IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 13, 2019, accepted December 29, 2019, date of publication January 9, 2020, date of current version January 24, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2965147

Survey: Sharding in Blockchains

GUANGSHENG YU[1,2], XU WANG[1,2], KAN YU[3], WEI NI[4], (Senior Member, IEEE), J. ANDREW ZHANG[1], (Senior Member, IEEE), AND REN PING LIU[1,2], (Senior Member, IEEE)

[1]Global Big Data Technologies Centre, University of Technology Sydney, Ultimo, NSW 2007, Australia
[2]Food Agility CRC Ltd., Ultimo, NSW 2007, Australia
[3]Department of Computer Science and Information Technology, La Trobe University, Melbourne, VIC 3086, Australia
[4]Data61, CSIRO, Marsfield, NSW 2122, Australia

Corresponding author: Guangsheng Yu (guangsheng.yu@uts.edu.au)

This work was supported in part by the Food Agility CRC Ltd., through the Commonwealth Government CRC Program, and in part by the UCOT Australia Pty Ltd.

ABSTRACT The Blockchain technology, featured with its decentralized tamper-resistance based on a Peer-to-Peer network, has been widely applied in financial applications, and even further been extended to industrial applications. However, the weak scalability of traditional Blockchain technology severely affects the wide adoption due to the well-known trillema of decentralization-security-scalability in Blockchains. In regards to this issue, a number of solutions have been proposed, targeting to boost the scalability while preserving the decentralization and security. They range from modifying the on-chain data structure and consensus algorithms to adding the off-chain technologies. Therein, one of the most practical methods to achieve horizontal scalability along with the increasing network size is sharding, by partitioning network into multiple shards so that the overhead of duplicating communication, storage, and computation in each full node can be avoided. This paper presents a survey focusing on sharding in Blockchains in a systematic and comprehensive way. We provide detailed comparison and quantitative evaluation of major sharding mechanisms, along with our insights analyzing the features and restrictions of the existing solutions. We also provide theoretical upper-bound of the throughput for each considered sharding mechanism. The remaining challenges and future research directions are also reviewed.

INDEX TERMS Blockchain, scalability, throughput, scale-out mechanism, sharding, survey.

I. INTRODUCTION

Working as distributed, incorruptible, and tamper-resistant ledgers, Blockchain technology has shown its great potential to tackle critical security and trust challenges in various applications, e.g., cryptocurrency, Internet-of-Things, and edge computing [1]–[3]. Running over a peer-to-peer network, Blockchain processes application requests in the form of Blockchain transactions [4]. The transactions are mined into blocks by Blockchain miners following consensus protocols, e.g., Proof-of-Work (PoW) for permissionless Blockchains and the Practical Byzantine Fault Tolerance (PBFT) for permissioned Blockchains [5], and the blocks are chained with their hash values [1].

The throughput of a Blockchain system, defined as the number of processed transactions per second of the Blockchain, is far from practical requirements and has become a crucial limitation stopping Blockchain from being widely adopted [6]. For example, Bitcoin can only handle up to approximately 10 transactions per second with its maximum block size of 1MB and average 10 minutes block period [7], which severely hinders the use of Blockchains in the high-frequency trading. To handle a great number of transactions, Blockchain has been considered as a secure base-layer (or a settlement center for cryptocurrencies) where transactions are processed off-chain and then settled in the Blockchain. For example, Lightning network and Raiden network (referring to the state-channel technology) support off-chain payments and broadcast a summary of a batch of off-chain payments to the Blockchain [8], [9]. Plasma (referring to the sidechain technology) builds various applications on the top of Ethereum [10]. These methods, known as the Layer-2 scaling, minimize the interaction with the Blockchain to reduce the latency from the users' perspective but do not improve the throughput of Blockchains [11].

The associate editor coordinating the review of this manuscript and approving it for publication was Nicola Andriolli.

In contrast, the Layer-1 scaling is designed for improving the throughput of Blockchains from the systematic perspective. A Blockchain system can be optimized in the following ways to handle a growing amount of work.

- reducing the communication and computation overhead;
- adding resources to a single node, i.e., vertical scaling;
- adding more nodes to the Blockchain, i.e., horizontal scaling [12].

Reducing Overhead: New Blockchain consensus protocols have been developed for high Blockchain throughput by reducing the overhead. For example, every PoW winner (i.e., a miner) is eligible for several blocks rather than a single block in Bitcoin-NG [13] and its variations [14], [15]. The traditional PBFT consensus protocol has been developed and optimized to reduce the communication overhead and achieve high throughput in large-scale networks [16]–[19]. However, $O(n)$ (n is the number of participating miners) is the lower bound that this type of technologies can reduce the overhead at most, as every participating miners have to exchange and store messages during every consensus round regardless of the route of transactions.

Vertical Scaling: Bitcoin tried to improve throughput by vertical scaling methods. For example, increasing the number of allowed transactions in a single block and/or reducing the block period can improve the throughput of Bitcoin but consume more resources, e.g., storage, computation, and bandwidth, of Bitcoin nodes [20]–[23]. Beyond this, The Greedy Heaviest Observed Subtree (GHOST) [24] is implemented by Ethereum to organize blocks in a tree instead of a chain of blocks and obtain a higher throughput [4]. The GHOST is subsequently extended to the directed acyclic graph (DAG). The DAG is adopted to organize transactions where every transaction contains hash values pointing to existing transactions [25]–[30]. The DAG structure allows transactions to be confirmed in parallel and thus improves the network utilization ratio given the resources of a node, which improves the throughput of the entire distributed system. However, the vertical scaling methods cannot infinitely improve the throughput, as a Blockchain system is designed to run in a decentralized and homogeneous network where the security is closely dependent on the consensus across the entire network. The larger-scale the network is, the more bandwidth is needed to achieve the network synchronization, while the bandwidth is the resource that cannot be indefinitely added [20]. This leads to the vertical scaling being compromised to the throughput of resources-limited nodes.

Horizontal Scaling: Sharding technology, dividing a whole Blockchain into multiple shards and allowing participating nodes to process and store transactions of a few shards (i.e., only parts of the Blockchain), holds the key to horizontal scaling, also known as the *scale-out* technology. By taking advantage of the sharding technology that allows partial transactions processing and storage on a single node, the whole Blockchain can achieve a linearly increasing throughput with the growing number of nodes. This is important for the adoption of Blockchains providing high quantity and quality of services to the public in large-scale networks with infinite growth, which has attracted the interest of researches regarding the improvement of the Blockchain scalability.

A number of studies have proposed new sharding mechanisms. Surveys of Blockchain scalability which used to only focus on **Reducing overhead** and **Vertical scaling** have been gradually taking the sharding technology into account. However, none of them was able to focus on sharding and systematically introduce the challenges of sharding, features and restrictions of the existing solutions, and the future trends.

A. OUR CONTRIBUTIONS

We provide a more systematic introduction of sharding mechanisms than existing surveys and papers. The key contributions are highlighted as follows.

1) Our work, for the first time, provides an introduction of state-of-the-art sharding mechanisms ranged from BFT-based to Nakamoto-based sharding mechanisms, while the latter has never been systematized in any of the existing surveys at the time of writing.
2) We gain our own insights analyzing the features and restrictions into the existing solutions to the intra-consensus-safety, atomicity of cross-shard transactions, and general challenges and improvements proposed by the considered sharding mechanisms.
3) We also provide a calculation to obtain the theoretical upper-bound of throughput for each considered sharding mechanism. Based on the result and the insights of the features and restrictions of each existing sharding solution, a comprehensive comparison is proposed.
4) Finally, we point out the current remaining challenges of sharding mechanisms, followed by suggestions for the future trend of designing reliable sharding mechanisms.

B. RELATED WORK

The relationship between the existing studies and our work is discussed. Note that, all the considered previous studies highlight the trend of scalability in the future of Blockchains, and intend to accommodate the existing solutions to scale Blockchain systems. These solutions include but not limited to upgrading Bitcoin (increasing block size or conducting Segregated Witness), scalable consensus algorithms, state-channels, and multiple sidechains structure.

Previous surveys including [31]–[38] discuss the aforementioned solutions, but involve no information about the sharding which has been realized to be the most practical solution so far for a *scale-out* Blockchain system. Thus, there have been several recent studies presenting their own sharding mechanisms, as well as surveys that manage to summarize them and propose new benchmarks [4], [3], [33], [39]–[51]. However, all of these studies compare the sharding with other kinds of solutions by either presenting a vague introduction of

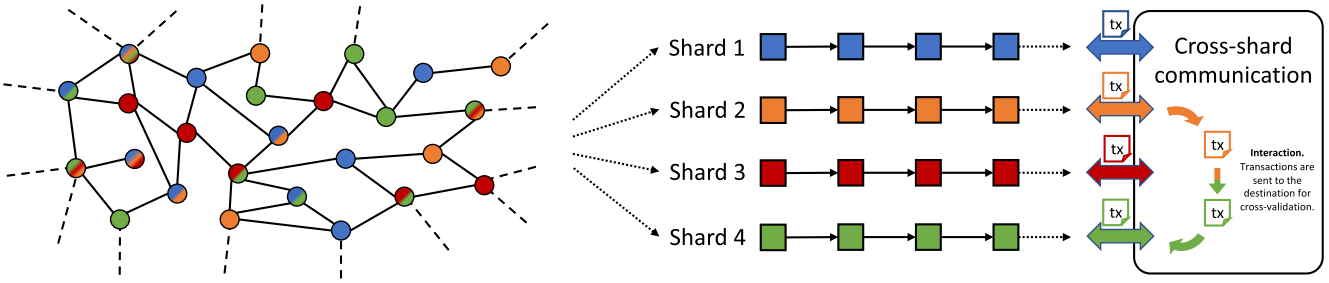

FIGURE 1. The sharding technology partitions the network into different groups, while each of the groups maintains its own ledger and processes and stores a disjoint set of transactions. By implementing a secure cross-shard communication protocol, such disjoint transaction sets that could not have been interacted become securely verifiable and interactively executable in parallel. Note that, nodes in some sharding mechanisms (e.g., Monoxide) can choose to participate in the processing of multiple shards and maintain their ledgers, as illustrated by the multicolored circles, while the unicolored circles denote the nodes only participating in a single shard to which they are assigned in terms of the color.

only one or two sharding mechanisms, or lacking the insights for evaluation, except [3], [39], [48], [49], [51] putting more efforts on introducing sharding. Reference [39] makes use of the scale cube architecture, highlighting that the horizontal scalability should only be improved by partitioning the data and consensus. However, it only provides a vague introduction of Ethereum 2.0, and the same problem exists in [3] where the consensus layer is decoupled from the ledger topology layer (which is inappropriate due to the importance of intra-consensus in a sharding system). Reference [48] presents an analytic model in a game-theoretical way that is designed to benchmark the existing sharding mechanisms, and aim for design guidance for future solutions. However, *sharding can be thought as the "multiple committees" upon the traditional Byzantine-Faulty-Tolerance (BFT)-based consensus*, as stated in [46], [48], has been outdated as [52] proposes a Nakamoto-based sharding mechanism (Monoxide). A unified comparison between such Nakamoto-based sharding mechanisms and the BFT-based sharding mechanisms is also absent in [49] and the most closely related survey [51] (where the BFT-based sharding mechanisms are focused, as well as the corresponding randomness generators).

To the best of our knowledge, our work outweighs all the existing surveys in a more systematic way, in regards to the key concept of various sharding mechanisms, and a comprehensive comparison for practitioners based on our insights.

C. PAPER OUTLINE

The rest of the paper is organized as follows. Section II briefly presents an overview of sharding technology and introduces the survey methodology. Section III presents an introduction of the considered sharding mechanisms, upon which the comparison and discussion are presented in Section IV. Section V concludes the survey.

II. SHARDING REVIEW AND SURVEY METHODOLOGY

A. OVERVIEW OF THE SHARDING TECHNOLOGY

Sharding is first proposed by [53] and commonly used in distributed databases and cloud infrastructure. Based on the pioneering proposals [54], [55] integrating sharding with permissioned and permissionless Blockchain, respectively, the sharding technology is thought to be able to partition the network into different groups (shards), so that the compulsory duplication of three resources (i.e., the communication, data storage, and computation overhead) can be avoided for each participating node, while these overheads must be incurred by all full nodes in traditional non-sharded-Blockchains. This partition is essential because the restriction incurred by the three resources owned by a single node may make the system unable to take full advantage of a scalable consensus algorithm. Sharding is so far one of the most practical solutions to achieve a *scale-out* system where the processing, storage, and computing can be conducted in parallel, as illustrated in Fig. 1. As such, the capacity and throughput being linearly proportional to the number of participating nodes or the number of shards become possible, while preserving decentralization and security. However, sharding poses new challenges to Blockchains, i.e., the *intra-consensus-safety*, *cross-shard-atomicity*, and the *general improvements*

TABLE 1. Notation definition.

Notation	Definition
$\lvert\cdot\rvert$	Size of the items
$\mathcal{C}_h$	Blockchain with a block height of h within a single shard
$\widehat{\mathcal{C}_h}$	Headerchain with a block height of h within a single shard
$\mathcal{B}$	Block, including $\mathcal{H}$, Txs, and Sigs. Note that, $\lvert\mathcal{H}\rvert$ and $\lvert Sigs\rvert$ are negligible for $\lvert\mathcal{B}\rvert$.
$\mathcal{H}$	Block header
Tx	Transaction
Sig	Signature
$\mathbf{T}$	Block period
$\mathcal{E}_k$	k-th epoch
$\mathbf{E}$	Epoch length
n	The number of shards
m	Size of each shard
h	Expected block height of chains among all the shards

regarding the storage, latency, etc, where the detail is our concentration and is described starting from Section III.

There have been a few studies working on these challenges regarding the sharding in permissionless Blockchains [52], [55]–[59], prior to which [54] proposes a sharded permissioned Blockchain that will not be discussed in this survey due to its forfeit of permissionless decentralization. Rather, the sharding in permissionless Blockchains is focused.

B. SURVEY METHODOLOGY

This survey focuses on sharding in permissionless Blockchains (as permissioned Blockchains do not take full advantage of the sharding technology due to the smaller network size and its forfeit of permissionless decentralization), and is based on the published research papers and other research references of Monoxide [52], Elastico [55], OmniLedger [56], Rapidchain [57], Chainspace [58], and Ethereum 2.0 [59]. Our methodology can be characterized as follows.

1) We clarify the demand for high scalability in Section I, based on the well-known trillema of decentralization-security-scalability in Blockchains. We discuss the potential solutions ranged from the Layer-1 scaling (on-chain scaling) to Layer-2 scaling (off-chain scaling), with the former being focused in order to address the throughput issue. Upon this, we elaborate on the importance of the *scale-out* technology of Layer-1 scaling, i.e., sharding, which is thought to be orthogonal to any other scalable technologies, and so far the most practical solution to achieve horizontal scalability in large-scale Blockchain networks.
2) We summarize six of the most well-known and typical sharding mechanisms in large-scale permissionless Blockchains, i.e., Monoxide, Elastico, OmniLedger, Rapidchain, Chainspace, and Ethereum 2.0, which are characterized in *intra-consensus-safety*, *cross-shard-atomicity*, and *general improvements*, respectively presented in Section III-A, Section III-B and Section III-C.
3) Based on the previous description of the considered sharding mechanisms, we provide our own insights in regards to each of the features, 1) what issues in a sharding system the features have addressed; and 2) the restrictions of these features. Besides, we provide a comparison, based on the insights and our calculation, as shown in Section IV-A, among the considered sharding mechanisms. Finally the result is characterized in Tables 2 and 3.

III. DESCRIPTION

As a Layer-1 solution to the scalabilty issue of Blockchain systems, and the most practical solution to push Blockchain systems to *scale-out* in terms of communication bandwidth, disk storage, and computation (i.e., full-sharded), there are two significant issues each sharding mechanism needs to resolve.

Intra-Consensus-Safety: how to secure the consensus algorithm inside a shard away from both the Nakamoto-based and BFT-based 1% attack [59] in a scalable way, while the latter can also be corresponding to a secure randomness generation process, as discussed in Section III-A; note that 1% attack is an attack strategy in sharded networks where attackers can dominate a single shard more easily than dominating the whole network;

Cross-Shard-Atomicity: how to support the cross-verification, and guarantee the *Atomicity* [60], [61] of cross-shard transactions for both unconditional transactions (simple payment) and conditional contract-oriented transactions in an efficient way (inefficient if the latency and overhead for achieving atomic-safe cross-shard transactions are higher than $O(n)$; n denotes the number of shards being partitioned or the number of participating nodes), as discussed in Section III-B;

General Improvements: based on the *intra-consensus-safety* and *cross-shard-atomicity*, we focus on the improving factor $\mathcal{N}$ regarding the multiple of optimized global throughput for each considered sharding mechanism, while $\mathcal{N}$ is subject to the linear order $O(n)$. On the other hand, the additional latency and overhead originated from the proposed solutions also reveal the new problems that sharding brings to us. In regard to this, some *general improvements* are discussed in Section III-C.

A. INTRA-CONSENSUS PROTOCOL

Sharding significantly increases the throughput in $O(n)$, but sacrificing security in intra-consensus protocols, i.e., the per-zone security or 1% attack [52], [59]. Concretely, it is categorized into the Nakamoto-based 1% attack and BFT-based 1% attack.

The total amount of mining power among the network, i.e., $\mathbb{P}$, guarantees the low probability for a single entity to dominate over 50% mining power. By purposely dividing the network into n partitions (shards), we can greatly increase the throughput in $O(n)$, where rational miners tend to ideally distribute their mining power in multiple shards (at most n shards) in order for the maximum rewards. However, this also decreases the security of PoW in each shard in $O(1/n)$. Such a system can be more prone to double-spend attack by a malicious miner that only needs to own the mining power $\mathcal{P} > \mathbb{P}/n \times 50\%$ due to the smaller shard size compared to the entire network size. This issue deteriorates as n increases in order for a larger throughput, which becomes the most serious barrier to PoW being implemented for the intra-consensus protocol of a sharding mechanism.

On the other hand, BFT-based consensus algorithms are considered instead of PoW in order to solve the security challenge, as discussed above. However, such designs introduce another kind of vulnerabilities other than that of the PoW-based one, as discussed in the following.

- It is of importance to carefully design a scheme to generate an unpredictable and unbiasable randomness without any third-parties in permissionless Blockchains.

TABLE 2. A comparison regarding the protocols (ranged from the settings of intra-consensus to the design of cross-shard atomicity, as well as the corresponding overhead) among the discussed sharding mechanisms in this paper is elaborated.

