attacker from tampering. TEE provides the proof of operations containing the signatures, which are required to accompany all valid messages.

In addition, there are two functions for processing in SGX (a TEE product): SGX_READ_RAND and SGX_GET_TRUSTED_TIME, which are used to generate unbiased random numbers and a timestamp, respectively, for realizing the assignment of committee members.

However, TEEChain relies too much on TEE in the process of randomness generation and consensus. If TEE fails, the blow to the system will be devastating.

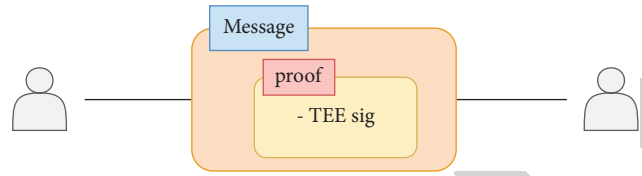

Figure 8: TEE-based PBFT in TEEChain.

4.2. Cross-Shard Transaction Atomicity. It is necessary to support cross-shard transactions and verification for a sharding mechanism. Therefore, some solutions [27, 28] propose to apply an individual global root chain to perform transaction verification, but if some additional measures are missing, the root chain cannot natively support cross-shard transactions. Therefore, the demand for cross-shard transaction security outweighs a naive communication protocol without supporting cross-shard protocol to achieve a higher improving factor $\mathcal{N}$. However, it is challenging to guarantee the data atomicity across multiple shards for supporting a cross-shard transaction protocol.

This section focuses on discussing and comparing the cross-shard atomicity protocols in the considered sharding mechanisms.

4.2.1. Monoxide-Relay Transactions. Monoxide proposes an asynchronous execution method, eventual atomicity, to bypass the overhead of state locking or unlocking, which decouples a single transaction into an initiated transaction and a relay transaction; the latter is added to the outbound transaction set. In this way, each consensus group can achieve efficient throughput without blocking each other. Eventual atomicity is not immediate atomicity, but a lock-free design. In particular, Chu-ko-nu mining algorithm is applied to maximize the global throughput in an asynchronous network.

As shown in Figure 9, a cross-shard payment transaction is generated that x tokens are transferred from user A to user B in different shards, which are decoupled into an initiate Tx and a relay Tx. In particular, the transaction that A transfers x tokens to B is successfully packaged, and then, relay Tx is used to transfer the tokens to shard b across shards.

In addition, transfers and receipts are represented as withdrawal and deposit operations, respectively, to guarantee the atomicity. As long as the withdrawal operation is confirmed, the deposit operation will eventually be confirmed. Therefore, Monoxide can achieve an improving factor of $\mathcal{N} = n/2$.

The uncertainty of cross-shard transactions in the originated shard and destination shard may result in additional transaction latency and replay.

4.2.2. Elastico-No Cross-Shard Transactions. Elastico applies the traditional PBFT consensus algorithm to select the leader in a shard, who confirms and transfers an agreement on local transactions to the final committee. After reaching consensus, the leader generates the final global block and broadcasts it to all validators and then adds it to the chain. All other validators can retrieve and

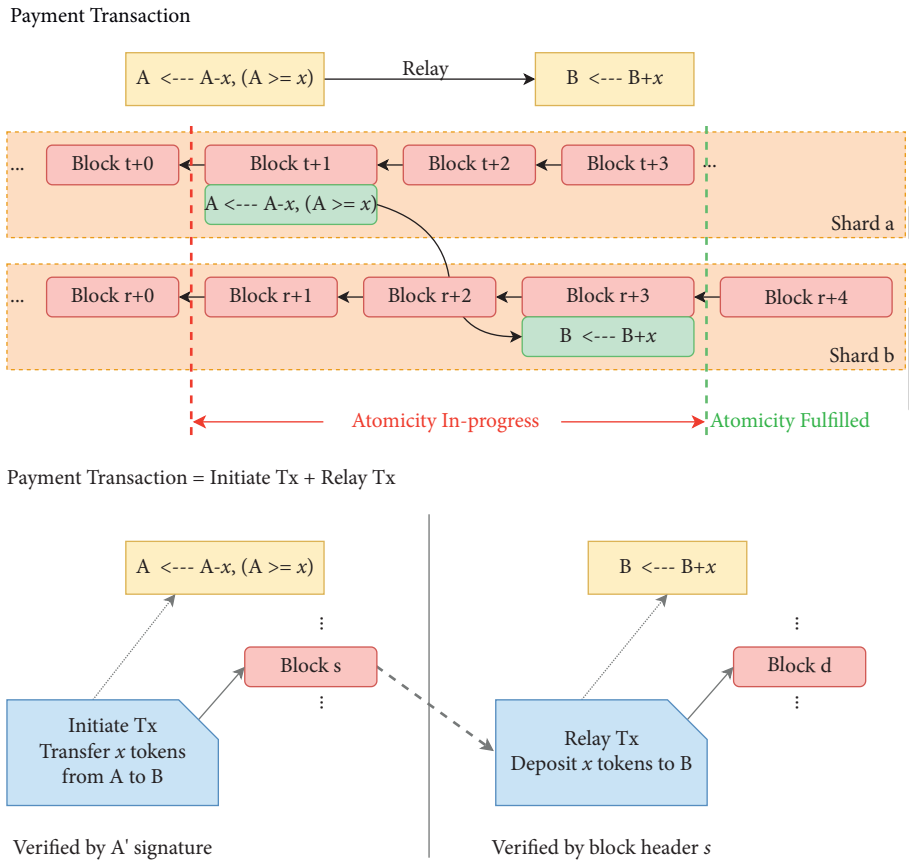

Figure 9: Monoxide atomicity.

verify these transactions. On the other hand, Elastico does not consider cross-shard transaction and fails to ensure transaction atomicity.

4.2.3. OmniLedger-Atomix Protocol. OmniLedger proposes a secure client-driven cross-shard transaction protocol (utilizing secure BFT shards) to ensure atomicity, which shifts the communication overhead outside the shards. As shown in Figure 10, the clients act as a gateway to support intershard messages exchange without direct communication between shards. The procedures are as follows:

(i) Initialize: a client creates a UTXO-based cross-shard transaction with two inputs and one output, and gossips the transaction to all input shards.

(ii) Lock: the shard leaders receive the cross-shard transaction from the client and create a proof of acceptance or a proof of rejection attached with the corresponding signatures according to UTXOs, to show that the verification result is successful or failed, respectively. If successful, the UTXOs will be locked in all shards of inputs.

(iii) Unlock: the client issues an unlock to commit consisting of the attached proof of acceptance and the locked cross-shard transaction and then gossips it to output shards after receiving all proof of acceptance from all input shards. Output shards will verify the transaction and store it in the local ledger. If a proof of rejection is returned, the client issues the state from unlock to abort and helps those input shards issuing a proof of acceptance to unlock the account.

