Benzene: Scaling Blockchain With Cooperation-Based Sharding

Zhongteng Cai, Junyuan Liang, Wuhui Chen, *Member, IEEE*, Zicong Hong, Hong-Ning Dai, *Senior Member, IEEE*, Jianting Zhang, and Zibin Zheng, *Fellow, IEEE*

Abstract—Sharding has been considered as a prominent approach to enhance the limited performance of blockchain. However, most sharding systems leverage a non-cooperative design, which lowers the fault tolerance resilience due to the decreased mining power as the consensus execution is limited to each separated shard. To this end, we present Benzene, a novel sharding system that enhances the performance by cooperation-based sharding while defending the per-shard security. First, we establish a double-chain architecture for function decoupling. This architecture separates transaction-recording functions from consensus-execution functions, thereby enabling the cross-shard cooperation during consensus execution while preserving the concurrency nature of sharding. Second, we design a cross-shard block verification mechanism leveraging Trusted Execution Environment (TEE), via which miners can verify blocks from other shards during the cooperation process with the minimized overheads. Finally, we design a voting-based consensus protocol for cross-shard cooperation. Transactions in each shard are confirmed by all shards that simultaneously cast votes, consequently achieving an enhanced fault tolerance and lowering the confirmation latency. We implement Benzene and conduct both prototype experiments and large-scale simulations to evaluate the performance of Benzene. Results show that Benzene achieves superior performance than existing sharding/non-sharding blockchain protocols. In particular, Benzene achieves a linearly-improved throughput with the increased number of shards (e.g., 32,370 transactions per second with 50 shards) and maintains a lower confirmation latency than Bitcoin (with more than 50 shards). Meanwhile, Benzene maintains a fixed fault tolerance at 1/3 even with the increased number of shards.

Index Terms—Blockchain, sharding, scalability, function decoupling, consensus algorithm

1 Introduction

As a decentralized ledger storing historical transactions, blockchain can be regarded as a massive decentralized database. Blockchain has been considered as a disruptive technology reshaping industrial and business sectors due to its decentralization, transparency, traceability, and immutability [1], [2]. However, blockchain systems such as Bitcoin and Ethereum have poor performance in terms of low transaction throughput and high latency, compared with conventional centralized payment systems. For example, Bitcoin can process 7 Transactions Per Second (TPS) with a 10-minute latency while centralized payment systems such as Visa can handle thousands of transactions per second with real-time confirmation [3]. The poor performance hinders blockchain systems from a wide adoption in practice. There have been a myriad of attempts for scaling up blockchain systems [4], [5], [6], [7], [8], [9]. One of the most popular and practical approaches to enhance blockchains is *sharding* [10]. The basic idea of sharding is to divide the entire network into different subsets (i.e., shards). Consequently, workloads of the entire system, including computation, communication, and storage, are distributed among shards to be executed in parallel. As a result, the throughput of a blockchain-sharding system can be significantly improved with the increased number of shards [11]. According to consensus algorithms being adopted, sharding approaches can be further categorized into Byzantine-Fault-Tolerance (BFT) shards (e.g., OmniLedger [12] and RapidChain [13]) and Proof-of-Work (PoW) shards (e.g., Monoxide [14]).

1.1 Motivation

Limitations of Existing Sharding Schemes. Most sharding approaches mainly adopt a *non-cooperative* design, in which an independent consensus reaches in each shard without reliance on other shards. As a result, participants of the consensus protocol are only limited to nodes belonging to a

- *Zhongteng Cai, Junyuan Liang, and Wuhui Chen are with the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou 510006, China.*
  E-mail: {caizht3, liangjy53}@mail2.sysu.edu.cn, chenwuh@mail.sysu.edu.cn.
- *Zicong Hong is with Hong Kong Polytechnic University, Hong Kong SAR, China. E-mail: zicong.hong@connect.polyu.hk.*
- *Hong-Ning Dai is with the Department of Computer Science, Hong Kong Baptist University, Hong Kong. E-mail: hndai@ieee.org.*
- *Jianting Zhang is with the Department of Computer Science, Purdue University, West Lafayette, IN 47907 USA. E-mail: zhan4674@purdue.edu.*
- *Zibin Zheng is with the School of Software Engineering, Sun Yat-sen University, Zhuhai, Guangdong Province 519082, China. E-mail: zhzibin@mail.sysu.edu.cn.*