			Monoxide	Elastico	OmniLedger	RapidChain		Ethereum 2.0	Chainspace
Network model			Partial-sync	Partial-sync	Partial-sync	Intra	Sync	Partial-sync	Partial-sync
						Total	Partial-sync		
Security model	Threat model		Attackers behave arbitrarily, Uncoordinated majority	Attackers behave arbitrarily, slowly adaptive	Attackers behave arbitrarily, slowly adaptive	Attackers behave arbitrarily, slowly adaptive		Attackers behave arbitrarily, Uncoordinated majority	Attackers behave arbitrarily, Uncoordinated majority
	FT	Intra	50%	33%	33%	50%		33%	33%
		Total	50%	25%	25%	33%		33%	25%
Intra-Consensus Protocol			PoW-based Chu-ko-nu mining	PBFT	ByzCoinX	50% BFT		BFT-based PoS	MOD-SMART implementation of PBFT
Randomness ($\mathcal{R}$)	Existence		No	Yes. $\mathcal{R}_{i+1}$ is generated by the final committee at the end of epoch i	Yes. The $\mathcal{R}_{i+1}$ is generated by using RandHound + VRF in the beginning of epoch i+1	Yes. $\mathcal{R}_{i+1}$ is generated by the reference committee at the end of epoch i		Yes. Each $\mathcal{R}$ is generated by using RANDAO + VDF on the beacon chain	Unknown
	Use		N/A	1. The seed of PoW puzzle for the next epoch; 2. Select the leader during intra-consensus	1. Select the leader and the sub-group allocation during intra-consensus; 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 intra-consensus; 3. Decentralized bootstrapping; 4. Epoch reconfiguration		1. Select the proposer of each shard; 2. Select the attesters for each shard; 3. Select the validators responsible for checkpointing from the global pool.	Unknown
Members	Allocation		One-off allocation based on the identity (address) of nodes	Allocation based on the least-significant bits of the result of PoW puzzles	Allocation based on $\mathcal{R}$	Allocation based on the result of PoW puzzles		Allocation based on $\mathcal{R}$	One-off allocation based on objects
	Safe Epoch reconfiguration		N/A	Unsafe	Yes, swapping-out bounded by 2/3 at a given time	Yes, swapping-out a constant number of node		Yes	N/A
Additional global Blockchain			Mixed targets: No Identical targets: Yes	Yes, a global ledger	Yes, identity Blockchain	Yes, reference Blockchain		Yes, the mainnet and beacon chain	No
Transaction structure			Account	UTXO	UTXO	UTXO		Account	Object-driven, contract-sharded
Cross-shard Tx	Support		Yes	No	Yes	Yes		Yes	Yes
	Method		Async, Lock-free	N/A	Sync, Lock/Unlock	Sync, Lock/Unlock		Sync, Lock/Unlock	Sync, Lock/Unlock
Complexity	Communication		Mixed PoW targets: $O(m + n\log_2 n)$ Identical PoW targets: $O(m + n)$	$O(m^2 + n)$	$O(\log_2 m + n)$	$O(m^2 + m\log_2 n)$		$O(m^2 + n)$	$O(m^2 + n)$
	Storage		$\Omega(\lvert C\rvert) \sim O(\lvert C\rvert + n\lvert C_h\rvert)$	$O(n\lvert C\rvert)$	$O(\lvert C\rvert)$	$O(\lvert C\rvert + \lvert C_r\rvert)$		$\Omega(\lvert C\rvert + n\lvert H\rvert + \lvert C_g\rvert) \sim O(n\lvert C\rvert + \lvert C_g\rvert)$	$O(\lvert C\rvert + \lvert C_{nh}\rvert)$
Features and Restrictions			Insight 1, Insight 9, Insight 15	Insight 2, Insight 3, Insight 15	Insight 4, Insight 5, Insight 10, Insight 11, Insight 14, Insight 16	Insight 6, Insight 12		Insight 7, Insight 8, Insight 13, Insight 15	Insight 2, Insight 11, Insight 17

The randomness can be used to 1) allocate validators (an alias for nodes participating in the intra-consensus process in the context of BFT-based systems) into different shards at the beginning phase and every reconfiguration phase; 2) select the leader of each shard; and 3) decide which shards a cross-shard transaction should broadcast to, etc. Without such a strictly-chosen randomness, malicious validators may be able to bias the allocation and control the elections at will, such as collusion within a shard (with a small number of validators due to the weak scalability of traditional BFT-based consensus algorithms [62], e.g., PBFT [5]).

- Then it ends up encountering the dilemma of BFT-based 1% attack that the weak scalability of BFT-based consensus algorithm restricts the shard size, i.e., the number of members in a shard, while too small a size can potentially decrease the security of the intra-consensus with a strict fault-tolerance (FT), as described by the following cumulative binomial distribution,

$$s(k, m, p) = P[X \leq c] = \sum_{k=0}^{c} \binom{m}{k} p^k (1-p)^{m-k},$$
$$f(k, m, p) = 1 - s(k, m, p), \tag{1}$$

where X is the random variable that represents the number of times a malicious miner is picked [13], [55], [56], [63]; m denotes the shard size; c denotes the number of malicious members within a shard; and p denotes the total FT among the entire network. It is strongly suggested that $s(k, m, p)$ should be greater than

TABLE 3. A comparison (regarding the results of throughput and cost calculated in Section IV-A) among the discussed sharding mechanisms in this paper is elaborated. Based on the result, the latency is also obtained and shown. Note that, we consider cloud servers with outbound bandwidth 25MB/s, 4vCPU of Turbo boost, and 1TB basic disk storage space (N/A: not available).

		Monoxide	Elastico	OmniLedger	RapidChain	Ethereum 2.0	Chainspace
Shards' settings	Number of shards (n)	$2^{10}\sim2^{18}$	$<10^2$	$<2^6$	$<2^8$	$<2^9$	$<10^2$
	Shard size (m)	$10^2\sim10^4$	$<10^2$	$2^2\sim2^{10}$	$(2^2-1)\sim2^8$	$<10^2$	$<10^2$
Epoch length		N/A	$\sim$10min	$\geq$one day	$\leq$one day	one week	Exists, details not provided
Latency	Transaction confirmation	23s	<900s	$\sim$100s	70s	6s$\sim$8s [95]	2s
	Epoch reconfiguration	N/A	N/A	1000s	200$\sim$350s	Unknown	Unknown
Upper-bound	Improving factor ($\mathcal{N}$)	n/2	n	1$\sim$n/2	n/2	n/3	1$\sim$n/2
	Throughput	1.23$\sim$2.56Mtps	48ktps	28.8ktps	128ktps	134ktps	<400tps
	Cost	30$\sim$80 USD/hour	30$\sim$35 USD/hour	0.2$\sim$0.3 USD/hour	0.2$\sim$0.3 USD/hour	0.4$\sim$0.45 USD/hour	N/A

99% [63], while only $m \gtrsim 144$ can satisfy, of which the traditional BFT-based consensus algorithm cannot be capable[1]. In order to resolve this, highly scalable BFT-based consensus algorithms with large shard size require more attractions.

In this section, we compare and discuss the *intra-consensus* protocols of the considered sharding mechanisms, i.e., Monoxide, Elastico, Chainspace, OmniLedger, RapidChain, and Ethereum 2.0. Note that the Shasper used in Ethereum 2.0 features its novel and engineering-oriented design that combines the two major issues (*intra-consensus-safety* and *cross-shard-atomicity*) and kills two birds with one store. Elastico and Chainspace use PBFT for *intra-consensus* that are not discussed in detail in this section, while the randomness generator of Chainspace is not discussed as the detail is not provided in [58].

Also note that, a threat model where the attackers can refuse to participate or collude others (behave arbitrarily) takes effect in all discussed sharding mechanisms in this survey. Also, Elastico [55], OmniLedger [56], and RapidChain [57] assume the slowly adaptive attackers (who can only succeed to attack in a long time), while Monoxide [52], Ethereum 2.0 [59], and Chainspace [58] assume a model of uncoordinated majority where all participators are game-theoretically rational, i.e., egoism (with an upper-bounded fraction that can coordinate the majority). Therein Chainspace [58] also introduces an audit scheme to prevent attacks from dishonest shards.

[1]A few sharding mechanisms are incurring a total 25% FT based on the 33% FT in each shard, e.g., Elastico, OmniLedger, and Chainspace. This can be a BFT-based 1% attack, by dispersing validators into as many shards as possible to maximize the possibility to control some shards. Elastico and Chainspace suffer from this security issue, while OmniLedger implements a scalable BFT-based consensus algorithm to address this issue.

1) NAKAMOTO-BASED-MONOXIDE - CHU-KO-NU MINING

Monoxide is the first sharding mechanism that eliminates the need for generating randomness, and implements Nakamoto consensus algorithm for its intra-consensus. It introduces a one-off bootstrapping in the beginning, to allocate each node (including miners and non-miners) into different shards based on their identity addresses. By using the proposed Chu-ko-nu mining, Monoxide can achieve a large-scale network with a huge number of shards and a flexible shard size. It involves a Merkle Patricia Tree (MPT) [64] root consisting of all proposed blocks among multiple shards, thus the $\mathbb{P}/n$ can be multiplied by a factor k (k denotes the number of shards a particular miner manage to mine on). Consequently, dispersing mining power can be re-aggregated to solve the 1% attack.

Chu-ko-nu mining is inspired by the merged mining first proposed in [65] and discussed in [66]. Merged mining shares the mining power among a parent chain and multiple auxiliary chains based on the same kind of PoW algorithms being run. As such, those auxiliary chains with relatively smaller mining power can be protected by the total mining power of the parent chain. Likewise, Monoxide shares a similar idea but conducting the mining process across multiple parallel shards without any hierarchy. By involving an MPT root consisting of all proposed blocks among the shards that a specific miner cares about, the effective mining power can be amplified by a factor of k. Defined in [52], the effective mining power differs from the physical mining power, in the sense that the physical mining power is calculated in *hashrate* (the number of hash values that a miner can probe the nonce per second) which directly corresponds to the total mining power $\mathbb{P}$, and the hardware performance (e.g, CPU or GPU), while the effective mining power is indirectly obtained by observing the block period and difficulty. They are expected to be equaled in a non-sharded system, while with Chu-ko-nu mining, the normal block can be replaced by a batch-chaining-block (containing the information of the involved shards, e.g., 1) the identity of each shard; 2) from/to which

shard the proposed block is received/sent; and 3) the MPT proof of the proposed new block of the local shard associated with the given MPT root, etc), so that a one-off physical mining can be done to meet the different (or identical) difficulties associated with its shard. Thus, the similar block periods among the shards contribute to an effective mining power of $\mathbb{P}k/n \simeq \mathbb{P}$ as $k \to n$, hence addressing the 1% attack.

To be specific, the PoW expression for a miner conducting Chu-ko-nu mining is described as (2),

$$\mathbb{H}(\eta \parallel \mathbb{H}(x \parallel MPT_M)) \leq \gamma, \tag{2}$$

where γ denotes the PoW target corresponding to a certain difficulty; $\mathbb{H}$ denotes the hash function; η denotes the *nonce* that fulfills (2); x denotes the header content, including the aforementioned information of the involved shards and the other fields defined in the normal PoW, as well as the inbound and outbound relay transactions in regards to the cross-shard communication (discussed in Section III-B.1); MPT_M denotes the MPT root consisting of all proposed blocks of each involved shard, i.e., $[\mathcal{B}_0, \mathcal{B}_1, ..., \mathcal{B}_{n-1}]$ if $k = n$, where each proposed block excludes its η, and contains its identity and the list of relay transactions.

Thus, the miner can subsequently send the finalized block to its corresponding shard with a satisfied η, as well as a proof,

$$[MPT_M, \eta, \mathcal{B}_i, \pi_i], \tag{3}$$

where π_i denotes the MPT proof of $\mathcal{B}_i$ in the given MPT with a root of MPT_M. Any node can verify $\mathcal{B}_i$ with π_i, and malicious miners have to revert the history in all involved shards, i.e., from 0 to $n-1$ in this case, to double-spend the transactions because of MPT_M being already updated with the change of leaves. Thus, the effective mining power is amplified by a factor of n.

Note that, Chu-ko-nu mining can handle both the mixed and identical PoW targets of shards in one batch.

- In the case of mixed PoW targets, a miner is allowed to finalize blocks and send them to any shards i to j whose PoW targets have been fulfilled by the current given η, with the rest of shards whose targets have yet to be satisfied. After that, the mining process resumes, while MPT_M is updated because of the just finalized blocks from shards i to j.
- In the case of identical PoW targets, a miner can also finalize blocks and send them to all shards regardless of whether the given η fulfills the PoW targets or not (assume the PoW targets are asymptotically equal[2], and there must be some shards accepting its block and some rejecting). In addition to this, a global subnet maintaining and broadcasting headers from all shards where all miners must participate can significantly reduce the communication overhead, by eliminating the need of π_i.

[2]Rational miners tend to mine on as many shards as possible so that the PoW difficulties will be self-adapted to be identical.

Having known these two modes, it is observed that accepting/rejecting a block of a single shard is independent of the decisions from other shards, i.e., asynchronization. Such a feature greatly promotes the throughput of Monoxide in a secure way, and also allows the *cross-shard-atomicity* in Monoxide, i.e., Relay transactions, as discussed in Section III-B.1.

However, in order to meet the requirement of $\mathbb{P}k/n \simeq \mathbb{P}$, Monoxide needs most of miners to conduct Chu-ko-nu mining across as many shards as possible, i.e., $k = n$ in the best case. However, this implies the fact that if miners only mine on k out of n shards, i.e., $\mathbb{P}k/n$, where $k \ll n$, the factor expected to amplify the effective mining power will be too small to secure the mining process, hence reducing the attack cost. On the other hand, rational miners tend to mine on all n shards to reap the maximum profit, which may also result in the power centralization due to the huge cost of bandwidth, disk storage, and computing processors that only the professional mining facilities can afford.

Insight 1: The amplification to the effective mining power relies on an incentive scheme that should encourage miners to mine across $k \to n$ shards in Chu-ko-nu mining. This also poses the issue of power centralization and additional overhead to Monoxide.

2) BFT-BASED-ELASTICO

Using BFT-based algorithms for the intra-consensus is an alternative to bypass the vulnerability of Nakamoto-based algorithm (**Insight 1**). Thus, including but not limited to Elastico, OmniLedger, RapidChain, Chainspace, and Ethereum 2.0 choose to implement BFT-based algorithm. Therein, Elastico uniformly (re) allocates potential validators in terms of the different least-significant bits of the unpredictable PoW solutions at the beginning of each epoch, followed by running PBFT for the intra-consensus. The randomness used during the mining is generated by a proposed distributed commit-and-xor scheme.

Consensus Algorithm - PBFT's Restrictions in Sharding

Due to the weak scalaibilty of PBFT, Elastico incurs an unacceptable failure probability of 8% with $f(k, m, p) = f(6, 16, 0.25)$ based on the result of [62], while it still incurs 2.76% with $f(k, m, p) = f(34, 100, 0.25)$ even extending to a larger-scale network of $m = 100$ (which can be the bottleneck [56]) by running powerful servers in cloud. This security issue has been hindering Elastico to be practically used, which are greatly resolved and improved by OmniLedger and RapidChain.

Insight 2: The traditional non-scalable PBFT incurs unacceptably high failure probability with total FT of only 25%, unless increasing the size of the consensus group, which leads to a chicken-and-egg problem due to huge communication overhead.

Generating Randomness - Distributed Commit-And-Xor Scheme

The distributed commit-and-xor scheme is implemented for the randomness generation in Elastico. It can be

categorized into the commit-and-then-reveal scheme [67], with an exception that the final result (randomness) varies depending on the different combinations of seeds λ_i every validator chooses. Concretely, the randomness generation is conducted by a global subset, i.e., the final committee, and it follows the procedures shown as below.

1) Each member of the final committee chooses a random seed λ_i in secret, and broadcasts $Hash(\lambda_i)$ to any other members in the final committee. After that, members in the final committee agree on a single set of hash values $\mathbb{S}$ [68], with numbers of $Hash(\lambda_i)$ ranging from $[2m/3, 3m/2]$ (m denotes the size of the final committee)[3].
2) Only if $\mathbb{S}$ collects at least $2m/3$ signatures, every validator in the final committee reveals their own seed λ_i to the public. By collecting and verifying all $2m/3$ (or $m/2+1$) pairs of $(\lambda_i, Hash(\lambda_i))$, the final randomness can be finalized by taking an XOR operation among them. Note that, in the case of $3m/2$ pairs are received, the chosen λ_i values need to be attached with the PoW solution in order to verify if the randomness is matched. This is because the combination of the seeds chosen by a validator can vary ($m/2+1$ out of $3m/2$).

This design, however, is not perfectly unbiased. It is exponential biased and bounded by the size of λ_i, i.e., $|\lambda_i|$, and m. In order to prevent the attacks from biasing the randomness by deliberately choosing a specific set of $m/2+1$ values of λ_i in his favor, $|\lambda_i|$ should be large enough as m also increases. This incurs large communication overhead, in addition to the overhead of the extra verification during PoW process. In the case of only $2m/3$ values of $(\lambda_i, Hash(\lambda_i))$ being received, the lack of Verifiable Secret Sharing (VSS) [69]–[73] forces all senders of these $2m/3$ values to be online all the time with no network outage or delay.

Insight 3: The distributed commit-and-xor scheme of Elastico has weak availability and robustness, and it is not a perfectly unbiased randomness generator unless paying more for the communication overhead.

3) BFT-BASED-CHAINSPACE

Chainspace uses an optimal implementation of PBFT, Mod-SMaRt [74], which accounts for the intra-part of the $\mathcal{S}$-**BAC** protocol proposed by Chainspace. However, Mod-SMaRt does not scale PBFT to address the issue of 1% attack. It decouples the communication and consensus primitives, while it only reduces the overhead of the latter with an unchanged overhead of $O(n^2)$ by replacing the process with the Validated and Provable Consensus (VP-Consensus). In addition, the high failure probability of the intra-consensus in Elastico also takes effects in Chainspace, which restricts the use of Chainspace in a large-scale network. Note that, the stages of *Propose* and *View change* take as input the elected leader, while the detail of randomness generator is not provided in [58].

[3]In fact, Elastico takes the discrepancies into account, where there can be $3m/2$ messages received by a validator while there are only m validators in the shard due to the network delay. In this case, other validators can choose only $(3m/2) \times (1/3) + 1 = m/2 + 1$ values of $Hash(\lambda_i)$ to generate their own randomness. In contrast, validators receiving only $2m/3$ values need to choose all $2m/3$ values of $Hash(\lambda_i)$ to generate their own randomness

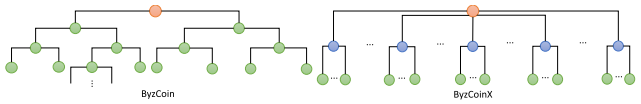

FIGURE 2. (Left) ByzCoin implements a tree with a fixed branching factor and an increasing depth. (Right) ByzCoinX implements a shadow tree with a fixed depth and an increasing branching factor.