Consequently, the Atomix protocol in OmniLedger can achieve an improving factor of $\mathcal{N} = n/2$. Besides, when the entire network is involved, it can achieve the worse improving factor of $\mathcal{N} = 1$.

OmniLedger ensures the UTXO transaction security by relying on a coordinated client. However, if the client is malicious, the client driver protocol will be blocked indefinitely, and the payer's funds may be locked forever. Furthermore, the extra overhead is inevitable for the client.

4.2.4. RapidChain: Transaction Routing. OmniLedger guarantees the atomicity of transaction processing. Still, it also has the following apparent shortcomings: first, the client needs to broadcast the transactions to the entire network, and at least one proof is generated and submitted for each transaction, causing a lot of communication overhead. Another problem is that the system relies too much on the client and requires the client to understand the entire network structure, which violates the principle of light client.

Differing from OmniLedger, RapidChain weakens the function of the client, who no longer needs to request asset

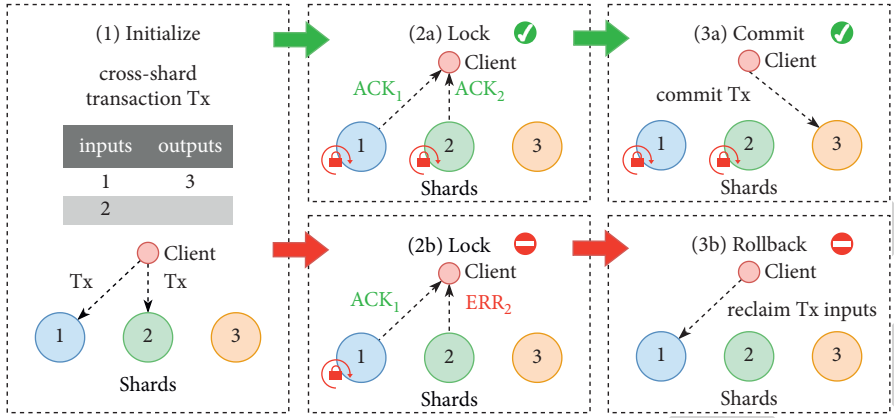

Figure 10: Atomicity protocol of OmniLedger.

proof from each input committee, nor does it need to understand the entire network. It only needs to send a transaction to any committee, and then, the transaction will be routed to the output committees through the inter-committee routing protocol.

If a cross-shard transaction Tx is generated with two inputs In_1 and In_2 and one output Out_3, as shown in Figure 11, which belong to different shard committees $Shard_1$, $Shard_2$, and $Shard_3$, then the leader of the output committee will create three new transactions Tx_1, Tx_2, and Tx_3, as follows:

(i) Tx_1 transforms the input In_1 from $Shard_1$ into the output Out_1 of $Shard_3$

(ii) Tx_2 transforms the input In_2 from $Shard_2$ into the output Out_2 of $Shard_3$

(iii) Through the above two transactions, all inputs and outputs belong to the same shard and then create a transaction Tx_3 with the fact that the input is the output of Tx_1 and Tx_2 and the output is the same as the original transaction

The leader of the output committee will send Tx_1 and Tx_2 to $Shard_1$ and $Shard_2$, respectively, through the inter-committee routing protocol. The leaders of $Shard_1$ and $Shard_2$ will verify the validity of the corresponding transaction. If valid, they will add the transaction to the local ledger and reply to $Shard_3$. The leader in $Shard_3$ will add Tx_3 to the local ledger after receiving all replies from the input committees.

Consequently, there are two types of transactions involved in the throughput, as shown by the $\mathcal{T}$ at the bottom of Figure 11; thus, it can achieve an improving factor of $\mathcal{N} = n/2$.

The RapidChain routing protocol draws on the design ideas of the Kademlia routing algorithm. Simply put, each node stores the routing information of all nodes of the same committee member and only stores the information of the $\log_2\log_2(n)$ nodes in the $\log_2(n)$ committees closest to the committee where it belongs, significantly reducing communication overhead. However, if the specific routing table is polluted, the eclipse attack [70] will be concerned.

4.2.5. Chainspace: The Interpart of S-BAC. The Sharded Byzantine Atomic Commit (S-BAC) Protocol proposes an optimal PBFT algorithm to achieve intrashard consensus and ensure atomicity. In particular, the elected shard leader that works as a BFT initiator is responsible for ensuring the cross-shard transaction atomicity. In this way, Chainspace composes the interpart of S-BAC.

Similar to OmniLedger, the key optimization in the BFT consensus process must be carried out, rather than reply on a simple client-oriented model. There are three same procedures involved in RapidChain, including initialize, lock, and unlock, with the same security problem. On one hand, $\mathcal{T}$ involves only one input and no output; thus, the improving factor is $\mathcal{N} = n$. On the other hand, when $\mathcal{T}$ involves all objects, the improving factor can be $\mathcal{N} = 1$.

4.2.6. Ethereum 2.0: Receipt-Based. Ethereum 2.0 applies the beacon chain to ensure cross-shard transaction atomicity. Differing from UTXO-based blockchain, only users with registered accounts can conduct transactions in Ethereum 2.0, while cross-shard transaction supported by smart contracts have not yet been implemented and released until now.

According to the primary transaction in Ethereum 2.0, Shasper can achieve an improving factor of $\mathcal{N} = n/3$.

Shasper applies accepted transaction receipts to verify and record the operation validity. In addition, involved validators can conduct cross-validation to obtain the operation results. According to the identities involved in receipts, the cross-shard transactions can be divided into many sub-transactions being executed in different shards, respectively. The idea of receipts is similar to the synchronous locking/unlocking scheme in OmniLedger and RapidChain, while the receipt plays the fundamental role of locking.

4.2.7. TEEChain: 2PC and 2PL. After analyzing the shortcomings of OmniLedger and RapidChain, TEEChain proposes the two-phase commit (2PC) and two-phase locking (2PL) protocol to achieve the cross-shard transaction communication between committees, while satisfying