Manuscript received 21 January 2022; revised 13 October 2022; accepted 27 November 2022. Date of publication 13 December 2022; date of current version 27 December 2022.
The work was supported in part by the National Key Research and Development Plan under Grant 2021YFB2700302, in part by the National Natural Science Foundation of China under Grant 62172453, in part by the National Natural Science Foundation of Guangdong province under Grants 2022A1515010154, 6142006200403, and XM2021XT1084, in part by the Major Key Project of PCL under Grant PCL2021A06, in part by the Program for Guangdong Introducing Innovative and Entrepreneurial Teams under Grant 2017ZT07X355, and in part by the Pearl River Talent Recruitment Program under Grant 2019QN01X130.
(Corresponding author: Wuhui Chen.)
Recommended for acceptance by R. Tolosana.
Digital Object Identifier no. 10.1109/TPDS.2022.3227198

1045-9219 © 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: INDIAN INST OF INFO TECH AND MANAGEMENT. Downloaded on April 15,2024 at 07:10:20 UTC from IEEE Xplore. Restrictions apply.

single shard[1]. Although non-cooperative sharding schemes relief workloads imposing on each node compared with non-sharding systems, they also confront critical security concerns. First, non-cooperative sharding leads to a lower fault tolerance due to the separation of honest participants. Compared with attacking the entire non-sharding system, it is easier for an adversary to attack a single shard in a sharding system (also known as *1% attack* [15]). Consequently, the fault tolerance resilience of a sharding system is lower than that of a non-sharding system when adopting the same consensus algorithm. With respect to BFT-based sharding schemes, to ensure that each shard can satisfy the intra-shard fault tolerance with high probability (e.g., 99%), the sharding system is subject to a lower total fault tolerance. For instance, OmniLedger [12] has 33% intra-shard fault tolerance and 25% total fault tolerance. Regarding PoW-based sharding schemes, since honest mining power is divided into isolated shards, the adversary is allowed to conduct 1% attack by focusing its mining power on a single shard [14]. As proved in Theorem 1 (in Section 5.1.1), the system fault tolerance decreases from 1/2 in non-sharding Bitcoin to $1/(s+1)$ in the non-cooperative sharding system, where s represents the number of shards. The root cause of the decreased fault tolerance resilience of non-cooperative sharding schemes lies in the separation of honest participants, thereby leading to the decreased mining power.

Second, security challenges further hinder the full optimization of the system performance. For example, the BFT-based sharding requires the reconfiguration of periodic epochs to prevent a single shard from being captured by a slowly adaptive adversary [12]; this demands extra time costs and in turn weakens the system performance [13]. Moreover, blocks in a PoW-based sharding system need to wait for more sequential blocks compared with non-sharding systems, since the separation of mining power leads to a higher probability for a single block being orphaned.

To tackle these challenges, we investigate a *cooperation-based sharding*. The basic idea of cooperation-based sharding is to let different shards cooperate with each other to confirm transactions while preserving parallelism. In other words, each shard reaches a consensus with the participation of mining power from other shards. Such cooperation increases the number of participants in the consensus, thereby enhancing the per-shard security.

Challenges in Cooperation-Based Sharding. Realizing the cooperation-based sharding, however, poses the following non-trivial challenges. (1) *How to design a cooperation-based sharding architecture to realize cross-shard cooperation while preserving parallelism?* The architecture in previous non-cooperative sharding systems cannot efficiently support the cooperation among shards. The cooperation-based sharding demands a novel design, which can enable efficient cross-shard cooperation while preserving parallelism. (2) *How to conduct cross-shard verification with low per-node overheads?* To enable cooperation among shards, nodes need to verify blocks from other shards. A trivial approach is to store all historical transactions of other shards for block verification, thereby causing considerable storage overheads. Such overheads need to be reduced to achieve a scalable sharding solution. (3) *How to design a cooperation-based consensus protocol to assure the per-shard security?* The cooperation-based sharding needs to design a consensus protocol utilizing the mining power from multiple shards while not compromising the system performance.