4) BFT-BASED-OMNILEDGER

OmniLedger combines RandHound [75] and Algorand-based Verifiable Random Function (VRF) [19] to produce an unpredictable and unbiasable randomness under a 25% FT for re-allocation and leader-election of each shard and sub-group. Also, a new scalable BFT-based consensus algorithm, ByzCoinX, is proposed by optimizing ByzCoin [63], which resolves the dilemma of BFT-based 1% attack in sharding, by increasing the shard size to hundreds and up to a thousand.

Consensus Algorithm - ByzCoinX

Initially, ByzCoin [63] was the first scalable consensus protocol that combines PoW and BFT algorithms in a tree-based structure, by means of scalable collective signing (CoSi) [76], [77].

ByzCoinX[4] optimizes ByzCoin in terms of the better latency and more robust FT for a shard with hundreds of validators. Concretely, ByzCoinX implements a shallow tree with a fixed depth-3 and an increasing branching factor; see Fig. 2. Based on the shard size, each group leader is responsible for a group forming a sub-tree with a fixed number of group members. Note that, unlike ByzCoin implementing PoW to elect the group leader within a shifting window, ByzCoinX elects each group leader by the randomness generated at the beginning of the current epoch, followed by evenly allocating the rest of the validators into each group (thus the validators account for the leaves of each sub-tree). Also, the group leaders maintain their roles until a view change phase occurs, which eliminates the shifting window, as well as the difference of keyblocks and microblocks, as defined in ByzCoin. The leaders of each sub-tree aggregate at least 2/3 signatures from its children (leaves), followed by the signature regarding each group being sent to the root (protocol leader). The decision can be finalized whenever the root receives at least 2/3 signatures from its children (group leaders).

By using such a new tree-based structure, ByzCoinX can outperform ByzCoin by a better latency for a shard with hundreds of validators due to the shorter path from leaves to the root with a fixed depth, and a robust fault-tolerance due to the increasing branching factor. When the number of validators goes above a threshold, the latency of ByzCoin

[4]https://github.com/dedis/cothority/tree/master/byzcoinx

outperforms that of ByzCoinX due to the increasing branching factor. On the other hand, ByzCoinX can achieve a failure probability around 1.5% with $f(k, m, p) = f(48, 144, 0.25)$, and even 1% with $f(342, 1024, 0.3)$ at the cost of latency, as shown in Fig. 10 of [56].

Insight 4: ByzCoinX improves the scalability with a lower failure probability for the intra-consensus of OmniLedger, by sacrificing the transaction latency in large-scale networks.

Generating Randomness - Combination of RandHound and VRF

In order to address the issue of **Insight 3**, OmniLedger implements a scalable bias-resistant distributed randomness generator, RandHound [75], combined with a VRF-based leader election algorithm proposed by Algorand [19].

RandHound takes advantage of the following technologies to achieve an unbiasable and unpredictable randomness generator,

- Publicly VSS (PVSS) [71] that allows participating validators to be offline during the reveal phase (as opposed to the traditional commit-and-then-reveal scheme used in Elastico), by broadcasting the secret shares of the original λ_i in advanced;
- Schnorr Signature [78] that is the foundation of CoSi [76], [77] used in ByzCoinX and the threshold signatures [79]–[83],

so that the communication complexity can be reduced to $O(cm^2)$ from $O(m^3)$ (m denotes the total number of participating validators; c denotes the size of sub-group).

Several sub-groups are created by dividing the entire group of the participating validators, with c validators conducting PVSS within their sub-groups, respectively. Thus, a client (the leader randomly elected by the VRF) can receive the secret shares based on his choice from the corresponding sub-groups in a global run of CoSi. Consequently, the client can construct collective randomness by recovering the received secret shards. Meanwhile, a proof to verify the produced randomness is also recorded for third-party verifications.

OmniLedger implements a VRF-based election in order to randomly choose such a leader as the client among these participating validators. To be specific,

$$\mathcal{R}_{\mathcal{E},view,i}, \pi_{\mathcal{E},view,i} = VRF(config_{\mathcal{E}}||view, sk_i), \qquad (4)$$

where $config_{\mathcal{E}}$ denotes the settings pre-defined by a third-party; sk_i denotes the private key of a validator-i; $view$ denotes a view number related to a timeout Δ; $\mathcal{R}_{\mathcal{E},view,i}$ and $\pi_{\mathcal{E},view,i}$ denote the final randomness and its proof with specific epoch $\mathcal{E}$ and $view$ for validator-i. By default, the validator with the smallest $\mathcal{R}_{\mathcal{E},view,i}$ is selected to be the leader, and $view$ increases if this round of RandHound is timeout. In the case of $view > 5$ (proven $< 1\%$ by [56]), the RandHound is replaced by a coin-tossing scheme inspired by [84] that only implements a typical PVSS [72] in a poor complexity of order $O(m^3)$. On the other hand, this protocol still relies on third-party settings $config_{\mathcal{E}}$ pre-defined in the genesis block to prevent the attackers from biasing the result by secretly rerunning the protocol.

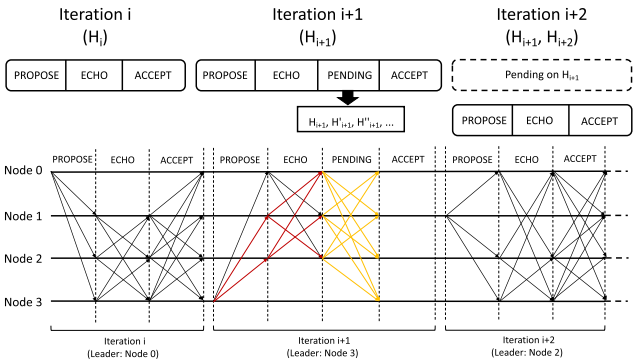

FIGURE 3. RapidChain implements a synchronous BFT-based consensus protocol by pre-scheduling the timeout, based on which the consensus speed can be adjusted by the system, hence achieving FT of 50%. In addition, RapidChain significantly improves the throughput by pipelining the conseusus process, i.e., re-proposing the previous pending blocks while agreeing on the current proposed block. The dark red arrows denote that the leader gossips more than one version of H_{i+1}, while the yellow arrows denote *pending* associated with the proposed header of iteration $i+1$.

Insight 5: The combination of RandHound and VRF suffers from the reliance on a third-party initial randomness pre-defined in the genesis block. A falling-back to an inefficient scheme occurs in the context of asynchronous networks, which limits the salability that RandHound could have guaranteed.

5) BFT-BASED-RAPIDCHAIN

RapidChain [57] implements a VSS-based [69] distributed random generation (DRG) protocol to agree on an unbiased randomness. On top of the DRG protocol, RapidChain addresses **Insight 5** by introducing a deterministic random graph where a certain fraction (50% with high probability [57]) of the number of malicious validators can be guaranteed in the initial set (the reference committee, similar to the final committee in Elastico), which will be discussed in Section III-C.4. Inspired by [85], in addition, RapidChain resolves the dilemma of BFT-based consensus algorithm in sharding, by increasing the FT of the intra-consensus protocol up to 50%.

Consensus Algorithm - 50% BFT With Pipelining

RapidChain aims for higher FT (50% BFT) of the intra-consensus protocol to address the dilemma of BFT-based 1% attack for sharding mechanisms with a small shard size. To be specific, RapidChain runs an autonomous pre-scheduled scheme within a shard to agree on a timeout Δ, based on which the consensus speed can be adjusted by the system to prevent the asynchronization. This ensures a synchronous network in the long-term, in which a non-responsive synchronous (with constant rounds) BFT-based consensus protocol with FT of 50% can be used.

However, re-proposing the pending block by the new leader in the next iteration greatly reduces the throughput by roughly half, while the current leader that is corrupted equivocates the consensus (if based on the original version of [85]). In order to address this issue, the pipelining is used

where pending blocks can be re-proposed along with the new block that is considered *safe*; see Fig. 3, (H_{i+1}, H_{i+2}) are proposed during iteration $i+2$. Note that, a new proposed block is considered *safe* so long as it points to a pending block that has been collected $m/2+1$ votes. Also note that, a valid vote can be either,

- *Temporary Vote:* an *echo* associated with the proposed header, H_i of iteration i; or,
- *Permanent Vote:* an *accept* associated with the proposed header, H_i of iteration i (if and only if there is only one version of header H_i received from the leader, and at least $m/2+1$ *echo*es of the same H_i received from others, tagging the header as *pending* otherwise).

As there exist multiple versions of headers associated with a specific iteration, e.g., $[H_{i+1}, H'_{i+1}, H''_{i+1}...]$ of iteration $i+1$, only one version is selected by the leader of iteration $i+2$ to be re-proposed along with H_{i+2}. Here, H_{i+2} is considered *safe* as H_{i+1} has been collected $m/2+1$ *echo*es serving as a proof in iteration $i+1$. Consequently, (H_{i+1}, H_{i+2}) are accepted if any nodes have received at least $m/2+1$ *echo*es associated with both H_{i+1} and H_{i+2}.

Referring to (1), the design of 50% BFT achieves a failure probability around 1.5% with $f(k, m, p) = f(17, 32, 0.33)$, and even 1% with $f(51, 100, 0.39)$ at a cost of communication overhead.

Insight 6: Differing from ByzCoinX in OmniLedger, the 50% BFT of RapidChain solves the BFT-based 1% attack by increasing the FT of intra-consensus protocol, nevertheless, this can only suit small-sized shards (not scalable with communication overhead of $O(n^2)$). In addition, the pre-scheduled scheme defining the timeout is not conceivably proved synchronous enough to run the pipelining 50% BFT.

Generating Randomness - VSS-Based DRG Protocol

The proposed DRG protocol by RapidChain, in fact, only implements a basic VSS-shares scheme, where all participating validators can reconstruct the final randomness r by the share of r (the share equals to $\sum_{l=1}^{m} \rho_{lj}$ calculated by other validators except validator-j) received from other validators. Note that, $\rho \in \mathbb{F}_p$ denoting a finite field of prime order p, and m denotes the size of the reference committee. As a result, the DRG protocol encounters a similar issue to that of any other typical VSS scheme, i.e., non-scalable (even though it suits with the 50% BFT in small-sized shards).

6) BFT-BASED POS-ETHEREUM 2.0

Ethereum has been running publicly as the first decentralized Blockchain platform (Blockchain 2.0 [86], [87]) that implements a Turing-complete programming language to develop smart contracts for the first time since 2014 [64]. With the gradually rising demands of high throughput, Casper-FFG with sharding (Shasper) is proposed [59] to allow the current Ethereum mainnet (a PoW-based single chain, also referred to Ethereum 1.0) to migrate to the new architecture stably and securely. Note that, we mainly focus on Shasper that has been running on testnet at the time of writing (referred to Ethereum 2.0), rather than the still-up-in-the-air Casper-CBC [88], based on which Ethereum plans to end up implementing a PoW-free Proof-of-Stake (PoS)-based sharded structure. Note that, only the intra-consensus protocol and cross-shard transactions of Shasper (referring to Phases 0-1, and Phase 4 in [89]) are discussed in this paper, because the other subprotocols have not yet been finalized based on the description in [59].

Consensus Algorithm - Solving the Intra-Consensus in a Global Way

Shasper also chooses to use the second method (presented in Section III-A), a BFT-based consensus algorithm, to solve the 1% attack issue of *intra-consensus*. Concretely, the Casper-FFG of Shasper can be regarded as a variation of BFT-based PoS consensus algorithms [15], [19] with careful designs for generating randomness, as opposed to the virtual-mining PoS consensus algorithms [90]–[92]. Note that, we assume a scalable BFT algorithm similar to ByzCoin [63] and ByzCoinX of OmniLedger is used in Shasper.

Shasper decouples the member allocation and consensus process, which leads to the fact that the intra-consensus within a shard also involves those validators from other shards being the attesters. The members of attesters group associated with a specific shard can be updated every slot. This also implies that an eligible validator in Shasper should at least store all block headers (headers is called collations in Shasper) of all shards regardless of which shard this validator is allocated at the beginning of every epoch. The procedures are summarized as follows.

1) To become a validator, a node needs to deposit a certain amount of *ETH* (currently it is set to 32*ETH* [93], [94]) in an official smart contract[5] on the original PoW-based mainnet. Having known the deposit, the system registers this node as a valid validator on a new individual chain, i.e., the beacon chain, while the beacon chain takes the role of a coordination device of the whole Shasper protocol in regards to managing the global validator pool, randomness generation, incentive, and message exchange.
2) An infrequent shuffling for the global validator pool is executed to re-allocate all validators to different shards based on the generated randomness. Such an epoch is currently set to 6.4*mins* [93], [95]. During each epoch, a proposer is elected based on the randomness from the local validator pool in each shard every 8*s* slot [93]. A proposed collation containing transactions of each shard is broadcast to all attesters assigned to the same shard, followed by a finalized collation being stored in the local ledger if the consensus process succeeds.
3) In addition to the hash value of each block on the PoW-based mainnet required to be stored on the beacon chain, a checkpoint is finalized by

[5]https://github.com/ethereum/eth2.0-specs/blob/dev/specs/core/0_deposit-contract.md

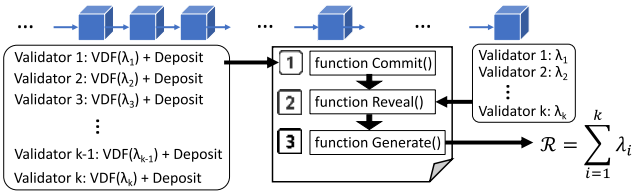

FIGURE 4. Ethereum 2.0 implements RANDAO and Verifiable Delay Function to generate randomness.

400 validators randomly selected from the global validator pool for each shard every 100 collations [96]. After that, these selected validators aggregate all checkpoints and upload them to the beacon chain. By storing the checkpoints as well as the collation headers of all shards, the beacon chain is able to obtain the local state and a group of finalized transactions (and its corresponding receipts) of each shard, referring to the *State root* and *Txgroup root* fields in the beacon chain headers, respectively. As a result, the deterministic finality can be achieved rather than a probabilistic one that Ethereum 1.0 used to rely on.

It is worth noting that the members (attesters) participating in the intra-consensus of a shard are, in fact, not limited to the indigenous validators (who have been allocated in a shard at the beginning of the epoch, and randomly selected by the generated randomness from the global pool). The group of attesters can be re-allocated for each proposed collation in a times slot, which provides the strongest security but incurs huge overhead when, 1) each shard conducts the consensus among continuously updated validators; 2) validators need to store data of more shards; and 3) the 1-slot-period re-allocation has to be executed.

Insight 7: The security level of Ethereum 2.0 - Shasper provides more flexible allocation for intra-consensus than that of any other considered sharding mechanisms, nevertheless, by incurring larger overhead.

Generating Randomness - Combination of RANDAO and VDF

RANDAO [97] is implemented based on the commit-and-then-reveal scheme [67] written in a pre-defined smart contract running on the beacon chain. To be specific, there are three functions defined in the smart contract, each of which must run in order; see Fig. 4. They are described as follows,

1) *Commit()*: all participating validators select a seed λ in secret (e.g., the hash of the parent block), after they have been deposited $32ETH$ in the smart contract. Then each of the validators runs a Verifiable Delay Function (VDF) [98] as a "hash onion" [96], [99],

$$VDF(\lambda_i) = Hash(Hash(Hash(...Hash(\lambda_i)))), \quad (5)$$

where the VDF conducts sufficient times of *Hash*(), e.g. 10, 000 times shown in [96] for a sufficiently long period ($102min$ [93]). As such, some malicious manipulation can be significantly prevented, e.g., deciding not to reveal its commitment if $\sum_i^{k-1} \lambda_i$ is found biased to k-th validator. The unbiased randomness is guaranteed by the VDF where only the serial computing can be run regardless of the computation power that is owned by this validator. Also note that, each validator can only commit once.

2) *Reveal()*: validators reveal their own seed λ to the smart contract, thus the contract can verify if the seed matches up with their corresponding commitment by verifying the 10, 000 preimages,

$$Hash^{-1}(Hash^{-1}(...Hash^{-1}(VDF(\lambda_i)))). \quad (6)$$

3) *Generate()*: the smart contract generates a randomness by adding up all λ_i. Punishment is applied to those who fail to reveal their own λ in time (corresponding to the time overhead of the defined VDF).

However, this design still suffers from three flaws, as shown in the following.

- A VDF consisting of n times $Hash(\cdot)$ incurs a computation overhead of $O(n)$, which is inefficient. There have been a few advanced VDF schemes proposed by the recent researches [100]–[102].
- This design is prone to the censorship attack [103]. Malicious validators can send irrelevant transactions with a high gas fee to fill up a block. Thus, the *Commit* may have to be interrupted as the gas limit of the block is run out.
- This design is also prone to the grinding attack [104] if the seed λ is based on the hash of the parent block, because validators can send arbitrary transactions, and try to find out the most biased seed by collecting different sets of transactions.

Insight 8: Current design of randomness generator in Ethereum 2.0 incurs high computation overhead, and is overwhelmingly dependent on the incentive scheme (punishment). It is prone to censorship attack and grinding attack, if the attack cost is acceptable.

B. ATOMICITY OF CROSS-SHARD

It is of importance that a sharding mechanism can support the cross-shard-verification and cross-shard transactions for validators allocated in different shards, according to the result shown in [56], [57] (showing that the probability of cross-shard transactions approaches to 100% as the total number of shards increases). Maintaining an individual global root chain may be one of the solutions to verification, but it does not natively support cross-shard transactions without any additional mechanism, e.g., lock/unlock operation in synchronous networks or lock-free operation in asynchronous networks. The demand for a secure protocol of cross-shard transactions gradually outweighs a naive mechanism lacking the support of cross-shard transactions (even it can achieve a high improving factor $\mathcal{N}$).

Differing from the traditional database system, the support of cross-shard transactions proposes a challenge to guarantee

the *Atomicity* of the data that was first defined in [60], [61] across multiple shards. Not only a simple payment transaction involving withdraw and deposit operations needs to be atomically protected, but also the demand for the complicated conditional statements attracts more attention to the contract-oriented *Atomicity*.

In this section, we compare and discuss the protocols to achieve *cross-shard-atomicity* in the considered sharding mechanisms. We focus on the design of cross-shard transaction, including Monoxide that supports asynchronous lock-free simple payment transactions; OmniLedger, RapidChain, and Ethereum 2.0 that supports simple payment transactions with lock/unlock scheme; and Chainspace that supports cross-shard operations for smart contracts (Elastico is vaguely discussed as it does not support atomic-safe cross-shard transactions.

1) MONOXIDE-RELAY TRANSACTIONS

In order to bypass the overhead of lock/unlock operation that greatly constrains the throughput and performance in regards to cross-shard transactions, Monoxide proposes *Eventual Atomicity* where a single cross-shard transaction is decoupled into an originated transaction in the local shard, and a relay transaction being put into the outbound transactions set (and hence becoming an inbound transaction when it is received by the destination shard). Rather than the immediate atomicity, *Eventual Atomicity* features its lock-free design and takes advantage of Chu-ko-nu mining across parallel shards in an asynchronous network, in order to maximize the global throughput via simple message exchange.