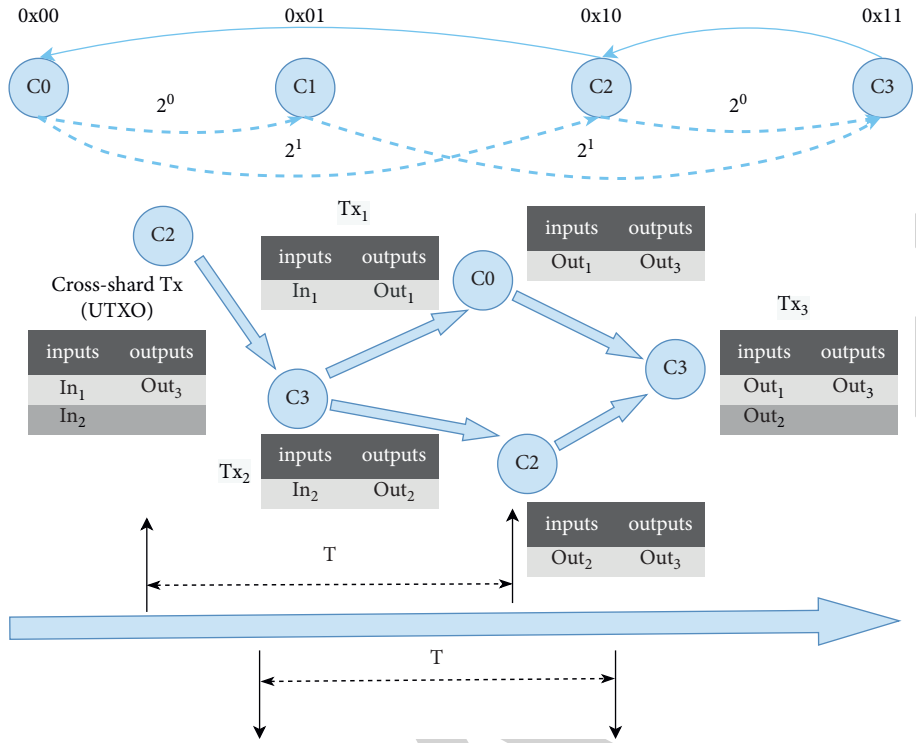

Figure 11: Atomicity protocol of RapidChain.

isolation and atomicity. The communication between committees is divided into three steps, namely prepare, pre-commit, and commit. Ref represents the reference committee acting as a coordinator in the two-phase commit, which consists of a group of nodes and runs PBFT to reach a consensus on transactions. S_1, S_2, and S_3 are the committees (shards) involved in this transaction.

Concretely, the reference committee implements a simple state machine for 2PC, as shown in Figure 12; when a transaction BeginTx is initialized by the client and sent to the reference committee, there are three procedures as follows:

(i) Prepare: once the Ref receives BeginTx, the state machine transitions to the started state. The nodes in the Ref send PrepareTx message to committees in different shards, and each shard committee executes PrepareTx and reaches a consensus on PrepareTx. If the consensus is reached and the transaction has been locked, the nodes in each shard committee will send a PrepareOK message to the Ref.

(ii) Precommit: after the state machine transitions to the started state, the reference committee Ref will create a counter to count the number of PrepareOK messages from different shards. Then, the state machine transitions to the preparing state during this process. Once it receives the PrepareNotOK message, the state machine transitions to the aborted state. Otherwise, when the counter reaches the counting threshold (count = 0), the state machine transitions to the committed state.

(iii) Commit: if the state machine transitions to the committed state, the nodes in Ref can broadcast the CommitTx message to all shards. Otherwise, the transaction will be discarded.

From the above three procedures, it can be seen that a temporary agreement for transaction submission is reached in the first stage (prepare and pre-commit), and the transaction between shards is actually submitted in the second stage (commit). Similar to RapidChain, TEEChain can achieve an improving factor of $\mathcal{N} = n/2$. Furthermore, based on the assumption of TEEChain, the BFT-based protocol running in Ref achieves liveness. However, the reference committee can be a bottleneck in cross-shard transaction processing.

4.3. General Improvements. This section has listed several general improvements in the considered sharding mechanisms for the critical technical challenges. These solutions can usually be implemented to solve the new problems caused by sharding to the entire system.

4.3.1. Reducing Transaction Latency. Users are sensitive to the waiting latency in the final transaction confirmation. To solve the BFT-based 1% attack, OmniLedger and Ethereum 2.0 implement a scalable BFT consensus. RapidChain increases the fault tolerance within a single shard, while remaining the issue of transaction latency.

According to the evaluation shown in Ref. [28, 57], the transaction latency deteriorates because the scalable BFT consensus solves 1% attack by increasing the shard size. Thus, OmniLedger proposes a two-level trust-but-verify transaction validation scheme to get low latency and high

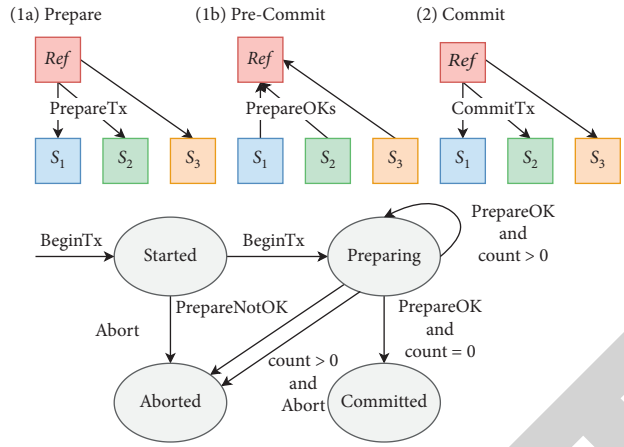

Figure 12: Atomicity protocol of TEEChain.

throughput, providing the real-time transaction confirmation within each shard, which is the same as Ethereum 2.0, as shown in Figure 13. Concretely, the validators are split into two groups of optimistic validators and core validators. To verify transactions in parallel, the optimistic group is further divided into multiple subgroups with several validators. Subsequently, the core validators conduct the second verification and determine whether there are inconsistent and malicious transactions. If none, the transactions can be stored in the shard ledger. However, optimistic validators may still suffer 1% attack after being grouped, and the real-time transaction is achieved at the expense of security, similar to IOTA [11], which can be applied in specific scenarios with lower security requirements.

Since the nonscalable 50% BFT-based consensus will incur more considerable communication overhead to increase the transaction latency, thus, under a 50% consensus, only the block summary is agreed. RapidChain implements a gossip protocol based on an information dispersal algorithm (IDA) [71, 72] to reduce transaction latency. In particular, the initial message sent by the sender is split into n fragments. An (α, β) erasure coding scheme is proposed to encode the β fragments to α fragments. By querying neighbors for β fragments, other nodes can reconstruct the initial message, thereby significantly reducing the communication latency.

4.3.2. Intershard Communication Protocol. The intershard communication protocol is different from the atomicity protocol that focuses on cross-shard transactions atomicity. It aims to reduce the data transmission overhead among shards. The first type of sharding mechanism, including Monoxide, Elastico, and Ethereum 2.0, implements a global root chain to transfer messages, where each validator needs to store the entire chain. The second type of sharding mechanism, including OmniLedger and Chainspace, applies the full-mesh connection mechanism to improve the communication efficiency by simplifying the communication process.

Inspired by the Kademlia-based routing algorithm [73], RapidChain establishes a routing table and proposes a novel intershard communication protocol to decrease communication overhead caused by the full-mesh connection, where each validator only stores the routing information of the closest $\log_2 \log_2 (n)$ nodes in the $\log_2 (n)$ committees, instead of all nodes in the network.