1. Although different shards may process cross-shard transactions (such as [13]), they do not cooperate in confirming blocks of each individual shard.

1.2 Main Results and Contributions

Our Solutions. To tackle these challenges, we present a novel scalable sharding system, namely *Benzene*, analogous to the planar ring structure of the chemical "Benzene". Our Benzene realizes the cross-shard cooperation with the following novelties. (1) *Benzene decouples system functions to enable cooperation while preserving parallelism.* Previous sharding systems [12], [13], [14] couple the functions of both transaction recording and consensus execution together. Differently, in Benzene, functions of transaction recording and consensus execution are separated from each other. Transaction recording is conducted by each shard independently, and consensus execution is conducted by all shards in cooperation. Moreover, we decouple the blockchain structure to support different functions. In Benzene, we design a novel double-chain architecture, which consists of i) *proposer chains* assigned for independent transaction recording in each shard and ii) *vote chains* assigned for cooperation-based consensus. Proposer chains record transactions, while vote chains are mainly composed of lightweight block headers. Such a design enables lightweight cooperation while preserving the concurrency nature of sharding. (2) *Benzene designs cross-shard block-verification mechanism based on Trusted Execution Environment (TEE).* TEEs [16], [17] can be deployed in each shard to provide cross-shard block-verification services. Nodes can directly verify validation proofs provided by TEEs rather than verifying original proposer blocks (i.e., blocks in proposer chains) from other shards. It is worth mentioning that nodes in the blockchain network do not have to trust all TEEs as some of them might be compromised by the adversary. Conversely, honest nodes can check the validity of validation proofs so as to improve the reliability. (3) *Benzene designs a cooperation-based consensus protocol based on cross-shard voting.* Proposer blocks that have been authenticated by cross-shard verification are voted by vote blocks (i.e., blocks in vote chains), which are generated in all shards. The proposer block with the most number of votes in each shard will be considered as the consensus result. Such a protocol introduces more mining power into a single shard, thereby enhancing the per-shard security. Besides, cross-shard transactions are confirmed in related shards sequentially. They are first confirmed in the originated shard, sent to the targeted shard along with related validation proofs, packaged again in a proposer block and finally get confirmed in the targeted shard. Forks may occur once a cross-shard transaction is aborted in the originated shard to preserve atomicity, but such a situation occurs quite rarely when the transaction is embedded deep enough in the originated shard.

Main Results. We implement Benzene and conduct both intensive prototype experiments and simulations to evaluate its performance with a comparison with the state-of-the-art sharding/non-sharding schemes. For a high-level

Authorized licensed use limited to: INDIAN INST OF INFO TECH AND MANAGEMENT. Downloaded on April 15,2024 at 07:10:20 UTC from IEEE Xplore. Restrictions apply.

TABLE 1
Comparison of Benzene With Other Sharding/Non-Sharding Blockchain Protocols

Protocols	Throughput (TPS)	Confirmation Interval	Fault Tolerance	Storage	Scalability	Function Decoupling
Traditional non-sharding (Bitcoin)	2,780	15	$1/2$	X	✗	✗
Voting-based non-sharding (Prism)	6,170	3	$1/2$	X	✗	✓
Non-cooperative sharding	9,330	89	$1/(s+1)$	X/s	✓	✗
Monoxide	9,260	15	$1/2$	X	✓	✗
Benzene (this paper)	32,370	13	$1/3$	X/s	✓	✓