Concretely, the miners of shard a, i.e., an originate shard for a cross-shard transaction t, generate a relay transaction t_r in its local outbound transaction set if the withdraw operation passes the verification. Here, the withdraw operation is verified in the form of a local transaction t_l, decoupled from t, and stored in the local ledger. On the other hand, there are two additional MPT roots regarding, 1) the outbound transaction set; 2) the inbound transactions and local non-cross-shard transactions (denoted as MPT_O and MPT_I, respectively, and stored in the batch-chaining block defined in Chu-ko-nu mining). By means of MPT_O and MPT_I, the miners of shard b, i.e., the destination shard for t, are able to verify t_r via the attached proof,

$$[ShardID, ShardSize, BlockHeight, i, t_r, \pi_{t_r}], \tag{7}$$

where i denotes the index of t_r in the outbound transaction set generated by shard a; $BlockHeight$ denotes the height of block $\mathcal{B}$ that is stored t_l; π_{t_r} denotes the MPT proof of t_r in the given MPT with a root of MPT_O stored in the header of $\mathcal{B}$. Thus, it can be consequently observed that a cross-shard transaction in Monoxide achieves an improving factor of $\mathcal{N} = \frac{n}{2}$ as it is split into the locally-executed transactions and relay transactions expected to be outbound.

However, differing from the cross-shard transactions that can be proactively rejected by an acknowledgement from an entity (this is in charge by clients in OmniLedger, as discussed later), the chain forking in Monoxide can cause a reversion of the history and orphanize the block containing the t_l that has been executed within a shard. Without any existing of acknowledgement reminding the originated shard the status of t_r in the destination shard, the forking not only invalidates t_r in the destination shard (if t_r has been sent out before the forking occurs), but also invalidates all the subsequent cross-shard transactions relayed to any other shards. This implies the following drawbacks.

Incompatibility to Smart Contracts: There does not exist an upper-bound of timeout indicating if *Eventual Atomicity* of a cross-shard transaction has been finalized, leading to the incompatibility of conditional transactions, e.g., complicated operations in smart contracts.

Additional Latency: There must be λ confirmation blocks delaying the execution of the inbound transaction, i.e., t_r, in order to ensure the corresponding t_l in the originated shard is finalized and unlikely reverted. Also, the absence of acknowledgement and strict upper-bound of timeout deteriorates the latency and throughput due to the inevitable message loss, which incurs additional latency.

Unexpected Replay: To invalidate the inbound transactions t_r and all the subsequent t_rs due to the failure and reversion of t_l in the originated shard, and prevent the history of all destination shards from being reverted, the history needs to be rebuilt from the genesis block of each shard. This incurs unexpected overhead even if a checkpoint scheme is introduced, e.g., the shard pruning in OmniLedger [56].

Insight 9 In order to maximize the global throughput, Eventual Atomicity achieves the lock-free asynchronous cross-shard transactions at the cost of incurring Incompatibility to Smart Contracts, Additional Latency, and Unexpected Replay.

2) ELASTICO-NO CROSS-SHARD TRANSACTIONS

The elected leader of the traditional PBFT consensus algorithm in each shard finalizes and sends an agreement in regards to local transactions to a global subset, i.e., the final committee, as discussed in Section III-A.2. A final global block is stored in the global ledger and broadcast to all validators among the network, so that validators can verify the transactions from other shards. However, Elastico does not provide a secure protocol to ensure the atomicity across shards via this global ledger. There will be a fund loss as an unexpected dead-lock occurs if the cross-shard transaction sent to the destination shard gets rejected.

3) OMNILEDGER-ATOMIX PROTOCOL

To simplify the *cross-shard-atomicity*, OmniLedger proposes a client-driven Atomix protocol that is UTXO-based, where the communication overhead is shifted outside the shards. This indicates that the clients act as a gateway exchanging messages across multiple shards, by paying an extra cost of overhead.

Concretely, it consists of the following procedures.

1) *Initialize:* A UTXO-based cross-shard-transaction is created and gossiped to all input shards (ISs) by a client, where the inputs of this transaction spend UTXOs in some ISs, while outputs create new UTXOs in some output shards (OSs).
2) *Lock:* The cross-shard-transaction received from the client is stored in the local ledger within the shard after the verification is conducted. Meanwhile, either a *proof-of-acceptance* or a *proof-of-rejection* is created by the shard leaders attached with the corresponding CoSi, in the case that success or failure is returned by the verification, respectively. Therein, a *proof-of-acceptance* contains an MPT proof and the transaction itself.
3) *Unlock:*
   - *Unlock to Commit:* The client issues an *Unlock to Commit* consisting of the locked cross-shard transaction and the attached *proof-of-acceptance*, and gossip it to OSs, as soon as it receives *proof-of-acceptance* from all ISs. After the success of verification, OSs store the cross-shard transaction in the local ledger.
   - *Unlock to Abort:* The client issues an *Unlock to Abort* to those ISs issuing a *proof-of-acceptance* to unlock the state, once it receives a *proof-of-rejection* from one IS.

Consequently, a cross-shard transaction containing inputs from one single IS and OS can achieve an improving factor of $\mathcal{N} = \frac{n}{2}$, as this transaction is only stored in two shards, i.e., this IS and OS. On the other hand, inputs and outputs of multiple ISs and OSs result in the transaction being stored among the involved shards, i.e., an improving factor of $\mathcal{N} = 1$ in the worst case that the entire network is involved.

Insight 10: Atomix Protocol is, in fact, a band-aid at best. It sacrifices the support of light-weighted clients, but requires powerful performance for a client-driven exchange of messages.

Insight 11: Atomix Protocol has poorer support for UTXO-based cross-shard transactions as the number of participating shards increases, which is unable to take full advantage of the UTXO format.

4) RAPIDCHAIN-THREE-WAY CONFIRMATION

To verify a UTXO-based cross-shard transaction, there proposes a three-way confirmation in RapidChain to optimize the Atomix Protocol in OmniLedger, as shown in the bottom part of Fig. 5. Concretely, $k-1$ sub-transactions (Tx_0 and Tx_2) destined for each committee that stores its own I_i of the cross-shard transaction, with I_i as the inputs and I_i' as the outputs, respectively, and k is the number of inputs of this cross-shard transaction, are created by the output committee, i.e., C_3 as the C_{out}. After passing the verification on each input committees, i.e., C_2 and C_0 as the two C_{in}(s) of the original cross-shard transaction, Tx_0 and Tx_2 are stored in their own local ledger, respectively. Finally, all C_{in}(s) send the corresponding

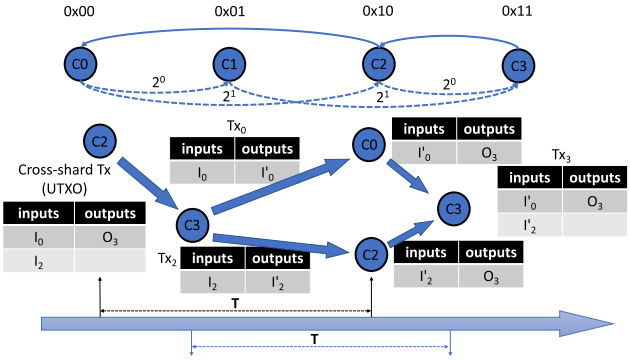

FIGURE 5. (Top) Each committee (shard) maintains a routing table containing $\log_2 n$ other committees. The routing table improves the efficient communication among multiple shards, as described in Section III-C.2. Committee C_0 can locate C_3 (via C_2) responsible for transactions with prefix 0x11. (Bottom) To cross-validate a UTXO-based cross-shard transaction requires this transaction to be spilt in three-way confirmation.

transactions back to C_3, and end up aggregating Tx_3 to be finally stored in the local ledger of C_3.

In order to determine the improving factor $\mathcal{N}$, we assume that a single committee can only be either a sender committee or a receiver committee (practically a shard can be both a sender or a receiver) at the same time for simplicity. In the worst case where a full-sized cross-shard transaction contains only the input from a single committee, C_{in} has to send this full-sized transaction twice (each corresponds to invoking the inter-communication once), i.e, 1-st and 3-rd handshaking. On the other hand, the period from C_{in} sending C_{out} the cross-shard transaction to it finishing verifying the sub-transactions received, equals to the period from C_{out} finishing verifying the original cross-shard transaction to it finishing verifying the confirmations sent by C_{in}, i.e., one block period. It is because the original cross-shard transaction is spilt into,

- the sub-transactions that are supposed to be stored in the local ledger of each C_{in} (a full-sized of the original cross-shard transaction with inputs from a single committee or inputs involving all committees);
- the final transaction that is supposed to be stored in the local ledger of C_{out} (another full-sized of the original cross-shard transaction) at the end of the protocol.

Consequently, either of these two kinds of transactions accounts for the intra-throughput of a committee, hence one block period, as shown by the **T** at the bottom of Fig. 5. Therefore, an improving factor of $\mathcal{N} = \frac{n}{2}$ can be achieved.

Insight 12: The routing table and three-way confirmation resolve the issue of OmniLedger, by significantly reducing the overhead of communication, even with a large number of participating shards in a single UTXO-based cross-shard transaction. However, by polluting specific routing tables, the eclipse attack [105] becomes a concern.

5) ETHEREUM 2.0-USING RECEIPTS

Having known the beacon chain, validators can not only address the issue of *intra-consensus*, but also address the

issue of *cross-shard-atomicity*, i.e., cross-verifying the normal transactions in each shard the validators care about, and enabling the cross-shard transactions. Note that, Shasper so far can only support a simple account-based (as opposed to the UTXO-based) payment transaction, while the design contract-oriented cross-shard transaction has not been finalized and presented.

The cross-shard transactions in Shasper rely on the receipts. Receipts correspond to accepted cross-shard transactions that are used to verify and log the validity of the transactions' operations. Also, the result of these operations can be obtained by the involved validators conducting cross-validation in the destination shards. By means of receipts whose identities are contained in *Txgroup root* field (Receipt root), the cross-shard transactions are split into multiple sub-transactions being executed in the originated and destination shards, respectively. This can be regarded as a variation of the synchronous lock/unlock scheme implemented in OmniLedger and RapidChain, while the receipts take the actual role of the lock.

Concretely, a proposed cross-shard transaction, t, is split into a group of t_1, t_2, and t_3.

1) The preliminary withdraw operation is executed and stored after t_1 is verified in the originated shard (input shard, namely IS). A receipt corresponding to t_1, denoted as r_1, is included in *Txgroup root* of the latest collation being proposed by the chosen proposer.
2) Having waited for a period that t_1 has been deterministically finalized by the checkpoints (this period can be shortened to meet different requirements, which is similar to the trust-but-verify transaction validation scheme proposed in OmniLedger; see the first point of Section III-C and **Insight 14**), a *proof-of-receipt* is sent to the destination shard (output shard, namely OS) as the second sub-transaction, i.e., t_2.
3) The OS can mark the r_1 as spent, as validators of the OS are able to verify the status of r_1 by the corresponding *Txgroup root* that is stored in the beacon chain, and the received *proof-of-receipt*. Meanwhile, the deposit operation is executed.
4) The OS sends a *proof-of-response* as t_3 to the original IS, indicating that the whole process of t has been finalized. Validators of the IS can finally confirm this fact by verifying the corresponding receipt of *proof-of-receipt* on the beacon chain.

Consequently, a cross-shard transaction that is account-based in Ethereum 2.0 - Shasper can achieve an improving factor of $\mathcal{N} = \frac{n}{3}$ due to the preliminary transaction, *proof-of-receipt*, and *proof-of-response*.

Insight 13: Ethereum 2.0 - Shasper introduces account-based cross-shard transactions by implementing the global (stored by all validators) beacon chain to exchange the essential message, i.e., the receipts and proofs. However, Shasper cannot be more than a transitional version due to the disadvantage of possible overhead.

6) CHAINSPACE-THE INTER-PART OF $\mathcal{S}$-BAC

$\mathcal{S}$-**BAC** refers to Sharded Byzantine Atomic Commit, whose intra-part makes use of an optimal PBFT, Mod-SMaRt, to handle the intra-consensus process; see Section III-A.3. Upon the intra-consensus being finalized within a shard (Chainspace allocates nodes in different shards based on the objects management, as described in Section III-C.6), the elected leader of the shard, the BFT-Initiator, takes responsibility for the atomicity of cross-shard transactions. It is worth noting that Chainspace makes use of the concept of BFT to ensure such atomicity, which constitutes the inter-part of $\mathcal{S}$-**BAC**.

Concretely, it resembles the Atomix Protocol in OmniLedger, with a crucial optimization where BFT consensus process must be conducted instead of a naive client-driven model. It consists of the following procedures.

1) *Initialize and Intra-Consensus:* An object-based cross-shard-transaction T is created by a client and gossip to all shards that manage the input objects, upon which the intra-consensus is conducted in each of these shards with an *accept* or *commit* broadcast to other concerned shards. Objects are set to *active* by the matching shards if ending up a commitment of T.
2) *Lock:* All involved objects in T are locked whenever a *commit* is received.
3) *Unlock:*
   - *Unlock to Commit:* The lock of each involved object in T is released if and only if *commit* is received from all concerned shards, upon which the objects are set to *inactive* and the output objects are created via BFT consensus process in a certain shard.
   - *Unlock to Abort:* The same locks are released whenever an *abort* is received, upon which the objects are set back to *active* and may be used by other subsequent transactions.

Similar to the problem the Atomix Protocol of OmniLedger has encountered, i.e., **Insight 11**, the improving factor upon a cross-shard transaction can be ranged from $\mathcal{N} = n$ to $\mathcal{N} = 1$ with T containing only one input object and no object being output, and T involving all objects around the entire network, respectively.

C. GENERAL IMPROVEMENTS

In this section, some general key challenges and improvements particularly proposed by the considered sharding mechanisms are listed. Such improvements can be generally implemented to address the new issues the considered sharding solutions pose to the entire system. They include transaction latency, inter-communication protocol, shards ledger pruning, decentralized bootstrapping, securing the epoch reconfiguration, and sharded smart contract.

1) REDUCING TRANSACTION LATENCY

Apart from the throughput, the transaction latency, referring to how long a transaction is deterministically confirmed and

finalized, is most likely more sensitive to individual users. It has been shown that the BFT-based 1% attack (refers to Section III-A) can be either resolved by implementing a scalable BFT consensus, e.g., OmniLedger and Ethereum 2.0, or increasing the FT within a single shard, e.g., RapidChain. However, it remains the issue of transaction latency, as described below.

- *The transaction latency deteriorates as a scalable BFT consensus features a large scale shard size to address the 1% attack,* according to the evaluation shown in [56], [63]. Thus, Omniledger introduces the *trust-but-verify transaction validation* scheme running within each shard to provide the real-time transaction confirmation time, which can also be implemented in any compatible sharding scheme, such as Ethereum 2.0. Concretely, validators of a shard are split into an optimistic group and a core group. The optimistic group is further split into multiple small sub-groups (even a sub-group with only one validator is allowed), hence each sub-group can verify the transactions in a real-time manner. Subsequently, the core group conducts the second verification, where the inconsistent and malicious transactions can be censored. Note that, there can be multiple inputs from multiple optimistic sub-groups to this second verification in a concurrent manner. Finally, the transactions passing the second verification can be contained in the proposed block and stored in the local ledger.

Insight 14: The real-time transaction latency is achieved by sacrificing the security, as the further 1% attack can still happen in optimistic groups. Similar to IoTA [25], this real-time transaction latency can only be used in specific scenarios with lower security requirements.

- *The transaction latency deteriorates as a non-scalable 50% BFT consensus incurs larger communication overhead.* Thus, upon the 50% consensus only agreeing on a digest of the block. RapidChain implements the *information dispersal algorithm* (IDA)-based gossip protocol [106], [107] to transmit large payload more efficiently. Concretely, the sender divides the original message into some n-equal-sized chunks, followed by applying an (m, n) erasure code scheme to encode the n chunks to m chunks. As a result, each node can reconstruct the original message by receiving valid n chunks from its neighbors with the help of some proofs, e.g, the MPT proofs, hence significantly reduces the latency.

2) INTER-COMMUNICATION PROTOCOL

Differing from the protocol to achieve the *atomicity-cross-shard*, the inter-communication protocol focuses on the overhead of data transmission among shards. The related schemes discussed in this survey include the following two major types.

- A global root chain acting as a message distributor is implemented, while each validator (or miner in the context of Monoxide) needs to store this chain. Sharding mechanisms using this kind include Ethereum 2.0, Monoxide with identical PoW targets, and Elastico[6].

Insight 15: The bottleneck is shifted to the global root chain due to its single-chained structure, as opposed to sharded structure. This can only be a transitional version but not a real solution.

- The most straightforward way is used by OmniLedger and Chainspace, i.e., full-mesh connection. This requirement tends to hold in those latency-sensitive systems, which incurs an considerable overhead.

In order to bypass the full-mesh connection, RapidChain proposes a novel inter-communication protocol based on a routing table stored by each validator; see the top side of Fig. 5. It is inspired by Kademlia-based [108] routing protocol, where each validator in a shard maintains a routing table containing all members of its shard as well as $\log_2 \log_2 n$ validators of other $\log_2 n$ shards which are distance 2^i for $0 \leq i \leq \log_2 n - 1$ away. The inter-communication is conducted by having all validators in the sender shard send messages to all validators on the receiver side. By taking advantage of P2P network, the communication overhead can be significantly reduced.

3) SHARDS LEDGER PRUNING

The reason most of the existing Blockchain system with a single-chained structure [1], [64], [109]–[111] tends to store the full version of its chain is that they intend to improve the communication and computation overhead of censorship and audition. Storing a full version of ledger of every shard incurs an unacceptable overhead of disk storage to validators, referring to the calculation in Section IV, as validators need to track the history of each shard in order to support the cross-shard transactions, as well as the re-allocation (bootstrapping) during each epoch. To solve this, OmniLedger proposes the design of state blocks (SB).

SBs of a shard summarizes the state as well as all transactions of its shard associated with each epoch. At the end of each epoch $\mathcal{E}_k$, the selected leader of a shard i constructs an MPT consisting of all the transactions, while the corresponding MPT root is stored in the header of $\mathrm{SB}_{i,k}$. As such, the body of $\mathrm{SB}_{i,k-1}$ can be pruned if $\mathrm{SB}_{i,k}$ passes the verification by other validations in shard i to become the new genesis block of $\mathcal{E}_{k+1}$. The regular blocks are also pruned as soon as $\mathrm{SB}_{i,k+1}$ is generated at the end of $\mathcal{E}_{k+1}$, during which it is the clients' responsibility to create and store the transaction proofs to prove the existence of a past transaction to other shards for cross-shard transactions.