4.3.3. Shard Ledger Pruning. Most existing blockchain systems support a single-chain structure [1, 5, 74]. To improve communication efficiency and reduce the computational overhead of data audition and verification, they tend to require validators to store a full version of the ledger. However, for a sharding system, this will bring an unacceptable storage overhead to the validators. Therefore, OmniLedger prunes the shard ledger by applying state blocks (SBs) similar to the fixed checkpoint in PBFT [50] to aggregate all the states of the sharding ledger. To create a state block $\mathrm{SB}[j, e]$ in shard j at the epoch e, the validator performs the following steps:

(i) The leader of the shard j saves the UTXO to a sorted Merkle tree at the end of epoch e and stores the root hash in the header of the state block $\mathrm{SB}[j, e]$.

(ii) The shard validator runs a consensus on the header (note that each validator can construct the same sorted Merkle tree for validation).

(iii) If the consensus is successful, the leader will pass the block header and save it into the ledger, so that $\mathrm{SB}[j, e]$ becomes the genesis block of epoch $e + 1$.

(iv) Finally, UTXO in $\mathrm{SB}[j, e]$ can be safely discarded. To create a proof of transaction, we retain the normal blocks of epoch e until the end of epoch $e + 1$.

Furthermore, to avoid downtime, all validators in the epoch $e + 1$ shard need to include an empty state block as a substitute at the beginning of each epoch. However, such state block design can face the same client security problem like that in OmniLedger and Ethereum 1.0.

4.3.4. Decentralized Bootstrapping. The randomness generator involved in sharding is responsible for randomly generating an initial set of nodes that can participate in

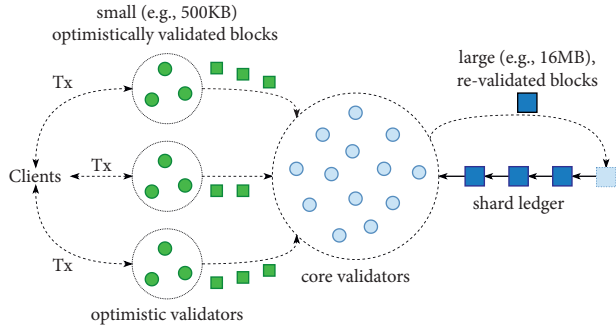

Figure 13: Two-level trust-but-verify validation scheme.

intrashard consensus, while ensuring that honest nodes account for the majority in the initial set.

Therefore, RapidChain uses a random seed to establish a sampler graph election network for supporting decentralized bootstrapping, where validators are uniformly assigned to several groups. Each member calculates the PoW-based result and its identity ID according to the random number generated by the DRG protocol based on VSS [27]. As shown in Figure 14, a subgroup can be obtained from each group. When this process is iterated, a unique root group can be made up of more than 50% honest validators. Therefore, the communication overhead can be reduced from $O(t^2)$ to $O(t\sqrt{t})$, where t represents the total number of validators.

4.3.5. Epoch Reconfiguration. Shards in blockchain are vulnerable to partition attacks; that is, within a single shard, offline or malicious nodes exceed the tolerable ratio, or the protocol cannot be performed normally. In severe cases, wrong information may be recorded on the ledger. Therefore, new validators must be reallocated to other shards in each epoch to prevent attacks from slowly adaptive adversaries.

The committee reconfiguration method of Elastico is to dismantle all committee nodes and reelect all committees, but this method has two fatal disadvantages: first, the reelection of all committees incurs more overhead and takes a longer time, resulting in the failure to significantly improve the performance of the sharding system; second, if the nodes are constantly disrupted, it is difficult to maintain an independent ledger for each committee. Since the original intention of state sharding is to store the ledger separately in the shards to reduce storage space, however, it is obvious that the method of all disruption and reconfiguration cannot achieve the desired effect.

Ethereum 2.0 frequently updates the intrashard consensus members and requires each validator to track the status for speeding up the reconfiguration phase. A random arrangement scheme is implemented by OmniLedger to replace the validators in each shard to ensure that the number of validators exchanged at a given time is limited by $k = \log(t/n)$. New registered validators in the global blockchain are randomly allocated to different shards. The number of honest validators is enough to reach a consensus, and the throughput is improved. However, such solution incurs a larger latency, which makes the 24-hour epoch not suitable for highly adaptive adversaries.

On the other hand, RapidChain integrates cuckoo rule [75, 76] to propose a lightweight reconfiguration protocol, named bounded cuckoo rule, which only allows a certain number of validators to exchange among committees after each epoch. In particular, the reference committee divides committees into two categories: active committees with a majority of active members and inactive committees with a majority of inactive members. When a new node joins, the reference committee randomly adds the node to an active committee and allocates a fixed number of nodes in the current committee to different inactive committees. This scheme incurs less overhead and supports a more frequent committee reconfiguration to suit more highly adaptive adversaries.

5. Comprehensive Comparison of State-of-the-Art Sharding Protocols

5.1. The Upper Bound of Throughput. In this section, we discuss and provide a comprehensive comparison regarding the above considered sharding mechanisms, ranging from the intrashard consensus protocol security to the cross-shard transaction atomicity, as well as the corresponding overhead. The result is shown in Table 2. Due to the poor performance of achieving sharding with smart contracts, Chainspace is not discussed in this section. Instead of considering the individual protocols, we map out the landscape by extracting and evaluating the high-level design themes in the considered blockchain sharding solutions.

Referring to Ref. [77, 78], we use local calculate-optimized servers with 200 Mbps bandwidth, 8vCPU of Intel Xeon, and 2 TB disk storage. To fill the bandwidth and conduct comparisons, we select bandwidth as the upper bound and define some parameters and initial values, such as $|H| = 500$ B, $|Tx| = 250$ B, and $|\mathrm{Sig}| = 65$ B. Then, we take the size of a single identity by $|\mathrm{ID}| = 32$ B.

In particular, the calculations of throughput and disk storage are presented as follows.

5.1.1. Monoxide. Monoxide is the sole sharding mechanism that supports PoW algorithm for intrashard consensus. We refer to Ethereum [5] and set $|\mathscr{B}| = 20$ KB, $\mathscr{T} = 10s$, $n = 2^{18} = 262,144$, $m = 128$, and $h = 1,000,000$.

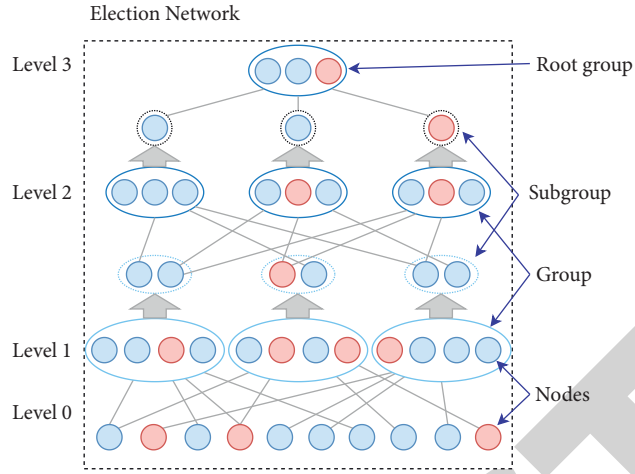

Figure 14: Election network.

(i) Throughput: the intrashard throughput can be calculated by $|\mathscr{B}|/(|Tx|\mathscr{T}) = 8.192$ tps. By multiplying the intrashard throughput and the improving factor of $n/2$, the entire network throughput of Monoxide can be calculated. Considering the different mixed/identical targets, the former is calculated by z with $n = 262,144$. The latter is calculated by $(8.192n)/2 = 2.048$ Mtps with $n = 524,288$. Therefore, the throughput of Monoxide is nearly from 1.024 Mtps to 2.048 Mtps.

(ii) Storage: H, Txs, and Sigs are involved in $\mathscr{B}$; thus, the storage space of $|\mathscr{B}|$ can be calculated by $h|\mathscr{B}| = |S_h| = 19$ GB. Furthermore, all participating miners are required by the Chu-ko-nu mining to store all received block headers to ensure security; therefore, much more storage space is needed in Monoxide.

5.1.2. Elastico. Similar to Monoxide, Elastico also requires validators to store all block data globally. In this way, we simply take the intrashard throughput as $800tps$ and set $|\mathscr{B}| = (800\text{ tps} \times Tx \times \mathscr{T}) + (2m|\text{Sig}|)/3 \simeq 1.9$ MB, where $\mathscr{T} = 10s$, $n = 48$, $m = 2^6 = 64$, $h = 1,000,000$, and $e = 10$ min.

(i) Throughput: by multiplying the intrashard through put and the improving factor of n, the entire network throughput of Elastico can be calculated by 800 n = 38.4 ktps.

(ii) Storage: Elastico does not consider the ledger pruning scheme. All validators are required to store the global ledger, which involves all blocks from different shards and consumes a huge amount of storage space. The storage can be calculated by $nh|\mathscr{B}| \simeq 87$ TB.

5.1.3. OmniLedger. Referring to Ref. [28], we simply take the intrashard throughput in OmniLedger as 1000 tps and set $|\mathscr{B}| = 32$ MB, $n = 48$, $m = 1024$, $h = 1,000,000$, and $e = 24$ hours. Thus, $\mathscr{T} = |\mathscr{B}|/(1000|\text{Tx}|) = 134$ s.

(i) Throughput: by multiplying the intrashard throughput and the improving factor of $n/2$, the entire network throughput of OmniLedger can be calculated by $(1000n)/2 = 24$ ktps.

(ii) Storage: the identity chain and local pruned chain in each shard cover the storage space in OmniLedger. For the identity chain, the block height is calculated by $(h\mathscr{T})/e = 1551$; thus, $\mathscr{B}_{\text{ID},1551} = 1551\,\text{nm}|\text{ID}| = 2.27$ GB. On the other hand, an MPT with the aggregated $\mathscr{B}s$ at the epoch of e_k can be constructed to achieve the ledger pruning, and the state block e_{k+1} is finalized at the end of e_k. Validators only need to store the header of each state block, along with $\mathscr{B}s$ at each epoch. The storage can be calculated by $h|H| + (|\mathscr{B}|e)/\mathscr{T} \simeq 20.6$ GB.

5.1.4. RapidChain. Referring to Ref. [27], we simply take the intrashard throughput in RapidChain as 1000 tps and set $|\mathscr{B}| = 8$ MB, $n = 256$, $m = 256$, $h = 1,000,000$, and $e = 24$ hour. Thus, $\mathscr{T} = |\mathscr{B}|/(1000|\text{Tx}|) = 33.6$ s.

(i) Throughput: by multiplying the intrashard throughput and the improving factor of $n/2$, the entire network throughput of RapidChain can be calculated by $(1000n)/2 = 128$ ktps.

(ii) Storage: the identity chain in the local routing table and local pruned chain in each shard cover the storage space in RapidChain. The routing table records the identities of all other members in the current committee, as well as $\log_2 \log_2 n$ validators of other $\log_2 n$ committees, i.e., $|\text{ID}|m + |\text{ID}|\log_2(\log_2 n)\log_2 n = 9$KB.

5.1.5. Ethereum 2.0. Referring to Ref. [5], we take the number of validators selected in each round (one epoch e contains several rounds) as 10 and set $|\mathscr{B}| = 1.2$ MB, $\mathscr{T} = 10s$, $n = 512$, $m = 8$, $h = 1,000,000$, and $e = 1$ week. In addition, the total number of validators in Shasper is 400 and the interval for checkpoint is 100 s.

(i) Throughput: the intrashard throughput can be calculated by $|\mathscr{B}|/(|\mathrm{Tx}|\mathscr{T}) = 503$ tps. By multiplying the intrashard throughput and the improving factor of $n/3$, the entire network throughput of Ethereum 2.0 can be calculated by $(787n)/3 = 86$ ktps.

(ii) Storage: the PoW-based main chain, the beacon chain, and the local shard chain cover the storage space in Ethereum 2.0. For the main chain, the data involved in Txs will be gradually transferred to the beacon chain for security and storage. The storage space of the current main chain has reached TB level. For the beacon chain, $H_c s$ needs to be stored in each $\mathscr{B}$ for all involved shards, and it can be calculated by $nh|H_c| = 238$ GB. On the other hand, the identity storage space of all active validators in the beacon chain can be calculated by $|\mathrm{ID}|\mathrm{nm} = 128$ KB. Furthermore, validators are required to store the current shard ledger, and the storage can be calculated by $h|\mathscr{B}_c| = 1.14$ TB.

5.1.6. TEEChain. TEE is applied to design fault-tolerant, scalable consensus protocol and an effective sharding protocol in TEEChain. Refer to Ref. [26], and we simply take the intrashard throughput in TEEChain as 2000 tps and set $|\mathscr{B}| = 2$ MB, $n = 16$, $m = 32$, $h = 1,000,000$, and $e = 24$ hour. Thus, $\mathscr{T} = |\mathscr{B}|/(2000|Tx|) = 4.2$ s.

(i) Throughput: by multiplying the intrashard throughput and the improving factor of $n/2$, the entire network throughput of TEEChain can be calculated by $(2000n)/2 = 16$ ktps.

(ii) Storage: TEEChain does not consider the disk storage.

5.2. Comparison and Discussion. According to the above calculation, this study provides a comprehensive comparison and the results are shown in Table 3.