The design of SBs is similar to stable checkpoints in PBFT [5], fast-sync mode in Ethereum [109],

[6]Elastico maintains a final committee where the finalized block is proposed and stored in the global root chain, based on the agreement from each shard. The global chains implemented by OmniLedger and RapidChain, i.e., the identity Blockchain and reference Blockchain, respectively, do not account for this kind as the messages exchanged by these two chains are not related to the actual transactions.

and stable checkpoints of Node Hash-Chains in Chainspace [58]. According to the evaluation in [58], such kind of pruning incurs an overhead of $O(m + \log T)$ for a *partial audit* and $O(T)$ for a *full audit*, where m denotes the shard size, and T denotes the number of transactions. The *partial audit* allows any users to obtain a proof to verify the existence of any transactions in any shards; the *full audit* allows a full verification by replaying the entire history of a shard. However, the design of SB raises two issues, 1) the overhead of transaction proofs might become the bottleneck, but it can still be relieved by introducing the Simple Payment Verification (SPV) [1], [109], several multi-hop backpointers [112]–[114], or Proofs of Proof of Work (PoPoW) [115], [116]; and 2) **Insight 16**,

Insight 16: The design of State blocks faces the same problem as that of the Atomix Protocol in OmniLedger and light-client protocol in Ethereum 1.0 (if used in Ethereum 2.0), i.e., shirking the most important duty to the client side.

4) DECENTRALIZED BOOTSTRAPPING

For sharding mechanisms involving a randomness generator that is responsible for a PoW-based entry ticket in the BFT-based intra-consensus protocol, it is important to select the initial set with an honest majority, e.g., the final committee in Elastico, and the reference committee in RapidChain[7].

Thus, RapidChain proposes a decentralized bootstrapping in the form of *sampler-graph election network* [57], with only a hardcoded seed and some network settings. In such an election network, participating validators are uniformly distributed into a few groups, within each of which a PoW-based result is computed by each member based on the randomness generated by the VSS-based DRG protocol (Section III-A.5) and its identification ID. Based on the result, a subgroup can be obtained for each group. Finally, a unique *root group* (it randomly selects the members of the reference committee) can be obtained with 50% honest majority (high probability), when this process is iterated. Consequently, the communication overhead can be improved from $\Omega(n^2)$ to $O(n\sqrt{n})$ with n denoting the total number of participating validators.

5) SECURING THE EPOCH RECONFIGURATION

For sharding mechanisms running a BFT-based intra-consensus protocol, (new) validators have to be swapped-out and re-allocated in other shards every epoch in order to prevent attacks from slowly adaptive adversaries, i.e., attacker can corrupt or Distributed Denial of Service (DDoS)-attack validators, but it takes a bounded time for such attacks to take effect. This indicates that the epoch length should be carefully designed to be lower than the bounded time.

[7]OmniLedger eliminates the necessity of an initial global set that responsible for verifying the PoW result, by using RandHound and VRF. However, an initial global randomness is still needed to derive VRF. Ethereum 2.0 builds the design on top of PoW-based mainnet, where the PoS-based Casper is used instead of PoW.

Recall that Elastico and Chainspace do not provide such a solution, while Ethereum 2.0 solves the intra-consensus with a global validator pool by frequently updating the member participating in the intra-consensus protocol for each shard. Both of them require validators to track the status of each shard to speed up the reconfiguration phase. OmniLedger implements a random permutation scheme to swap-out the validators, ensuring the number of validators being swapped is bounded by $k = \log n/m$ at a given time, where n denotes the total number of participating validators; m denotes the number of shards. Here, new validators that require to register their ID on a global identity Blockchain are also assigned to random shards. As such, the number of remaining honest validators can be sufficient to reach consensus while some are swapped-out, thus the idle phase can last shorter to improve the throughput. However, this scheme incurs a significant delay and scales moderately, which cause 1-day-long epoch that does not suit highly adaptive adversaries (when the bounded time becomes smaller).

In contrast, RapidChain proposes a light-weighted reconfiguration protocol based on the Cuckoo rule [117], [118], where only a constant number of validators are allowed to move between committees in each epoch. To be specific, the reference committee (C_r) announces a PoW puzzle based on the randomness generated in epoch $i-1$ ($\mathcal{R}_i$) by the DRG protocol, thus validators that wish to participate in epoch $i+1$ (including those that have participated in epoch $i-1$ and i) can solve the puzzle and inform C_r by the end of epoch i. During epoch $i+1$, C_r defines the active and inactive lists of validators of epoch $i+1$, and swap-out a constant number of validators from one to another committee based on $\mathcal{R}_{i+1}$ generated in epoch i. Finally, C_r agrees on a reference block stored in the local ledger of C_r, and broadcasts it to the entire network. This design, compared to that of OmniLedger, incurs less overhead and allows a more frequent epoch reconfiguration to suit more highly adaptive adversaries.

6) SHARDED SMART CONTRACT

None of the considered sharding mechanism has achieved the smart-contract-oriented sharded so far except Chainspace that introduces such functionality for the first time. Concretely, Chainspace, inspired by the UTXO model, proposes a new transaction structure based on new atoms *Objects* denoted as o. Here, o records state in the system with two kinds of unique identifier, i.e., id(o) (a cryptographically id that cannot be forged within a polynomial time) and types(o) (a pointer to a smart contract c that defines types(o)). Meanwhile, a contract c, referred to a special types of o, defines a *namespace* consisting of types(c) (the set of types that the specific c has defined) and a *checker* v denoted as $v(input) \rightarrow \{True, False\}$, as shown in (9). Such v is used to verify procedures proc(c), denoted as $p(input) \rightarrow output$ (defining the operation logic, as shown in (8)), by means of a pure function returning a Boolean value.

$$c.p(\mathbf{x}, \mathbf{r}, parameters) \rightarrow \mathbf{y}, returns; \tag{8}$$

$$c.v(p, \mathbf{x}, \mathbf{r}, parameters, \mathbf{y}, returns, dependencies) \rightarrow \{True, False\}; \quad (9)$$

$$[c, p, \mathbf{x}, \mathbf{r}, \mathbf{y}, parameters, returns, dependencies] \in Trace \in Transaction. \quad (10)$$

Note that, $\mathbf{x}$ denotes the input objects that must be active beforehand, and be set to inactive when the corresponding new output objects $\mathbf{y}$ set to active. $\mathbf{r}$ denotes the reference objects that must also be active, nevertheless, the status of $\mathbf{r}$ remains unchanged afterward. The *dependencies*, in the form of a list of *Trace*s from other contracts other than c, is along with all the other items (as shown in (10)) so that a single *Trace* can be obtained to constitute a *Transaction*.

The method to allocate nodes in different shards in Chainspace is by placing the nodes that manage, record, and verify the same set of o to a single shard, denoted as $\phi(o)$. Further, $\Phi(T)$ is defined to denote the *concerned nodes* of a transaction T, where *concerned nodes* represent the set of nodes managing all $\mathbf{x}$ or $\mathbf{r}$ of T. To verify a transaction T, all $\phi(o)$ with o being involved in T as input or reference should ensure the active status. Meanwhile, all $\Phi(T)$ (excluding the *dependencies*) should run the checker v of the corresponding contract c to validate the *Trace*s. As such, a cross-shard consensus algorithm that guarantees the atomicity of smart contracts, i.e., $\mathcal{S}$-**BAC**, is proposed (as discussed in Section III-B.6).

Insight 17: By modifying the transaction structure and involving the concept of the new atoms and objects, it can safely shard a smart contract with strong atomicity, but at the cost of considerable overhead and hence low throughput.

Up to this point, we have elaborated on the designs and protocols of each considered sharding mechanisms in terms of the *intra-consensus*, *cross-shard atomicity*, and *general improvements*, based on which a comprehensive comparison is presented in Table 2.

IV. EVALUATION

A. THE UPPER-BOUND OF THROUGHPUT

This section estimates the theoretical upper-bound of each discussed sharding mechanism, given the outbound bandwidth, disk storage space, and CPU process capability. Note that, Chainspace is not discussed in this section, because it pays the price in poor performance to be able to achieve sharding for Turing-complete smart contracts (**Insight 17**).

We choose a typical compute-optimized type of servers in either AWS or Ali cloud service, i.e., `c5.xlarge`. It features outbound bandwidth up to 200Mbps (25MB/s)[8], 4vCPU of Intel Xeon (Skylake) from 2.5GHz to 3.5GHz with Turbo boost, and 1TB basic disk storage space. This roughly costs 0.3USD/hour and 0.33USD/hour in AWS and Ali cloud service, respectively, with the storage fee around 100GB/0.01USD/hour. Table. 1 lists the notations of necessary parameters used in the calculation. We set the parameters to some values in order that bandwidth can be filled. Here, bandwidth is selected to be the upper-bound rather than disk storage and computation processing as the latter two metrics can be easily scaled in the cloud and cost much less than that of bandwidth.

Also note that the randomness generations of Elastico, OmniLedger, RapidChain, and Ethereum 2.0 are not discussed in this section, although the generation phase also incurs the overhead. This is because the generation is conducted only once in each $\mathcal{E}$, resulting in a predictable data burst that can be transiently scaled (the randomness generation is discussed in detail in Section III-A).

To be specific, the basic calculation of bandwidth, disk storage, and computation processing are defined as follows,

- *Bandwidth:* Dedicated channel for outbound message transmitting for the intra-consensus protocol and cross-shard operation on a single miner at the same time. Note that, whether a cross-shard transaction (cross-shard *Tx*) accounts for the intra-shard bandwidth or inter-shard bandwidth depends on whether the *Tx* should be inserted in local $\mathcal{C}$ of destination shard within a single $\mathbf{T}$.
- *Disk Storage:* Data storage permanently committed to the local database, including data both in the local shard and other shards.
- *Computation Processing:* CPU computation processing mainly corresponds to the verification of each *Tx* and *Sig*s of each $\mathcal{B}$ or $\mathcal{H}$. Without loss of generality, We consider that the verification of each *Tx* or *Sig* accounts for a single operation of computation processing.

1) MONOXIDE

Monoxide is the only sharding mechanism that supports Nakamoto consensus protocol with PoW for the intra-consensus among the discussed mechanisms in this paper. We consider $|\mathcal{B}| = 30KB$, $|\mathcal{H}| = 500B$, $|Tx| = 250B$, $|Sig| = 65B$ (we consider the signature format of Ethereum [64]), $\mathbf{T} = 12s$, $n = 262,144 = 2^{18}$, $m = 128$ and $h = 1,000,000$.

Bandwidth

- *Bandwidth Overhead Within Each Shard (Intra-Bandwidth):* This mainly corresponds to the transmitting of $\mathcal{B}$ within a single shard, i.e., $\frac{|\mathcal{B}|}{\mathbf{T}} = 2.5KB/s$.
- *Bandwidth Overhead Across All Shards (Inter-Bandwidth):* According to the eventual atomicity of cross-shard *Tx*s, a single cross-shard *Tx* is split into two parts that are inserted in $\mathcal{C}$ of source shard and destination shard, respectively. Each of the parts accounts for its corresponding intra-shard bandwidth. Thus, this mainly corresponds to the transmitting of the verification scheme of Chu-ko-nu mining. References [119] provides the expressions, as shown in the following,

[8]https://github.com/sivel/speedtestcli. `speedtest-cli` is used to test the bottleneck of inbound/outbound bandwidth on both AWS and Ali cloud. The average inbound bandwidth is 535.91Mbps, and the average outbound bandwidth is 202.56Mbps, while the latter matches with the 200Mbps displayed in the dashboard.

  - Mixed PoW targets of shards in one batch. This design allows miners to mine blocks in batch for different PoW targets and nonces. Blocks whose targets have been fulfilled can be sent out first, followed by the update of MPT and the further mining for those whose targets have yet to be fulfilled. This can be calculated by $\frac{n(|\mathcal{H}|+32\log_2(n))}{\mathbf{T}} = 22.4MB/s$, where $32\log_2(n)$ denotes the Merkle proof for Chu-ko-nu mining across shards.
  - Identical PoW targets of shards in one batch [119]. In this case, the design allows miners to mine blocks in batch for all n shards simultaneously with identical PoW targets and nonce. It sacrifices the decentralization to maintain a global subnet where all miners should participate, to broadcast $\mathcal{H}$ of all shards. We also let $n = 524,288 = 2^{19}$, hence the network size can be extended more, as calculated by $\frac{n|\mathcal{H}|}{\mathbf{T}} = 20.8MB/s$.

- *Throughput of a Single Shard (Intra-Throughput):* This is simply calculated by $\frac{|\mathcal{B}|}{|Tx|\mathbf{T}} = 10.24tps$.
- *Throughput of the Network (Inter-Throughput):* This can be calculated by multiplying the intra-throughput by the improving factor, i.e., $\frac{n}{2}$ for Monoxide (details refer to Section III-B.1), as shown in the following,

  - Mixed PoW targets of shards in one batch. This can be calculated by $\frac{10.24n}{2} = 1.23Mtps$, where $n = 262,144$.
  - Identical PoW targets of shards in one batch. This can be calculated by $\frac{10.24n}{2} = 2.56Mtps$, where $n = 524,288$.

The total bandwidth of both designs, i.e., identical and mixed PoW targets, have been upper-bounded, i.e., $20.8 < 22.4 < 25\ MB/s$. Here, the intra-bandwidth can be negligible due to its small size compared with that of the inter-bandwidth. Restricted by this, Monoxide can achieve nearly $1.23Mtps$ for mixed PoW targets, and $2.56Mtps$ for identical PoW targets by sacrificing the decentralization.

Disk Storage

As $\mathcal{B}$ contains $\mathcal{H}$, Txs, and Sigs, implying that $|\mathcal{B}|$ dominates in $|\mathcal{C}|$, as calculated by $h|\mathcal{B}| = |\mathcal{C}_h| = 28GB$. On top of that, Chu-ko-nu mining requires miners to track and synchronize block headers of all the shards they participate in (the more the number of shards being involved, the more secure Chu-ko-nu mining is), i.e., $\sum_i^{n-1}(|\widehat{\mathcal{C}_h}|) + h|\mathcal{B}| = (n-1)h|\mathcal{H}| + h|\mathcal{B}|$. This can be up to 119TB and 238TB for mixed and identical PoW targets, respectively. It indicates that a miner that only focuses on a single shard can reap a profit from the small disk storage, while Chu-ko-nu mining requires much more storage to guarantee security in the context of cross-sharding.

Computation Processing

Monoxide may have overwhelming computation processing than the other discussed sharding mechanisms due to the use of PoW. It requires as much processing as a normal PoW in a single shard as usual[9]. However, the hashrate varies with the total amount of computation power in a single shard (directly proportional to m) with a nearly fixed $\mathbf{T}$ to prevent a high orphan rate. We consider the hashrate to be the average Bitcoin hashrate of CPU used in the considered server (Intel Xeon), i.e., $66MH/s$ [120]. Here, any other PoW algorithms can replace as the kind of PoW is orthogonal to Monoxide. Besides, the computation processing also corresponds to the construction of the MPT of every pending block in each shard involved in the current round of Chu-ko-nu mining, as well as the verification of every intra-shard Tx and inter-shard Tx. These two kinds of Tx both account for the throughput of a single shard ($10.24tps$), which can be negligible compared to the PoW process. Thus, totally a $66MH/s$ of affordable CPU computation processing is needed in Monoxide.

In summary, a miner only conducting normal mining may only need to spend 0.21USD/hour and 0.24USD/hour in AWS and Ali cloud, respectively. In order to extend the disk space, miners participating in Chu-ko-nu mining across all shards need to spend about 36USD/hour and 40USD/hour in AWS and Ali cloud, respectively for mixed PoW targets, and 71USD/hour and 79USD/hour in AWS and Ali cloud, respectively for identical PoW targets. By only paying the price on the extended disk storage, Monoxide can achieve nearly $1.23Mtps$ for mixed PoW targets, and $2.56Mtps$ for identical PoW targets.

2) ELASTICO

Elastico is the first practical sharding mechanism where only the communication and processing are sharded while it still needs to be globally stored. We consider the intra-throughput is $1000tps$ (which is average among others with PBFT consensus algorithm [62]), $|\mathcal{B}| = (1000 \times 10 \times 250) + \frac{2m|Sig|}{3} \simeq 2.4MB$ where $\mathbf{T} = 10s$ and $|Tx| = 250B$, $|\mathcal{H}| = 500B$, $|Sig| = 65B$, $n = 48$, $m = 64$([10]), $h = 1,000,000$, and $\mathbf{E} = 10\ min$. The randomness is negligible due to its small size.

Bandwidth

Bootstrapping and ID generation are rarely conducted, also during which there is no block-oriented consensus being processed. On the other hand, the consensus of the final committee can use MPT root hash being transmitted to substitute $\mathcal{B}$ itself. Thus, the considered bandwidth here mainly corresponds to the intra-consensus protocol and cross-shard operation.

- *Bandwidth Overhead Within Each Shard:* This mainly corresponds to the transmitting of $\mathcal{B}$ during the

[9]Although the other discussed sharding mechanisms, e.g., Elastico and RapidChain, also conduct a PoW consensus during the stage of validators allocation to prevent the sybil attack, those miners participating in inter-shard communication may have to compete with those who do not attend in Monoxide. This is also the reason m does not account for any calculations of Monoxide. As a result, the hashrate of PoW in Monoxide is bound to be much higher than that of in Elastico or RapidChain, which should be considered in the calculation.

[10]This is $\frac{3}{2}$ of the minimum number of members in each shard, as defined in [55].

intra-consensus within a single shard, i.e., $\frac{m(|\mathcal{H}|+|\mathcal{B}|)+|\mathcal{B}|}{\mathbf{T}} = 14MB/s$. Here, an optimized PBFT can be used to prevent the block body from being broadcasting twice.

- *Bandwidth Overhead Across All Shards:* The bandwidth of a single miner corresponds to $n|\mathcal{B}|$ at most when it is a member of the final committee, and a global ledger is run and maintained locally. This is simply calculated by $\frac{n|\mathcal{B}|}{\mathbf{T}} = 11MB/s$. Note that, this does not indicate Elastico supports cross-shard *Tx*s as no atomicity can be guaranteed in Elastico, leaving a likely unsafe *Tx* being locked forever.
- *Throughput of a Single Shard:* This is simply defined as $1000tps$, as discussed previously.
- *Throughput of the Network:* This can be calculated by multiplying the intra-throughput by the improving factor of, i.e., n for Elastico. Thus, it is $1000n = 48ktps$.

The total bandwidth overhead of a single validator has been upper-bounded if we sum up the values of intra-bandwidth and inter-bandwidth, i.e., $14 + 11 \lesssim 25\ MB/s$. Restricted by this, Elastico can achieve nearly $48ktps$.

Disk Storage

As no ledger pruning scheme is introduced in Elastico, the periodical reshuffling of validators make all validators have to store a global ledger, which contains all $\mathcal{B}$ from all shards and costs a huge amount of disk storage. This can be simply calculated by $nh|\mathcal{B}| = 104.8TB$.

Computation Processing

The computation processing of PoW during the stage of reshuffling validators depends on the total amount of computation power among the entire network, given a fixed $\mathbf{T}$. As PoW does not account for the intra-consensus protocol in Elastico, while it is only conducted once every $\mathcal{E}$. We can neglect the computation processing of PoW in this calculation. In addition, the randomness generation is also conducted only once every $\mathcal{E}$ and can be negligible in this calculation (this assumption always holds for the rest of the discussed sharding mechanisms where a randomness is needed.). Thus, the following factors are considered for simplicity,

- As discussed above, Elastico does not support safe cross-shard *Tx*s due to the of a (un)lock scheme or a relay *Tx* scheme introduced in Monoxide. Thus, we have the verification for every individual *Tx* that equals to the intra-throughput, i.e., 1000H/s.
- If a considered miner is a member of the final committee, $2 \times \frac{2m|Sig|}{3\mathbf{T}} \simeq 555H/s$ can be obtained when the verification of $\mathcal{B}$ during PBFT process in the normal committees and final committee are both considered. In addition, each member of the final committee needs to verify *Tx*s that are aggregated from all m shards in the global ledger, i.e., $48kH/s$.

The total overhead of computation processing is roughly $50kH/s$, which is even smaller than that of Monoxide, i.e., $66MH/s$, and has yet to reach the bottleneck of the considered CPU.

In summary, validators participating in the final committee need to spend about 32USD/hour and 35USD/hour in AWS and Ali cloud, respectively. By paying the price on the extended disk storage, Elastico can achieve nearly $48ktps$.

3) OMNILEDGER

OmniLedger is the first practical sharding mechanism where bandwidth, storage, and processing are all sharded by means of a scalable intra-consensus, Atomix protocol, and the scheme of ledger pruning. We consider the intra-throughput is $1200tps$ (refers to Fig. 9 in [56]), $|\mathcal{B}| = 32MB$ (refers to Table 3 in [56]), $|Tx| = 500B$ (refers to *Size of Unlock Transactions* of Section IV in [56]), $|Sig| = 65B$ (this is not the size of CoSi [76]), $|\mathcal{H}| = 500B$, $n = 48$, $m = 1024$, $h = 1,000,000$, and $\mathbf{E} = 1\ day$. Thus, $\mathbf{T} = \frac{32M}{1200|Tx|} = 55s$ (nearly matches with Table 3 in [56]). The randomness is negligible due to its small size.

Bandwidth

Similar to Elastico, the considered bandwidth mainly corresponds to the intra-consensus protocol and cross-shard operation due to the conduct of Bootstrapping and ID generation for every one-day $\mathbf{E}$.

- *Bandwidth Overhead Within Each Shard:* This mainly corresponds to the transmitting of $|\mathcal{B}|$ during the intra-consensus within a single shard. Recall that, OmniLedger proposes ByzCoinX that implements a group-based scheme (rather than a tree-based scheme in ByzCoin [63]), where a single shard is partitioned into multiple consensus groups. Each group leader is selected based on the randomness generated for every epoch, and is unchanged unless a view change occurs. This group-based scheme can be a shadow-tree where the depth-3 is constant and the branching factor depends on the number of group leader. As a result, each validator only needs to broadcast $\mathcal{B}$ to its children in addition to a unicast of $\mathcal{B}$ to its parent. We consider the number of groups and group size are both $\sqrt{m}$ (refers to the same assumption of Section VI-D in [56]), the intra-bandwidth can be calculated by $\frac{\sqrt{m}|\mathcal{B}|+|\mathcal{B}|}{\mathbf{T}} = 19.2MB/s$, i.e., the bandwidth overhead of either the prepare phase or commit phase[11]. Here, the aggregated signature is negligible due to its small size compared to $\sum |Tx|$.
- *Bandwidth Overhead Across All Shards:* As Atomix protocol is client-driven, the inter-bandwidth mainly corresponds to the outbound bandwidth of clients rather than validators. Thus, the inter-bandwidth for a validator can be simply regarded as a unicast to the client, i.e., $\frac{|\mathcal{B}|}{\mathbf{T}} = 0.554MB/s$([12]). On the other hand, the client has to suffer from a huge amount of bandwidth overhead,

[11] *Tx*s are either transmitted in the prepare phase or commit phase, i.e., it is counted only once.

[12] As CoSi is used in ByzcoinX, $|\mathcal{B}|$ consists of the CoSi of each *Tx*, i.e., $\simeq 788.48B \times 1.2ktps = 0.9MB$, instead of $\frac{2m|Sig|}{3}$, where $788.48B$ refers to *Size of Unlock Transactions* of Section IV in [56].

i.e., $\frac{n|\mathcal{B}|}{\mathbf{T}} = 26.6MB/s > 25MB/s$, which has exceeded the upper-bound of the bandwidth of a single considered server.

- *Throughput of a Single Shard:* This is simply defined as $1200tps$ as discussed previously.
- *Throughput of the Network:* This can be calculated by multiplying intra-throughput by the improving factor, i.e., $\frac{n}{2}$ for OmniLedger with only one input shard and output shard involved; refer to Section III-B.3. Thus, it is $\frac{1200n}{2} = 28.8ktps$.

The total bandwidth overhead of a single validator has been upper-bounded if we sum up the values of intra-bandwidth and inter-bandwidth, i.e., $19.2 + 0.56 < 25\ MB/s$. Restricted by this, OmniLedger can achieve nearly $28.8ktps$, by shifting the bottleneck to clients.

Disk Storage

The disk storage in OmniLedger mainly corresponds to the ID Blockchain and the local pruned chain in each shard. We consider the size of a single ID, $|\mathcal{ID}| = 32B$.

- The block height of the ID Blockchain can be calculated by, $\frac{h\mathbf{T}}{\mathbf{E}} = 637$. Thus, $|\mathcal{B}_{\mathcal{ID},637}| = 637nm|\mathcal{ID}| = 0.93GB$.
- The shard ledger pruning can be achieved by constructing an MPT with the aggregated $\mathcal{B}s$ in the current $\mathcal{E}_k$, and end up finalizing a state block being the genesis $\mathcal{B}$ of $\mathcal{E}_{k+1}$ at the end of $\mathcal{E}_k$. Validators only need to store $\mathcal{H}$ of each state block, and all the regular $\mathcal{B}s$ of each $\mathcal{E}$. This can be calculated by $h|\mathcal{H}| + \frac{|\mathcal{B}|\mathbf{E}}{\mathbf{T}} \simeq 48GB$.

Computation Processing

This mainly corresponds to the computing overhead of the intra-consensus (ByzcoinX) and cross-shard operation (Atomix). The computing overhead in ByzcoinX consists of the verification of signature, i.e., $\frac{2m/3+1}{\mathbf{T}} = 12.4H/s$ and Txs, i.e., $1.2kH/s$ as defined. Validators log the cross-shard Txs in the local ledger and mark them as (un)locked one during the *Initialize* and *Unlock to Abort* of the client-driven Atomix protocol. This implies that the cross-shard Txs must account for the intra-Txs. As a result, a $1.2kH/s$ of the overhead of computation processing can be obtained, which is smaller than that of Monoxide, and has yet to reach the bottleneck of the considered CPU.

In summary, validators need to spend about 0.2USD/hour and 0.23USD/hour in AWS and Ali cloud, respectively. OmniLedger can achieve nearly $28.8ktps$ with fewer disk storage.

4) RAPIDCHAIN

RapidChain trades-off the protocol complexity for system robustness and achieves an efficient shard-driven cross-shard protocol by improving several parts of Elastico and OmniLedger. RapidChain also shards all of the bandwidth, storage, and processing. We consider the intra-throughput is $1000tps$, $|\mathcal{B}| = 8MB$ (refers to Fig. 3 in [58]), $|Tx| = 512B$, $|Sig| = 65B$, $|\mathcal{H}| = 500B$, $n = 256$, $m = 256$, $h = 1,000,000$ and $\mathbf{E} = 1day$. Thus, $\mathbf{T} = \frac{|\mathcal{B}|}{1000|Tx|} = 16.4s$. The randomness is negligible due to its small size.

Bandwidth

Similar to Elastico and OmniLedger, the considered bandwidth mainly corresponds to the intra-consensus protocol and cross-shard operation due to the conduct of Bootstrapping and ID generation for every one-day $\mathbf{E}$.

- *Bandwidth Overhead Within Each Shard:* RapidChain implements the IDA to transmit $\mathcal{B}s$ within a shard. We consider that the Reed-Solomon erasure codes [121] used in this protocol is (255, 233), leading to an actual $|\dot{\mathcal{B}}|$ roughly 12.5% larger than the metadata, i.e., $|\dot{\mathcal{B}}| = 9MB$. We further consider the parameter $\kappa = d = m - 1 = 255$, where κ and d denote the number of chunks and the number of neighbours of each validator, respectively. A single MPT proof incurs a size of $32\log_2(d) = 256B$. Thus, the bandwidth overhead to gossip $\mathcal{B}s$ by IDA is $\frac{|\dot{\mathcal{B}}|+256d}{\mathbf{T}} = 0.55MB/s$, where $|\dot{\mathcal{B}}|$ can be regarded as the size of chunks, and $256B$ denotes the total size of a single MPT proof sent to each neighbour.

  By means of the IDA-based gossip protocol, only $\mathcal{H}$ is needed in the intra-consensus protocol based on [85]. Thus, the bandwidth overhead can be calculated by $\frac{m|\mathcal{H}|\times 3}{\mathbf{T}} = 23kB/s$, which can be negligible. Note that, the multiplier 3 corresponds to 2-*nd*, 3-*rd*, and 4-*th* consensus rounds in every iteration, as described in Section III-A.5.
- *Bandwidth Overhead Across All Shards:* The cross-shard operation of RapidChain features a routing-table maintained by every validator in each shard. Every validator communicates with other $\log_2(n) \simeq 8$ shards, and records $\log_2\log_2(n) \simeq 3$ nodes of each other shard. As such, this can be $\frac{2(8\times 3)|\mathcal{B}|}{\mathbf{T}} = 23.4MB/s$. Here, the senders, in the worst case, incur a double overhead of cross-shard operation due to the "three-way confirmation"; refer to Section III-B.4.

  Another IDA gossiping is conducted by the shard leader after receiving the cross-shard $\mathcal{B}$, this can be another $\frac{|\dot{\mathcal{B}}|+256d}{\mathbf{T}} = 0.55MB/s$.
- *Throughput of a Single Shard:* This is simply defined as $1000tps$, as discussed previously.
- *Throughput of the Network:* This can be calculated by multiplying intra-throughput by the improving factor, i.e., $\frac{n}{2}$ in RapidChain (details refer to Section III-B.4). Thus, it is $\frac{1000n}{2} = 128ktps$.

The total bandwidth overhead of a single validator has been upper-bounded if we sum up the values of intra-bandwidth and inter-bandwidth, i.e., $23.4 + 0.55 \times 2 < 25MB/s$. Restricted by this, RapidChain can achieve nearly $128ktps$.

Disk Storage

The disk storage in RapidChain mainly corresponds to the ID in the local routing table, the local pruned chain in each shard by using the same scheme as that of OmniLedger, and

the ID Blockchain for a member of the reference committee. We consider the size of a single ID to be the same as that of OmniLedger, i.e., $|\mathcal{ID}| = 32B$.

- The routing table of a validator stores $\mathcal{ID}$ of all members in its committee, as well as $\log_2 \log_2 n$ validators of other $\log_2 n$ committees, i.e. $32m+32\log_2(\log_2(n))\log_2(n) = 9kB$.
- RapidChain suggests using the shard pruning scheme proposed in OmniLedger. Thus it can be calculated by $h|\mathcal{H}| + \frac{|\mathcal{B}|\mathbf{E}}{\mathbf{T}} \simeq 42GB$.

Computation Processing

Similar to Elastico, only the reconfiguration phase incurs the computation processing of PoW in RapidChain. We can also neglect this kind of computation overhead. Thus, the computation processing overhead mainly corresponds to the following two factors,

- The verification of *Tx*s and the corresponding *Sig*s, i.e., $\simeq 1000H/s$.
- As the leader of an output committee, the *Tx*s need to be verified when the leader first receives these *Tx*s from input committees. However, these *Tx*s will not be logged into the local ledger prior to the final confirmation; refer to Fig. 5, which implies the fact that the verification of these cross-shard *Tx*s should be independent to that of the local *Tx*s, i.e., $\simeq \frac{1000(n-1)}{\mathbf{T}} \simeq 16kH/s$.

As a result, a $16k + 1k = 17kH/s$ of the computation overhead can be obtained, which is still smaller than that of Monoxide, and has yet to reach the bottleneck of the considered CPU.

In RapidChain, it costs validators that participate in the reference committee nearly the same price as that of OmniLedger, i.e., 0.2USD/hour and 0.23USD/hour in AWS and Ali cloud, respectively, but with a significant breakthrough of the global throughput of nearly $128ktps$, i.e., $\sim$ 4.5x.

5) ETHEREUM 2.0

The Shasper of Ethereum 2.0 is a design that resolves the two major issues defined in Section III at the same time. Meanwhile, it also shards all of the bandwidth, storage, and processing. We consider $|\mathcal{B}_c|$ (collation in a shard) $= 1.5MB$, $|\mathcal{H}_c| = |\mathcal{H}_b|$ (size of a header on the beacon chain) $= 500B$, $|Tx| = 250B$, $|Sig| = 256B$, $\mathbf{T} = 8s$ (local chains and the beacon chain), $n = 512$, $m = 8$, $h = 1,000,000$ and $\mathbf{E} = 1week$. In addition, We also consider the number of attesters selected in each slot (several slots in one $\mathcal{E}$) is 9, the number of validators responsible for checkpoints is 400, and the checkpoint period is 100 [96]. The randomness is negligible due to its small size.

Bandwidth

To reach the consensus within a shard in Ethereum 2.0, the attesters are randomly selected from the global validators pool outside the local shard. This leads to the bandwidth mainly corresponding to only the intra-consensus, as well as all the other cross-shard operation. We consider that ByzCoinX proposed in OmniLedger is used for a large-scaled consensus group in this calculation as the actual protocol is not discussed and given in Ethereum 2.0. To be specific, We consider there exist $\sqrt{400} = 20$ sub-leaders, each of which contains $\sqrt{400} = 20$ children.

- *Bandwidth Overhead Within Each Shard:* This mainly corresponds to the transmitting of $\mathcal{B}_c$ within a single shard, i.e., $\frac{|\mathcal{B}_c|}{\mathbf{T}} = 192KB/s$.
- *Bandwidth overhead across all shard:* This mainly corresponds to two parts, i.e., to reach the consensus within a shard, and to upload to the beacon chain with another consensus in a single checkpoint period.

  Every $\mathbf{T} = 8s$, a proposer is randomly selected from the local validator pool within a shard, followed by 9 attesters are also randomly selected from the global validator pool. Note that, validators are evenly allocated in each local validator pool of each shard based on the randomness generated every $\mathcal{E}$. Also note that a validator can be both a potential attester from a global pool, and a proposer selected from its local pool. The selected proposer needs to collect at least $2/3$ signatures from the attesters to finalize a $\mathcal{B}_c$ to be stored in the local ledger of this slot. This can be calculated by $\frac{9(|\mathcal{B}_c|+\mathcal{H}_c|)}{\mathbf{T}} = 1.7MB/s$.

  Every checkpoint period contains 100 $\mathcal{B}_c$s, while the 400 validators as a global checkpoint-committee need to sign the tip $\mathcal{B}_c$ during the checkpoint period. This is also called notarization in Ethereum 2.0. By anchoring the checkpoint, history can be deterministically finalized and cannot be reverted. Concretely, it consists of the following steps,

  1) *Finalize the Checkpoints:* The required data size can be calculated by $n(20|\mathcal{B}_c| + |\mathcal{B}_c|) = 15.75GB$.
  2) *Upload to the Beacon Chain:* The required data size for the selected validators to upload the checkpoints of all shards can be calculated by $n(|\mathcal{B}_c| + \frac{400\times 2|Sig|}{3}) = 516MB$.
  3) *Consensus on the Beacon Chain:* The required data size can be calculated by $(\sqrt{nm}|\mathcal{H}_b| + |\mathcal{H}_b|) = 31.7KB$, as each validator should be aware of the body of the corresponding $\mathcal{B}_c$ during the previous steps.

  The three steps take at most $100\mathbf{T} = 800s$ to be finished, hence the considered inter-bandwidth is $\frac{15.75GB+516MB+31.7KB}{800} = 20.8MB/s$.
- *Throughput of a Single Shard:* This can be calculated by $\frac{|\mathcal{B}_c|}{|Tx|\mathbf{T}} = 787tps$.
- *Throughput of the Network:* This can be calculated by multiplying intra-bandwidth by the improving factor, i.e., $\frac{n}{3}$ for Ethereum 2.0 (details refer to Section III-B.5). Thus, it is $\frac{787n}{3} = 134ktps$.

The total bandwidth overhead of a single validator has been bounded if we sum up the values of both kinds of bandwidth overhead, i.e., $192KB + 1.7MB + 20.8MB < 25MB/s$. Restricted by this, Ethereum 2.0 can achieve nearly $134ktps$.

Disk Storage

The disk storage in Ethereum 2.0 mainly corresponds to the PoW-based main chain, the beacon chain, and the local chain of each shard that a validator cares more about. We consider the considered validators are in single-shard mode[13]. We consider the size of a single ID, $|\mathcal{ID}| = 32B$

- It is intended that most of the business logic and data, i.e., Txs, will be moved to the beacon chain for storage, while the original PoW-based main chain is only responsible for additional computation-based security, as well as a smart contract used to register and manage the validators. As a result, it can be regarded as a $\mathcal{C}$ with empty bodies (as if a light node in Ethereum [109]), which accounts for about 400MB at the time of writing [122].
- Each block of the beacon chain, i.e., $\mathcal{B}_b$ needs to store $\mathcal{H}_c$s from all involved shards, i.e., $nh|\mathcal{H}c| = 238GB$. In addition, the $\mathcal{ID}s$ all active validators need to be stored in the beacon chain, i.e., $32nm = 128KB$.
- Validators require to download the entire local ledger of the shard in which they are allocated, i.e., $h|\mathcal{B}_c| = 1.43TB$.

Computation Processing

We can neglect the PoW overhead, as a validator can involve itself in mining on the PoW-based main chain or not at will in Ethereum 2.0. Thus, the computation processing overhead mainly corresponds to the following two factors,

- A validator that is elected to be the attester to verify transactions for a single shard, without the loss of generality, can also be elected to be the attester for other shards (which is not discussed in details in any of the documents). We neglect the overhead of verifying signatures due to the small size of each group of attesters. Thus, the overhead of verifying transactions in n proposed $\mathcal{B}_c$s can be $787n = 403kH/s$.
- Every checkpoint period ($100\mathcal{B}_c$s of each shard) the checkpoint committee consisting of 400 validators finalizes the checkpoint of each shard. This corresponds to,
  - the 2/3 signatures required to reach the consensus for each checkpoint in every single shard, i.e., $\frac{n(400\times 2/3)}{800} = 171H$;
  - verifying transactions incurring $\frac{n|\mathcal{B}_c|}{800|Tx|} = 4kH/s$;
  - uploading checkpoints to the beacon chain with the consensus, i.e., $\frac{2nm}{800\times 3} = 3.4H/s$.