In summary, Monoxide finds a new way of Nakamoto-based consensus and significantly improves the throughput. On the other hand, RapidChain and Ethereum 2.0 outperform Elastico and OmniLedger in transaction throughput and overhead. Furthermore, there is still much room for improvement in the performance of sharding contracts in Chainspace. TEEChain creatively introduces TEE to trade-off security and performance.

Concretely, we focus on four aspects: protocol settings, intrashard consensus, cross-shard transaction atomicity, and security and performance. The protocol settings show how the protocols are set up in an overall perspective, such as network model, randomness generation, participant allocation and reconfiguration, and transaction structure. The intrashard consensus and cross-shard transaction have already been described in the previous sections. Finally, we compare their security aspects and the achieved performance, such as communication and storage overhead.

5.2.1. Protocol Settings. The network model shows the synchrony of the underlying communication network. Except for RapidChain, all the other considered sharding mechanisms adopt the partial sync network model. For most solutions, a randomness is generated at different epochs, which is used to complete different stages of work, such as PoW puzzle for next epoch, shard formation, leader selection, decentralized bootstrapping, and epoch reconfiguration. In addition, there are two main transaction structures: UTXO-based and account-based models. In particular, TEEChain implements a general transaction structure.

5.2.2. Intrashard Consensus. The intrashard consensus protocols have been presented in detail in the previous sections. Considering the waste of resources generated by the PoW consensus, most sharding mechanisms use the BFT protocol to achieve intrashard consensus. Through algorithm optimization, RapidChain improves the fault tolerance from 1/3 to 50%. Chainspace and Ethereum 2.0 combine the BFT-type protocols with the Nakamoto-type protocols to improve scalability. In particular, TEEChain introduces the trusted execution environment (TEE) to enhance the security of PBFT protocol.

5.2.3. Cross-Shard Transaction Atomicity. Cross-shard transaction is supported in all the considered sharding mechanisms except for Elastico. To ensure the atomicity of blockchain data, most solutions adopt a synchronized lock/unlock scheme to achieve either success or failure of transaction results for all transactions. In particular, TEEChain proposes a two-phase lock and two-phase commit atomicity scheme, which outperforms other schemes.

5.2.4. Security and Performance. For security, we focus on the security model, including the threat model and fault tolerance (FT). The threat model represents the likelihood that the system will be suppressed by a malicious node. The attacker behaves arbitrarily, and most attackers are not cooperating with the system. In addition, the attacker is considered slowly adaptive so that the system is not affected in an epoch. Due to the BFT-type consensus algorithm during the intrashard consensus, the FT generally controls at 33%, through the optimization of security algorithms, and the introduction of a TEE can improve FT to a certain extent.

5.3. Different from Other Surveys. As far as we know, the current review literature [15, 30, 37] on blockchain sharding does not cover the same issues as we do.

Kim et al. [30] discussed the blockchain scalability solutions from four dimensions, including on-chain, off-chain, side chain/child chain, and interchain. However, strictly speaking, these solutions can be divided into on-chain and off-chain solutions. Neither the side chain/child chain nor the interchain solutions have changed the main chain. Furthermore, in view of different solutions, the paper did not propose a more profound theoretical analysis and ignores the more important sharding-based scalability solution. Our work considers the most promising shard-based blockchain scalability solution and provides a three-layer taxonomy and a more detailed comparison of the solutions.

Wang et al. [15] emphasized the scalability issues of the current blockchain and put forward the challenges of introducing sharding to blockchain. In view of these challenges, the existing blockchain solutions were analyzed. In particular, they compared five sharding-based blockchain mechanisms. However, they did not provide enough details about security analysis and comparison. On the basis of Ref. [15], we classify existing solutions, focus on several of the most distinctive sharding solutions, and provide a more comprehensive comparison based on detailed theoretical analysis and calculations.

Zhou et al. [37] proposed a three-layer architecture to systematically outline the scalability solutions of the blockchain. However, they elaborated on these solutions from a higher level, without showing the details about the specific mechanisms, and there were no analysis and comparison of security and performance among different solutions. In addition, they did not detail shard-based solutions. Instead, we focus on sharding solutions and present seven well-known shard-based blockchain mechanisms. Furthermore, we provide a comprehensive analysis (i.e., latency and throughput) of considered sharding mechanisms.

6. Conclusions

This study introduces the blockchain logical architecture and trilemma, outlines the knowledge of sharding, emphasizes the importance of sharding for scalable blockchain design, and systematizes the existing state-of-the-art sharding mechanisms, involving intrashard consensus protocol, cross-shard transaction atomicity, and general improvements. On this basis, we also propose the detailed calculations and unique insights analyzing the features and restrictions of the considered sharding mechanisms from multidimensions and make a comprehensive comparison and evaluation [19, 20].

Data Availability

No data were used to support the findings of this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Key Technologies R&D Program of China under Grant no. 2018YFB1402700, the National Natural Science Foundation of China under Grant nos. 6210 070 494 and 61 873 069, and the China Postdoctoral Science Foundation under Grant no. 2021T140074.

References

[1] S. Nakamoto, "Bitcoin: a peer-to-peer electronic cash system," *Decentralized Business Review*, Article ID 21260, 2008.

[2] O. Novo, "Blockchain meets iot: an architecture for scalable access management in iot," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1184–1195, 2018.

[3] R. Yang, F. R. Yu, P. Si, Z. Yang, and Y. Zhang, "Integrated blockchain and edge computing systems: a survey, some research issues and challenges," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1508–1532, 2019.

[4] X. Wang, X. Zha, W. Ni et al., "Survey on blockchain for internet of things," *Computer Communications*, vol. 136, pp. 10–29, 2019.

[5] G. Wood, "Ethereum: a secure decentralised generalised transaction ledger," *Ethereum Project YELLOW paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[6] B. Xu, D. Luthra, Z. Cole, and N. Blakely, "Eos: an architectural, performance, and economic analysis," vol. 11, p. 2019, 2018, https://whiteblock.io/wp-content/uploads/2019/07/eostest-report.pdf.

[7] A. Hafid, A. S. Hafid, and M. Samih, "Scaling blockchains: a comprehensive survey," *IEEE Access*, vol. 8, pp. 125244–125262, 2020.

[8] A. Kuzmanovic, "Net neutrality," *Communications of the ACM*, vol. 62, no. 5, pp. 50–55, 2019.

[9] U. Klarman, S. Basu, A. Kuzmanovic, and E. G. Sirer, "Bloxroute: a scalable trustless blockchain distribution network whitepaper," *IEEE Internet of Things Journal*, 2018.

[10] A. Singh, R. M. Parizi, M. Han, A. Dehghantanha, H. Karimipour, and K.-K. R. Choo, "Public blockchains scalability: an examination of sharding and segregated witness," *Blockchain Cybersecurity, Trust and Privacy. Advances in Information Security*, Springer, vol. 79, Cham, Switzerland, , 2020.