Note that, the verification of proposed $\mathcal{B}_c$s in each shard is independent to the verification of notarizing checkpoints. As a result, $\simeq 408kH/s$ of the computation overhead can be obtained, which is smaller than that of Monoxide, and has yet to reach the bottleneck of the considered CPU.

In Ethereum 2.0, validators need to spend about 0.39USD/hour and 0.42USD/hour in AWS and Ali cloud for disk extension, respectively, to achieve nearly $134ktps$. However, demand for stronger security incurs a huge overhead of disk storage for validators as they are most likely to be reallocated every $8s$-slot, which forces the validators to store the ledgers of every shard. As such, the huge overhead of disk storage is boosted to $\sim 100TB$ (similar to that of Monoxide and Elastico), i.e., a super-full node [59].

[13]The single-shard mode can be used rather than the super-full mode. A single-shard node processes the beacon chain blocks only, including the headers and signatures of the collation, i.e., $\mathcal{B}_c$ in each shard, but does not download and verify all the data of the $\mathcal{B}_c$s unless it cares more about.

B. COMPARISON AND DISCUSSION

This section, based on the calculation of the upper-bound of the throughput, provides a comparison among the considered sharding mechanisms, i.e., Monoxide, Elastico, OmniLedger, Rapidchain, Ethereum 2.0, and Chainspace. This comparison is also characterized as Table 3.

We conclude that RapidChain and Ethereum 2.0 implement optimizations that reduce restrictions of Elastico and OmniLedger, which leads to RapidChain and Ethereum 2.0 being the most advanced BFT-based sharding mechanisms in terms of throughput and cost. On the other hand, Monoxide pushes the upper-bound of throughput to Mega level, and opens up a new direction of the Nakamoto-based sharding mechanisms. Chainspace has plenty of room for performance improvement for sharded-smart contract.

Furthermore, we point out the challenges remaining unsolved practically, as well as the future trend being discussed.

1) FUTURE TREND FOR REDUCING THE OVERHEAD

Three common pitfalls in existing sharding mechanisms prevent the system from being horizontally scaled to the theoretical upper bound due to the communication and storage overhead.

- *An existing global chain that is needed to be stored by all participating miners/validators.* Such a global chain tends to be responsible for all global operations, such as generating randomness, cross-validating transactions in different shards, reshuffling operation. However, this simply poses the bottleneck threat back to a single global chain, which is the root issue sharding technologies would have tried to solve. **Insight 15** and SSChain [123] hit this pitfall. Note that SSChain simply utilizes a two-layer architecture where a global chain is set to deal with all data migration and reshuffling operations. **Trend 1: Restricting the use of a global chain in any operations, and the bottleneck requiring to be solved if used.**
- *Requiring miners/validators to store ledgers from other shards.* This is necessary in some of the existing sharding mechanisms in order to cross-validating transactions and reshuffling operation. However, it leads to miners/validators incurring high communication and storage overhead in $O(n)$ (n is the number of shards). **Insights 1**, **7**, **9**, **10**, **11**, **13** hit this pitfall. **Trend 2: Balancing the storage and communication overhead for miners/validators in sending cross-shard transactions and reshuffling, so that the order can be lower than**

$O(n)$**.** One of the potential solutions might be the fraud proof that enables light nodes to be as secure as full nodes without needing to store the whole ledger [124], yet it has not been mature at the time of writing.

- *Allocating participating nodes to shards based on their business requirements in order to bypass the overhead of using the sharding technology*. Business-driven members allocation for shards has been proposed and discussed in some designs, e.g., Ethereum 2.0 [96][14] in order to reduce, 1) the frequency that a participating node gets swapped out; and 2) the ratio of non-cross-shard transactions, for the ease of management and lower overhead. However, this results in a very long epoch reconfiguration for participating nodes and unevenly shard size, which ultimately poses a risk of crowed transactions to a single shard as time passes and the size and throughput increases, thus hitting the bottleneck of intra-consensus. **Trend 3: Avoiding simple business-driven members allocation that risks shards suffering from crowed transactions.**

2) FUTURE TREND FOR STRENGTHENING THE SECURITY AND ATOMICITY

This trend corresponds to the intra-consensus and atomicity of cross-shard transactions, respectively. We point out the potential direction on more secure intra-consensus and more efficient cross-shard transactions, as shown in the following.

Intra-.Consensus:

- *Trend 4: Scaling the unbiased and unpredictable randomness generator in large-scale networks with as few third-party hardcoded settings as possible.* The unbiased and unpredictable randomness plays an important role in BFT-based intra-consensus design. Improving this kind of algorithms can significantly prevent the validators from being under DDoS attacks. **Insights 3**, **5**, and **8** belong to this aspect.
- *Trend 5: Improving the PoW-based intra consensus, and generalizing it into other types of Nakamoto-based consensus algorithms.* Chu-ko-nu mining of Monoxide takes advantage of PoW to bypass the vortex of randomness, nevertheless, the security of which is dependent on the storage. As such, the future direction can be potentially decoupling the security and storage, and generalize the concept to other Nakamoto-based consensus algorithms, e.g., Proof-of-Stake.

Efficient Atomicity:

- *Trend 6: Enabling efficient conditional cross-shard transactions that enable contract-orient operations.* Only Chainspace and the future phase of Ethereum 2.0 claim to support such conditional cross-shard transactions so far, but at the cost of unacceptable overhead and latency, which requires more focus in the future trend.

[14]A possible design proposed by Ethereum 2.0 is to merge shards that interact more frequently than others.

V. CONCLUSION

This survey highlights the importance of sharding for the design of *scale-out* Blockchains and systematizes the state-of-the-art sharding mechanisms in regards to the intra-consensus security, atomicity of cross-shard transactions, and general challenges and improvements. We also proposed our calculations and insights analyzing the features and restrictions, based on which a comprehensive comparison among the considered sharding mechanisms was obtained.

A list of the key observations and conclusions are as follows:

- For the first time Monoxide proposes a Nakamoto-based sharding mechanism, but at the cost of storing headers of all shards to guarantee the maximum intra-consensus-safety.
- The traditional PBFT used in Elastico and Chainspace does not guarantee the intra-consensus-safety due to its weak scalability, while the BFT-based sharding mechanisms, i.e., OmniLedger, Rapidchain, and Ethereum 2.0, improve the intra-consensus-safety in the sense that scaling the traditional PBFT or increasing the fault tolerance of the traditional PBFT.
- The randomness generators of all considered sharding mechanisms in this paper need strict network settings, otherwise the unpredictiability and unbiasability in scaled networks will be compromised.
- Monoxide, OmniLedger, Rapidchain, and Ethereum 2.0 all propose their own solution to the issue of cross-shard transactions, none of which can support cross-shard smart contracts. Only Chainspace proposes a smart-contract-oriented sharding mechanism, but at the cost of low throughput.
- All considered sharding mechanisms introduce the optimizations to address the new challenges their proposed sharding mechanisms pose to the system, i.e., latency and storage, but further improvements are necessary.

ACKNOWLEDGMENT

The CRC program supports industry-led collaborations between industry, researchers and the community. UCOT Australia is a full-industry chain anti-counterfeiting traceability solution operator, dedicated to research and development of technology products based on Blockchain.

REFERENCES

[1] S. Nakamoto. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[2] O. Novo, "Blockchain meets IoT: An architecture for scalable access management in IoT," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1184–1195, Apr. 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 Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1508–1532, 2nd Quart., 2019.

[4] X. Wang, X. Zha, W. Ni, R. P. Liu, Y. J. Guo, X. Niu, and K. Zheng, "Survey on blockchain for Internet of Things," *Comput. Commun.*, vol. 136, pp. 10–29, Feb. 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0140366418306881

[5] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *Proc. OSDI*, vol. 99, Feb. 1999, pp. 173–186.
[6] M. Vukolić, "The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication," in *Proc. Int. Workshop Open Problems Netw. Secur.* Cham, Switzerland: Springer, 2015, pp. 112–125.
[7] F. Tschorsch and B. Scheuermann, "Bitcoin and beyond: A technical survey on decentralized digital currencies," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 2084–2123, 2016.
[8] J. Poon and T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments," Lightning Netw., San Francisco, CA, USA, Tech. Rep. DRAFT Version 0.5.9.2, 2016.
[9] *Raiden Network*. Accessed: 2015. [Online]. Available: https://raiden.network/
[10] J. Poon and V. Buterin, "Plasma: Scalable autonomous smart contracts," Lightning Netw. Ethreum, San Francisco, CA, USA, Tech. Rep. WORKING DRAFT, 2017, pp. 1–47.
[11] M. Jourenko, K. Kurazumi, M. Larangeira, and K. Tanaka, "SoK: A taxonomy for layer-2 scalability related protocols for cryptocurrencies," *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 352, Apr. 2019.
[12] R. Cattell, "Scalable SQL and NoSQL data stores," *SIGMOD Rec.*, vol. 39, no. 4, p. 12, May 2011.
[13] I. Eyal, A. E. Gencer, E. G. Sirer, and R. van Renesse, "Bitcoin-NG: A scalable blockchain protocol," in *Proc. 13th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, Santa Clara, CA, USA, Mar. 2016, pp. 45–59. [Online]. Available: https://www:usenix:org/conference/nsdi16/technical-sessions/presentation/eyal
[14] I. Bentov, R. Pass, and E. Shi, "Snow white: Provably secure proofs of stake," *IACR Cryptol. ePrint Archive*, vol. 2016, p. 919, Sep. 2016.
[15] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Proc. Annu. Int. Cryptol. Conf.* Cham, Switzerland: Springer, 2017, pp. 357–388.
[16] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song, "The honey badger of BFT protocols," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*. New York, NY, USA: ACM, 2016, pp. 31–42.
[17] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham, "HotStuff: BFT consensus with linearity and responsiveness," in *Proc. ACM Symp. Princ. Distrib. Comput. (PODC)*. New York, NY, USA: ACM, 2019, pp. 347–356, doi: 10.1145/3293611.3331591.
[18] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, and E. Wong, "Zyzzyva: Speculative byzantine fault tolerance," *ACM SIGOPS Operating Syst. Rev.*, vol. 41, no. 6, pp. 45–58, 2007.
[19] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proc. 26th Symp. Operating Syst. (SOSP)*. New York, NY, USA: ACM, 2017, pp. 51–68, doi: 10.1145/3132747.3132757.
[20] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. G. Sirer, D. Song, and R. Wattenhofer, "On scaling decentralized blockchains," in *Financial Cryptography and Data Security*, J. Clark, S. Meiklejohn, P. Y. Ryan, D. Wallach, M. Brenner, and K. Rohloff, Eds. Berlin, Germany: Springer, 2016, pp. 106–125.
[21] J. Garzik. *Bip102: Block Size Increase to 2MB*. Accessed: 2015. [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0102.mediawiki
[22] P. Wuille. *Bip103: Block Size Following Technological Growth*. Accessed: 2015. [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0103.mediawiki
[23] E. Lombrozo, J. Lau, and P. Wuille, "Bip141: Segregated witness (consensus layer)," Bitcoin Improvement Proposal, Tech. Rep. Bip141, 2015.
[24] Y. Sompolinsky and A. Zohar, "Secure high-rate transaction processing in bitcoin," in *Financial Cryptography and Data Security*, R. Böhme and T. Okamoto, Eds. Berlin, Germany: Springer, 2015, pp. 507–527.
[25] S. Popov, "The tangle," IoTA Found., Berlin, Germany, Tech. Rep. Version 1.3, 2016, p. 131.
[26] A. Churyumov. (2016). *Byteball: A Decentralized System for Storage and Transfer of Value*. [Online]. Available: https://byteball.org/Byteball.pdf
[27] L. Baird, "The swirlds hashgraph consensus algorithm: Fair, fast, byzantine fault tolerance," Swirlds, College Station, TX, USA, White Paper SWIRLDS-TR-2016-01, 2016.
[28] Y. Sompolinsky, Y. Lewenberg, and A. Zohar, "Spectre: A fast and scalable cryptocurrency protocol," *IACR Cryptol. ePrint Arch.*, vol. 2016, p. 1159, Jan. 2018.
[29] Y. Sompolinsky and A. Zohar, "Phantom: A scalable blockdag protocol," *IACR Cryptol. ePrint Arch.*, vol. 2018, p. 104, Mar. 2018.
[30] C. Li, P. Li, D. Zhou, W. Xu, F. Long, and A. Yao, "Scaling nakamoto consensus to thousands of transactions per second," May 2018, *arXiv:1805.03870*. [Online]. Available: https://arxiv.org/abs/1805.03870
[31] L. Kan, Y. Wei, A. H. Muhammad, W. Siyuan, G. Linchao, and H. Kai, "A multiple blockchains architecture on inter-blockchain communication," in *Proc. IEEE Int. Conf. Softw. Qual., Rel. Secur. Companion (QRS-C)*, Jul. 2018, pp. 139–145.
[32] H. Wang, Z. Zheng, S. Xie, H. N. Dai, and X. Chen, "Blockchain challenges and opportunities: A survey," *Int. J. Web Grid Services*, vol. 14, no. 4, p. 352, 2018.
[33] L. S. Sankar, M. Sindhu, and M. Sethumadhavan, "Survey of consensus protocols on blockchain applications," in *Proc. 4th Int. Conf. Adv. Comput. Commun. Syst. (ICACCS)*, Jan. 2017, pp. 1–5.
[34] W. Gao, W. G. Hatcher, and W. Yu, "A Survey of Blockchain: Techniques, applications, and challenges," in *Proc. 27th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Jul. 2018, pp. 1–11.
[35] W. Wang, D. T. Hoang, P. Hu, Z. Xiong, D. Niyato, P. Wang, Y. Wen, and D. I. Kim, "A survey on consensus mechanisms and mining strategy management in blockchain networks," *IEEE Access*, vol. 7, pp. 22328–22370, 2019.
[36] W. Yang, S. Garg, A. Raza, D. Herbert, and B. Kang, "Blockchain: Trends and future," in *Knowledge Management and Acquisition for Intelligent Systems*, K. Yoshida and M. Lee, Eds. Cham, Switzerland: Springer, 2018, pp. 201–210.
[37] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *Proc. IEEE Int. Congr. Big Data (BigData Congr.)*, Jun. 2017, pp. 557–564.
[38] S. Goswami, "Scalability analysis of blockchains through blockchain simulation," M.S. thesis, Dept. Comput. Sci., Univ. Nevada, Las Vegas, Las Vegas, NV, USA, 2017. [Online]. Available: https://digitalscholarship.unlv.edu/thesesdissertations/2976/
[39] M. Bez, G. Fornari, and T. Vardanega, "The scalability challenge of ethereum: An initial quantitative analysis," in *Proc. IEEE Int. Conf. Service-Oriented Syst. Eng. (SOSE)*, Apr. 2019, pp. 167–176.
[40] C. Worley and A. Skjellum, "Blockchain tradeoffs and challenges for current and emerging applications: Generalization, fragmentation, sidechains, and scalability," in *Proc. IEEE Int. Conf. Internet Things (iThings) IEEE Green Comput. Commun. (GreenCom) IEEE Cyber, Phys. Social Comput. (CPSCom) IEEE Smart Data (SmartData)*, Jul. 2018, pp. 1582–1587.
[41] S. Kim, Y. Kwon, and S. Cho, "A survey of scalability solutions on blockchain," in *Proc. Int. Conf. Inf. Commun. Technol. Converg. (ICTC)*, Oct. 2018, pp. 1204–1207.
[42] A. Chauhan, O. P. Malviya, M. Verma, and T. S. Mor, "Blockchain and Scalability," in *Proc. IEEE Int. Conf. Softw. Qual., Rel. Secur. Companion (QRS-C)*, Jul. 2018, pp. 122–128.
[43] F. Casino, T. K. Dasaklis, and C. Patsakis, "A systematic literature review of blockchain-based applications: Current status, classification and open issues," *Telematics Inform.*, vol. 36, pp. 55–81, Mar. 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0736585318306324
[44] D. Mechkaroska, V. Dimitrova, and A. Popovska-Mitrovikj, "Analysis of the possibilities for improvement of blockchain technology," in *Proc. 26th Telecommun. Forum (TELFOR)*, Nov. 2018, pp. 1–4.
[45] T. T. A. Dinh, R. Liu, M. Zhang, G. Chen, B. C. Ooi, and J. Wang, "Untangling blockchain: A data processing view of blockchain systems," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 7, pp. 1366–1385, Jul. 2018.
[46] Y. Xiao, N. Zhang, W. Lou, and Y. T. Hou, "A survey of distributed consensus protocols for blockchain networks," *CoRR*, vol. abs/1904.04098, pp. 1–34, Apr. 2019. [Online]. Available: http://arxiv.org/abs/1904.04098
[47] R. Wang, K. Ye, and C.-Z. Xu, "Performance benchmarking and optimization for blockchain systems: A survey," in *Blockchain—ICBC*, J. Joshi, S. Nepal, Q. Zhang, and L.-J. Zhang, Eds. Cham, Switzerland: Springer, 2019, pp. 171–185.
[48] 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.
[49] P. Singhal and S. Masih, "Metaanalysis of methods for scaling blockchain technology for automotive uses," *CoRR*, vol. abs/1907.02602, pp. 1–11, Jul. 2019. [Online]. Available: http://arxiv.org/abs/1907.02602

[50] A. Meneghetti, T. Parise, M. Sala, and D. Taufer, "A survey on efficient parallelization of blockchain-based smart contracts," *CoRR*, vol. abs/1904.00731, pp. 1–9, Feb. 2019. [Online]. Available: http://arxiv.org/abs/1904.00731

[51] G. Wang, Z. J. Shi, M. Nixon, and S. Han, "SoK: Sharding on Blockchain," in *Proc. 1st ACM Conf. Adv. Financial Technol. (AFT)*. New York, NY, USA: ACM, 2019, pp. 41–61, doi: 10.1145/3318041.3355457.

[52] J. Wang and H. Wang, "Monoxide: Scale out blockchains with asynchronous consensus zones," in *Proc. 16th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*. Boston, MA, USA: USENI Association, Feb. 2019, pp. 95–112. [Online]. Available: https://www.usenix.org/conference/nsdi19/presentation/wang-jiaping

[53] J. C. Corbett *et al.*, "Spanner: Google's globally distributed database," *ACM Trans. Comput. Syst.*, vol. 31, no. 3, pp. 8:1–8:22, Aug. 2013, doi: 10.1145/2491245.