[11] S. Popov, "The tangle," *White paper*, vol. 1, no. 3, 2018.

[12] A. Churyumov and Byteball, "A decentralized system for storage and transfer of value," 2016, https://byteball.org/Byteball.pdf.

[13] Y. Sompolinsky, Y. Lewenberg, and A. Zohar, "Spectre: a fast and scalable cryptocurrency protocol," *IACR Cryptol. ePrint Arch.*vol. 2016, no. 1159, 2016.

[14] 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*, pp. 17–30, Vienna, Austria, October 2016.

[15] G. Wang, Z. J. Shi, M. Nixon, S. Han, and Sok, "Sharding on blockchain," in *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, pp. 41–61, Zurich, Switzerland, October 2019.

[16] G. Yu, X. Wang, K. Yu, W. Ni, J. A. Zhang, and R. P. Liu, "Survey: sharding in blockchains," *IEEE Access*, vol. 8, pp. 14155–14181, 2020.

[17] G. Yu, X. Wang, K. Yu, W. Ni, J. A. Zhang, and R. Liu, *Scaling-out Blockchains with Sharding: An Extensive Survey*, Institution of Engineering and Technology (IET), London, UK, 2020.

[18] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: architecture, consensus, and future trends," in *Proceedings of the 2017 IEEE International Congress on Big Data (BigData Congress)*, pp. 557–564, IEEE, Honolulu, HI, USA, June 2017.

[19] D. Mingxiao, M. Xiaofeng, Z. Zhe, W. Xiangwei, and C. Qijun, "A review on consensus algorithm of blockchain," in *Proceedings of the 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 2567–2572, IEEE, Banff, Canada, October 2017.

[20] S. Kudva, S. Badsha, S. Sengupta, I. Khalil, and A. Zomaya, "Towards secure and practical consensus for blockchain based vanet," *Information Sciences*, vol. 545, pp. 170–187, 2021.

[21] S. Dziembowski, S. Faust, and K. Hostáková, "General state channel networks," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 949–966, Toronto, Canada, October 2018.

[22] X. Xu, I. Weber, M. Staples et al., "A taxonomy of blockchain-based systems for architecture design," in *Proceedings of the 2017 IEEE International Conference on Software Architecture (ICSA)*, pp. 243–252, IEEE, Gothenburg, Sweden, April 2017.

[23] M. Herlihy, "Atomic cross-chain swaps," in *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, pp. 245–254, Egham, UK, July 2018.

[24] J. Poon and T. Dryja, "The bitcoin lightning network: scalable off-chain instant payments," 2016, https://lightning.network/lightning-network-paper.pdf.

[25] S. S. Chow, Z. Lai, C. Liu, E. Lo, and Y. Zhao, "Sharding Blockchain," in *Proceedings of the 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, p. 1665, IEEE, Halifax, Canada, July-August 2018.

[26] 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*, pp. 123–140, Amsterdam Netherlands, June-July 2019.

[27] M. Zamani, M. Movahedi, M. Raykova, and Rapidchain, "Scaling blockchain via full sharding," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 931–948, Toronto, Canada, October 2018.

[28] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "Omniledger: a secure, scale-out, decentralized ledger via sharding," in *Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP)*, pp. 583–598, IEEE, San Francisco, CA, USA, May 2018.

[29] W. Gao, W. G. Hatcher, and W. Yu, "A Survey of Blockchain: Techniques, Applications, and Challenges," in *Proceedings of the 2018 27th International Conference on Computer Communication and Networks (ICCCN)*, pp. 1–11, IEEE, Hangzhou, China, July-August 2018.

[30] S. Kim, Y. Kwon, and S. Cho, "A Survey of Scalability Solutions on Blockchain," in *Proceedings of the 2018 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 1204–1207, IEEE, Jeju Island, South Korea, October 2018.

[31] L. S. Sankar, M. Sindhu, and M. Sethumadhavan, "Survey of consensus protocols on blockchain applications," in *Proceedings of the 2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pp. 1–5, IEEE, Coimbatore, India, January 2017.

[32] Z. Zheng, S. Xie, H. N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: a survey," *International Journal of Web and Grid Services*, vol. 14, no. 4, pp. 352–375, 2018.

[33] J. Xie, F. R. Yu, T. Huang, R. Xie, J. Liu, and Y. Liu, "A survey on the scalability of blockchain systems," *IEEE Network*, vol. 33, no. 5, pp. 166–173, 2019.

[34] W. Wang, D. T. Hoang, P. Hu et al., "A survey on consensus mechanisms and mining strategy management in blockchain networks," *IEEE Access*, vol. 7, pp. 22328–22370, 2019.

[35] A. Chauhan, O. P. Malviya, M. Verma, and T. S. Mor, "Blockchain and Scalability," in *Proceedings of the 2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pp. 122–128, IEEE, Lisbon, Portugal, July 2018.

[36] P. W. Eklund and R. Beck, "Factors that impact blockchain scalability," in *Proceedings of the 11th International Conference on Management of Digital Ecosystems*, pp. 126–133, Limassol, Cyprus, November 2019.

[37] Q. Zhou, H. Huang, Z. Zheng, and J. Bian, "Solutions to scalability of blockchain: a survey," *IEEE Access*, vol. 8, pp. 16440–16455, 2020.

[38] L. Kan, Y. Wei, A. H. Muhammad, W. Siyuan, L. C. Gao, and H. Kai, "A multiple blockchains architecture on inter-blockchain communication," in *Proceedings of the 2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pp. 139–145, IEEE, Lisbon, Portugal, July 2018.

[39] W. Yang, S. Garg, A. Raza, D. Herbert, and B. Kang, "Blockchain: trends and future," in *Proceedings of the Pacific Rim Knowledge Acquisition Workshop*, pp. 201–210, Springer, Nanjing, China, August 2018.

[40] M. Bez, G. Fornari, and T. Vardanega, "The Scalability Challenge of Ethereum: An Initial Quantitative Analysis," in *Proceedings of the 2019 IEEE International Conference on Service-Oriented System Engineering (Sose)*, pp. 167–176, IEEE, San Francisco East Bay, CA, USA, April 2019.

[41] M. H. Manshaei, M. Jadliwala, A. Maiti, and M. Fooladgar, "A game-theoretic analysis of shard-based permissionless blockchains," *IEEE Access*, vol. 6, pp. 78100–78112, 2018.

[42] Y. Xiao, N. Zhang, W. Lou, and Y. T. Hou, "A survey of distributed consensus protocols for blockchain networks," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 1432–1465, 2020.

[43] J. Wang and H. Wang, "Monoxide: scale out blockchains with asynchronous consensus zones," in *Proceedings of the 16th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI}19)*, pp. 95–112, Boston, MA, USA, February 2019.

[44] P. Singhal and S. Masih, "Metaanalysis of methods for scaling blockchain technology for automotive uses," 2019, https://arxiv.org/abs/1907.02602.

[45] J. C. Corbett, P. Hochschild, W. Hsieh et al., "Spanner," *ACM Transactions on Computer Systems*, vol. 31, no. 3, pp. 1–22, 2013.

[46] G. Danezis and S. Meiklejohn, "Centrally banked cryptocurrencies," 2015, https://arxiv.org/abs/1505.06895.

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

[48] M. Vukolić, "The quest for scalable blockchain fabric: proof-of-work vs. BFT replication," in *Proceedings of the International Workshop on Open Problems in Network Security*, pp. 112–125, Springer, Zurich, Switzerland, October 2015.

[49] F. Saleh, "Blockchain without waste: proof-of-stake," *Review of Financial Studies*, vol. 34, no. 3, pp. 1156–1190, 2021.

[50] M. CastroB. Liskov et al.,et al. "Practical byzantine fault tolerance," *OSDI*, vol. 99, no. 1999, pp. 173–186, 1999.

[51] S. Delgado-Segura, C. Pérez-Sola, G. Navarro-Arribas, and J. Herrera-Joancomartí, "Analysis of the bitcoin UTXO set," in *Proceedings of the International Conference on Financial Cryptography and Data Security*, pp. 78–91, Springer, Nieuwpoort, Curaçao, February-March 2018.

[52] J. Zahnentferner, "Chimeric ledgers: translating and unifying UTXO-based and account-based cryptocurrencies," *IACR Cryptology ePrint Archive*, vol. 2018, p. 262, 2018.
[53] J. Gray et al.,et al. "The transaction concept: virtues and limitations," *VLDB*, vol. 81, pp. 144–154, 1981.
[54] T. Haerder and A. Reuter, "Principles of transaction-oriented database recovery," *ACM Computing Surveys*, vol. 15, no. 4, pp. 287–317, 1983.
[55] M. Al-Bassam, A. Sonnino, S. Bano, D. Hrycyszyn, and G. Danezis, "Chainspace: a sharded smart contracts platform," 2017, https://arxiv.org/abs/1708.03778.
[56] V. Buterin, "Ethereum: platform review," *Opportunities and Challenges for Private and Consortium Blockchains*, 2016.
[57] 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 *Proceedings of the 25th {USENIX} security symposium ({USENIX} security 16)*, pp. 279–296, Austin, TX, USA, August 2016.
[58] A. Judmayer, A. Zamyatin, N. Stifter, A. G. Voyiatzis, and E. Weippl, "Merged mining: curse or cure?" in *Proceedings of the Data Privacy Management, Cryptocurrencies and Blockchain Technology*, pp. 316–333, Springer, Oslo, Norway, September 2017.
[59] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, and K.-L. Tan, "Blockbench: a framework for analyzing private blockchains," in *Proceedings of the 2017 ACM International Conference on Management of Data*, pp. 1085–1100, Chicago, IL, USA, May 2017.
[60] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, N. Zeldovich, and Algorand, "Scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th Symposium on Operating Systems Principles*, pp. 51–68, Shanghai, China, October 2017.
[61] E. Syta, P. Jovanovic, E. K. Kogias et al., "Scalable bias-resistant distributed randomness," in *Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP)*, pp. 444–460, IEEE, San Jose, CA, USA, May 2017.
[62] M. Stadler, "Publicly verifiable secret sharing," in *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 190–199, Springer, Saragossa, Spai, May 1996.
[63] C. P. Schnorr, "Efficient signature generation by smart cards," *Journal of Cryptology*, vol. 4, no. 3, pp. 161–174, 1991.
[64] P. Feldman, "A practical scheme for non-interactive verifiable secret sharing," in *Proceedings of the 28th Annual Symposium on Foundations of Computer Science (Sfcs 1987)*, pp. 427–438, IEEE, Los Angeles, CA, USA, October 1987.
[65] I. Abraham, S. Devadas, D. Dolev, K. Nayak, and L. Ren, "Efficient synchronous byzantine consensus," 2017, https://arxiv.org/abs/1704.02397.
[66] J. Sousa and A. Bessani, "From byzantine consensus to BFT state machine replication: a latency-optimal transformation," in *Proceedings of the 2012 Ninth European Dependable Computing Conference*, pp. 37–48, IEEE, Sibiu, Romania, May 2012.
[67] Y. Wang, "Blockchain Bft protocol for complete asynchronous networks," 2020, https://arxiv.org/abs/2005.04309.
[68] Z. Jia, R. Chen, and J. Li, "Delottery: a novel decentralized lottery system based on blockchain technology," in *Proceedings of the 2019 2nd International Conference on Blockchain Technology and Applications*, pp. 20–25, Xi'an China, December 2019.
[69] D. Boneh, J. Bonneau, B. Bünz, and B. Fisch, "Verifiable delay functions," in *Proceedings of the Annual international Cryptology Conference*, pp. 757–788, Springer, Santa Barbara, CA, USA, August 2018.
[70] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, "Eclipse attacks on bitcoin's peer-to-peer network," in *Proceedings of the 24th {USENIX}Security Symposium ({USENIX} Security 15)*, pp. 129–144, Washington, DC, USA, August 2015.
[71] N. Alon, H. Kaplan, M. Krivelevich, D. Malkhi, and J. Stern, "Scalable secure storage when half the system is faulty," in *Proceedings of the International Colloquium on Automata, Languages, and Programming*, pp. 576–587, Springer, Geneva, Switzerland, July 2000.
[72] N. Alon, H. Kaplan, M. Krivelevich, and D. Malkhi, "Addendum to scalable secure storage when half the system is faulty," *Information and Computation*, vol. 205, no. 7, pp. 1114–1116, 2007.
[73] P. Maymounkov, D. Mazières, and Kademlia, "Kademlia: a peer-to-peer information system based on the XOR metric," in *Proceedings of the International Workshop on Peer-To-Peer Systems*, pp. 53–65, Springer, Cambridge, MA, USA, March 2002.
[74] B. Wesolowski, "Efficient verifiable delay functions," in *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 379–407, Springer, Vienna, Austria, October 2019.
[75] B. Awerbuch and C. Scheideler, "Towards a scalable and robust DHT," *Theory of Computing Systems*, vol. 45, no. 2, pp. 234–260, 2009.
[76] S. Sen and M. J. Freedman, "Commensal cuckoo," *ACM SIGOPS Operating Systems Review*, vol. 46, no. 1, pp. 33–39, 2012.
[77] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 3–16, Vienna, Austria, October 2016.
[78] M. Herlihy, "Blockchains from a distributed computing perspective," *Communications of the ACM*, vol. 62, no. 2, pp. 78–85, 2019.