[54] G. Danezis and S. Meiklejohn, "Centrally banked cryptocurrencies," *CoRR*, vol. abs/1505.06895, pp. 1–15, May 2015. [Online]. Available: http://arxiv.org/abs/1505.06895

[55] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2016, doi: 10.1145/2976749.2978389.

[56] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "OmniLedger: A secure, scale-out, decentralized ledger via sharding," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2018, pp. 583–598.

[57] M. Zamani, M. Movahedi, and M. Raykova, "RapidChain: Scaling blockchain via full sharding," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*. New York, NY, USA: ACM, 2018, pp. 931–948, doi: 10.1145/3243734.3243853.

[58] M. Al-Bassam, A. Sonnino, S. Bano, D. Hrycyszyn, and G. Danezis, "Chainspace: A sharded smart contracts platform," *CoRR*, vol. abs/1708.03778, pp. 1–16, Aug. 2017. [Online]. Available: http://arxiv.org/abs/1708.03778

[59] V. Buterin. (Apr. 2019). *Ethereum Sharding FAQ*. Accessed: Aug. 1, 2019. [Online]. Available: https://github.com/ethereum/wiki/wiki/Sharding-FAQ

[60] J. Gray, "The transaction concept: Virtues and limitations," in *Proc. VLDB*, vol. 81, 1981, pp. 144–154.

[61] T. Haerder and A. Reuter, "Principles of transaction-oriented database recovery," *ACM Comput. Surv.*, vol. 15, no. 4, pp. 287–317, Dec. 1983.

[62] 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 *Proc. ACM Int. Conf. Manage. Data (SIGMOD)*. New York, NY, USA: ACM, 2017, pp. 1085–1100, doi: 10.1145/3035918.3064033.

[63] 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 *Proc. 25th USENIX Secur. Symp. (USENIX Secur.)*. Austin, TX, USA: USENIX Association, Aug. 2016, pp. 279–296. [Online]. Available: https://www.usenix:org/conference/usenixsecurity16/technical-sessions/presentation/kogias

[64] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, Apr. 2014.

[65] A. Judmayer, A. Zamyatin, N. Stifter, A. G. Voyiatzis, and E. Weippl, "Merged mining: Curse or cure?" in *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, J. Garcia-Alfaro, G. Navarro-Arribas, H. Hartenstein, and J. Herrera-Joancomartí, Eds. Cham, Switzerland: Springer, 2017, pp. 316–333.

[66] BitCoinWIKI. (Aug. 2015). *Merged Mining Specification*. Accessed: Aug. 1 2019. [Online]. Available: https://en.bitcoin.it/wiki/Merged_mining_specification

[67] M. Naor, "Bit commitment using pseudorandomness," *J. Cryptol.*, vol. 4, no. 2, pp. 151–158, Jan. 1991, doi: 10.1007/BF00196774.

[68] M. Pease, R. Shostak, and L. Lamport, "Reaching agreement in the presence of faults," *J. ACM*, vol. 27, no. 2, pp. 228–234, Apr. 1980, doi: 10.1145/322186.322188.

[69] P. Feldman, "A practical scheme for non-interactive verifiable secret sharing," in *Proc. 28th Annu. Symp. Found. Comput. Sci. (SFCS)*, Oct. 1987, pp. 427–438.

[70] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Advances in Cryptology—CRYPTO*, J. Feigenbaum, Ed. Berlin, Germany: Springer, 1992, pp. 129–140.

[71] M. Stadler, "Publicly verifiable secret sharing," in *Advances in Cryptology—EUROCRYPT*, U. Maurer, Ed. Berlin, Germany: Springer, 1996, pp. 190–199.

[72] B. Schoenmakers, "A simple publicly verifiable secret sharing scheme and its application to electronic voting," in *Advances in Cryptology—CRYPTO*, M. Wiener, Ed. Berlin, Germany: Springer, 1999, pp. 148–164.

[73] T. Rabin and M. Ben-Or, "Verifiable secret sharing and multiparty protocols with honest majority," in *Proc. 21st Annu. ACM Symp. Theory Comput. (STOC)*. New York, NY, USA: ACM, 1989, pp. 73–85, doi: 10.1145/73007.73014.

[74] J. Sousa and A. Bessani, "From byzantine consensus to BFT state machine replication: A latency-optimal transformation," in *Proc. 9th Eur. Dependable Comput. Conf.*, May 2012, pp. 37–48.

[75] E. Syta, P. Jovanovic, E. K. Kogias, N. Gailly, L. Gasser, I. Khoffi, M. J. Fischer, and B. Ford, "Scalable bias-resistant distributed randomness," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 444–460.

[76] D. Boneh, M. Drijvers, and G. Neven, "Compact multi-signatures for smaller blockchains," in *Advances in Cryptology—ASIACRYPT*, T. Peyrin and S. Galbraith, Eds. Cham, Switzerland: Springer, 2018, pp. 435–464.

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

[78] C. P. Schnorr, "Efficient signature generation by smart cards," *J. Cryptol.*, vol. 4, no. 3, pp. 161–174, 1991, doi: 10.1007/BF00196725.

[79] C. Stathakopoulous and C. Cachin, "Threshold signatures for blockchain systems," Swiss Federal Inst. Technol., Zürich, Switzerland, Tech. Rep. RZ3910 (#ZUR1704-014), 2017.

[80] Y. Desmedt and Y. Frankel, "Threshold cryptosystems," in *Advances in Cryptology—CRYPTO*, G. Brassard, Ed. New York, NY, USA: Springer, 1990, pp. 307–315.

[81] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, "Robust threshold DSS signatures," in *Advances in Cryptology—EUROCRYPT*, U. Maurer, Ed. Berlin, Germany: Springer, 1996, pp. 354–371.

[82] V. Shoup, "Practical threshold signatures," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.* Cham, Switzerland: Springer, 2000, pp. 207–220.

[83] A. Boldyreva, "Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme," in *Public Key Cryptography—PKC*, Y. G. Desmedt, Ed. Berlin, Germany: Springer, 2002, pp. 31–46.

[84] C. Cachin, K. Kursawe, and V. Shoup, "Random oracles in constantinople: Practical asynchronous byzantine agreement using cryptography," *J. Cryptol.*, vol. 18, no. 3, pp. 219–246, Jul. 2005, doi: 10.1007/s00145-005-0318-0.

[85] L. Ren, K. Nayak, I. Abraham, and S. Devadas, "Efficient synchronous byzantine consensus," *CoRR*, vol. abs/1704.02397, pp. 1–19, Apr. 2017. [Online]. Available: http://arxiv.org/abs/1704.02397

[86] M. Swan, *Blockchain : Blueprint for a New Economy*. Sebastopol, CA, USA: O'Reilly Media, 2015. [Online]. Available: http://shop.oreilly.com/product/0636920037040.do

[87] X. Li, P. Jiang, T. Chen, X. Luo, and Q. Wen, "A survey on the security of blockchain systems," *Future Gener. Comput. Syst.*, to be published, [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X17318332

[88] V. Zamfir. (Nov. 2018). *Casper-CBC FAQ*. Accessed: Aug. 1, 2019. [Online]. Available: https://github.com/ethereum/cbc-casper/wiki/FAQ

[89] J. Ray. (Mar. 2019). *Sharding Roadmap*. Accessed: Aug. 1, 2019. [Online]. Available: https://github.com/ethereum/wiki/wiki/Sharding-roadmap

[90] S. King and S. Nadal. *PPcoin: Peer-to-Peer Crypto-Currency With Proof-of-Stake*. Aug. 2012. [Online]. Available: https://pdfs.semanticscholar.org/0db3/8d32069f3341d34c35085dc009a85ba13c13.pdf

[91] C. Xu, K. Wang, P. Li, S. Guo, J. Luo, B. Ye, and M. Guo, "Making big data open in edges: A resource-efficient blockchain-based approach," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 4, pp. 870–882, Apr. 2019.

[92] (2018). *Reddcoin*. [Online]. Available: https://wiki.reddcoin.com/Main_Page

[93] V. Buterin. (Aug. 2018). *Convenience Link to Casper+Sharding Chain V2.1 SPEC*. Accessed: Aug. 1, 2019. [Online]. Available: https://ethresear.ch/t/convenience-link-to-casper-sharding-chain-v2-1-spec/2332

[94] J. Y. Park. (Dec. 2018). *Preparing for Ethereum PoS Staking in 2019*. Accessed: Aug. 1, 2019. [Online]. Available: https://medium.com/whaley-official/getting-prepared-for-ethereum-pos-staking-in-2019-3a3855e6a018
[95] J. Prestwich. (Jan. 2019). *What to Expect When ETH's Expecting*. Accessed: Aug. 1, 2019. [Online]. Available: https://hackernoon.com/what-to-expect-when-eths-expecting-80cb4951afcd
[96] LinkTime, Youtube. *Justin Drake-Ethereum, Sharding*. Accessed: Sep. 1, 2019. [Online]. Available: https://www.youtube.com/watch?v=J4rylD6w2S4
[97] (2017). *Randao: Verifiable Random Number Generation*. [Online]. Available: https://www.randao.org/whitepaper/Randao_v0.85_en.pdf
[98] D. Boneh, J. Bonneau, B. Bünz, and B. Fisch, "Verifiable delay functions," in *Advances in Cryptology—CRYPTO*, H. Shacham and A. Boldyreva, Eds. Cham, Switzerland: Springer, 2018, pp. 757–788.
[99] JustinDrake. (2018). *Minimal VDF Randomness Beacon*. Accessed: Aug. 1, 2019. [Online]. Available: https://ethresear.ch/t/minimal-vdf-randomness-beacon/3566
[100] B. Wesolowski, "Efficient verifiable delay functions," Cryptol. ePrint Arch., Cham, Switzerland, Tech. Rep. 2018/623, 2018. [Online]. Available: https://eprint.iacr.org/2018/623
[101] K. Pietrzak, "Simple verifiable delay functions," Cryptol. ePrint Arch., Cham, Switzerland, Tech. Rep. 2018/627, 2018. [Online]. Available: https://eprint.iacr.org/2018/627
[102] L. D. Feo, S. Masson, C. Petit, and A. Sanso, "Verifiable delay functions from supersingular isogenies and pairings," Cryptol. ePrint Arch., Cham, Switzerland, Tech. Rep. 2019/166, 2019. [Online]. Available: https://eprint.iacr.org/2019/166
[103] V. Buterin. (2015). *The Problem of Censorship*. Accessed: Aug. 1, 2019. [Online]. Available: https://blog.ethereum.org/2015/06/06/the-problem-of-censorship/
[104] A. Chepurnoy, "Interactive proof-of-stake," Jan. 2016, *arXiv:1601.00275*. [Online]. Available: https://arxiv.org/abs/1601.00275
[105] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, "Eclipse attacks on Bitcoin's peer-to-peer network," in *Proc. 24th USENIX Secur. Symp. (USENIX Secur.)* Washington, DC, USA: USENIX Association, Aug. 2015, pp. 129–144. [Online]. Available: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/heilman
[106] N. Alon, H. Kaplan, M. Krivelevich, D. Malkhi, and J. Stern, "Scalable secure storage when half the system is faulty," in *Automata, Languages and Programming*, U. Montanari, J. D. P. Rolim, and E. Welzl, Eds. Berlin, Germany: Springer, 2000, pp. 576–587.
[107] N. Alon, H. Kaplan, M. Krivelevich, D. Malkhi, and J. Stern, "Addendum to 'scalable secure storage when half the systemis faulty' [inform. comput. 174 (2) (2002) 203–213]," *Inf. Comput.*, vol. 205, no. 7, pp. 1114–1116, 2007.
[108] P. Maymounkov and D. Mazières, "Kademlia: A peer-to-peer information system based on the XOR metric," in *Proc. Revised Papers 1st Int. Workshop Peer-to-Peer Syst. (IPTPS)*, London, U.K.: Springer-Verlag, 2002, pp. 53–65. [Online]. Available: http://dl.acm.org/citation.cfm?id=646334.687801
[109] B. Wesolowski, "Efficient verifiable delay functions," in *Advances in Cryptology—EUROCRYPT*, Y. Ishai and V. Rijmen, Eds. Cham, Switzerland: Springer, 2019, pp. 379–407.
[110] Block.One. (Mar. 2018). *EOS.IO Technical White Paper V2*. [Online]. Available: https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md
[111] (2018). *NEO White Paper*. [Online]. Available: http://docs.neo.org/en-us/
[112] K. Nikitin, E. Kokoris-Kogias, P. Jovanovic, N. Gailly, L. Gasser, I. Khoffi, J. Cappos, and B. Ford, "CHAINIAC: Proactive software-update transparency via collectively signed skipchains and verified builds," in *Proc. 26th USENIX Secur. Symp. (USENIX Secur.)* Vancouver, BC, Canada: USENIX Association, Aug. 2017, pp. 1271–1287. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technicalsessions/presentation/nikitin
[113] A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, A. Poelstra, J. Timón, and P. Wuille. (2014). *Enabling Blockchain Innovations With Pegged Sidechains*. [Online]. Available: http://www.opensciencereview.com/papers/123/enablingblockchain-innovations-with-pegged-sidechains
[114] E. Regnath and S. Steinhorst, "LeapChain: Efficient blockchain verification for embedded IoT," in *Proc. Int. Conf. Comput.-Aided Design (ICCAD)*. New York, NY, USA: ACM, 2018, pp. 74:1–74:8, doi: 10.1145/3240765.3240820.
[115] A. Kiayias, A. Miller, and D. Zindros, "Non-interactive proofs of proof-of-work," *IACR Cryptol. ePrint Arch.*, vol. 2017, no. 963, pp. 1–42, 2017.
[116] A. Kiayias, N. Lamprou, and A.-P. Stouka, "Proofs of proofs of work with sublinear complexity," in *Financial Cryptography and Data Security*, J. Clark, S. Meiklejohn, P. Y. Ryan, D. Wallach, M. Brenner, and K. Rohloff, Eds. Berlin, Germany: Springer, 2016, pp. 61–78.
[117] B. Awerbuch and C. Scheideler, "Towards a scalable and robust DHT," *Theory Comput. Syst.*, vol. 45, no. 2, pp. 234–260, Aug. 2009, doi: 10.1007/s00224-008-9099-9.
[118] S. Sen and M. J. Freedman, "Commensal cuckoo: Secure group partitioning for large-scale services," *SIGOPS Oper. Syst. Rev.*, vol. 46, no. 1, p. 33, Feb. 2012, doi: 10.1145/2146382.2146389.
[119] W. Jiaming. (Jan. 2019). *Monoxide: A Solid Solution to Breaking the Blockchain Trilemma. [Blog] Notes of Decentralized Digital World*. Accessed: Aug. 1, 2019. [Online]. Available: https://zhuanlan.zhihu.com/p/56065714
[120] B-Wikipedia. (2019). *Non-Specialized Hardware Comparison*. [Online]. Available: https://en.bitcoin.it/wiki/Non-specialized_hardware_comparison
[121] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *J. Soc. Ind. Appl. Math.*, vol. 8, no. 2, pp. 300–304, Jun. 1960.
[122] W. Lim. (2018). *What are the Ethereum Disk Space Needs?* Accessed: Aug. 1, 2019. [Online]. Available: https://ethereum.stackexchange.com/questions/143/what-are-the-ethereum-disk-space-needs?noredirect=1&lq=1
[123] H. Chen and Y. Wang, "SSChain: A full sharding protocol for public blockchain without data migration overhead," *Pervasive Mobile Comput.*, vol. 59, Oct. 2019, Art. no. 101055. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1574119218306370
[124] M. Al-Bassam, A. Sonnino, and V. Buterin, "Fraud and data availability proofs: Maximising light client security and scaling blockchains with dishonest majorities," *CoRR*, vol. abs/1809.09044, pp. 1–34, Sep. 2018. [Online]. Available: http://arxiv.org/abs/1809.09044

GUANGSHENG YU received the B.Sc. and M.Sc. degrees from the University of New South Wales, Sydney, Australia, in 2015. He is currently pursuing the Ph.D. degree with the Global Big Data Technologies Centre, Faculty of Engineering and Information Technology, University of Technology, Sydney. His main research interests include blockchain consensus algorithms, scaling blockchains, privacy in blockchains, and the IoT applications with blockchains.

XU WANG received the B.E. degree in computer science from the Beijing University of Posts and Telecommunications, Beijing, China, in 2010, and the Ph.D. degree from Beijing Information Science and Technology University, Beijing, in 2019. His main research interests include blockchain, cyber security, complex networks, social networks, and network dynamics.

KAN YU received the B.Sc. degree from the Beijing University of Posts and Telecommunications, China, in 2005, the M.Sc. degree from the Chalmers University of Technology, Sweden, in 2010, and the Ph.D. degree from Malardalen University, Sweden, in 2014. He was a Visiting Scholar with The University of Sydney, in 2015. He was with the Huawei Beijing Research Centre and Huawei Australia, in 2007 and 2016, respectively. He is currently a Lecturer in Internet-of-Things (IoT) with La Trobe University. His current research interests include applying blockchain to the IoT, the industrial IoT, smart cities, and smart agriculture.

WEI NI (Senior Member, IEEE) received the B.E. and Ph.D. degrees in electronic engineering from Fudan University, Shanghai, China, in 2000 and 2005, respectively. He is currently a Team Leader with CSIRO, Sydney, Australia, and an Adjunct Professor with the University of Technology Sydney. He was a Postdoctoral Research Fellow with Shanghai Jiao Tong University, and the Deputy Project Manager of the R&I Center, Bell Labs, and Alcatel/Alcatel-Lucent, from 2005 to 2008. He was also a Senior Researcher in devices research and development with Nokia, from 2008 to 2009. His research interests include stochastic optimization, game theory, and graph theory and their applications to network and security.

J. ANDREW ZHANG (Senior Member, IEEE) received the B.Sc. degree from Xi'an Jiaotong University, China, in 1996, the M.Sc. degree from the Nanjing University of Posts and Telecommunications, China, in 1999, and the Ph.D. degree from Australian National University, in 2004. He is currently an Associate Professor with the School of Electrical and Data Engineering, University of Technology Sydney, Australia. He has published over 150 articles in leading international journals and conference proceedings. His research interests include areas of signal processing for wireless communications and sensing, and autonomous vehicular networks. He has won five best paper awards for his work.

REN PING LIU (Senior Member, IEEE) has supervised over 30 Ph.D. degree students. He is currently a Professor with the School of Computing and Communications, University of Technology Sydney, where he leads Network Security Lab. He is also a member of the Global Big Data Technologies Centre. Prior to that, he was a Principal Scientist with CSIRO, where he led wireless networking research activities. He specializes in protocol design and modeling. He has delivered networking solutions to a number of government agencies and industry customers. He has over 100 research publications. His research interests include Markov analysis and QoS scheduling in WLAN, VANET, the IoT, LTE, 5G, SDN, and network security. He was the winner of the Australian Engineering Innovation Award and the CSIRO Chairman Medal.