# of nodes = 2,000; Fault tolerance = 1/4; # of shards = s; Total storage overhead = X.

presentation, we mainly categorize existing schemes into 1) traditional non-sharding (such as Bitcoin), 2) voting-based non-sharding (such as Prism [18]), 3) non-cooperative sharding and 4) Monoxide [14]. Table 1 shows a comparison of Benzene with these representative approaches. Benzene achieves 32,370 TPS and confirms each block with 13 sequential blocks when there are 50 shards (details on the experimental results to be given in Section 6). Moreover, Benzene assures a fixed fault tolerance of $1/3$ while distributing the storage overheads among all shards. By contrast, other approaches cannot achieve a comparable performance to Benzene. The major reason is explained as follows. Although the voting-based non-sharding achieves a higher throughput and lower confirmation latency than Bitcoin-like systems (traditional non-sharding systems) through function decoupling, it imposes the same computation and storage overheads on each node, thereby leading to poor scalability. Meanwhile, although the non-cooperative sharding scales up traditional non-sharding systems by dividing the computation and storage overheads, it is subject to a longer confirmation interval and a decreased fault tolerance due to the separation of honest mining power. Monoxide introduces Chu-ko-nu mining, which allows miners to mine blocks for multiple shards, to protect single shards. However, it requires miners to store and verify blocks of other shards, thereby increasing per-node overhead. Differing from the above approaches, Benzene combines both sharding and function decoupling technologies, and designs lighweight cooperation protocol, to implement cooperation-based sharding, consequently achieving a considerable improvement in terms of the throughput and preserving a fixed fault tolerance with the increment of shards.

Contributions. The main contributions of this paper are summarized as follows:

- *Double-Chain Sharding Architecture.* We present the double-chain architecture to enable efficient cross-shard cooperation. We separate transaction-recording from consensus-execution so as to execute them at different blockchains. Such a design enables cross-shard cooperation in Benzene without affecting independent transaction recording processes in each shard.
- *TEE-Assisted Cross-Shard Verification.* We leverage TEEs to provide cross-shard block-verification services, thereby relieving the communication and storage overheads required by cross-shard cooperation. We reduce the per-node overheads without compromising security and decentralization.
- *Cooperation-Based Consensus.* We present a cooperation-based consensus protocol to assure the per-shard security and enhance the system performance. In our protocol, transactions in each shard are confirmed by vote blocks which are generated in parallel among all shards. This design enhances the fault tolerance of a single shard and reduces the confirmation latency.
- *Experiment Evaluation.* We conduct both prototype experiments and simulations to evaluate the performance of Benzene. Our results show that the throughput of Benzene is linearly improved with the number of shards increasing to 200 and achieves 32,370 TPS with 50 shards. Moreover, Benzene achieves a lower confirmation latency than the traditional non-sharding system with a fixed fault tolerance of $1/3$, when there are more than 50 shards.

The rest part of this paper is organized as follows. In Section 2, we introduce previous studies related with this paper. Section 3 presents system model and adversary model of our Benzene. Section 4 describes the system design of Benzene in detail. Security analysis and performance analysis of our system are given in Section 5. Experimental setup and evaluation results are proposed in Section 6. Finally, we conclude our work in Section 7.

2 Related Work

2.1 Blockchain-Sharding System

A number of blockchain-sharding systems have been proposed and investigated [19], [20], [21], [22], [23], [24], [25]. In particular, Elastico [26] partitions the network into several committees to process transactions in parallel while relying on a final committee to agree on a global result. However, since each committee in Elastico uses a non-scalable PBFT [27] consensus as its intra-committee consensus, it limits a committee in a small scale and also leads to a high failure probability [12]. OmniLedger [12] designs a scalable BFT-based consensus algorithm, namely ByzCoinX, to increase the size of each shard, consequently reducing the failure probability. However, OmniLedger is still subject to 33% intra-shard fault tolerance and 25% total fault tolerance, which is the same as Elastico [13]. To increase the fault tolerance, RapidChain [13] runs an intra-shard BFT-based consensus with 50% fault tolerance in a synchronous network though such an approach is only suitable for a small-scale shard. By contrast, Benzene can guarantee the high node scalability with PoW-based consensus [28]. Dang et al. [29] apply sharding to permissioned

Authorized licensed use limited to: INDIAN INST OF INFO TECH AND MANAGEMENT. Downloaded on April 15,2024 at 07:10:20 UTC from IEEE Xplore. Restrictions apply.

blockchain system, therefore achieves a high performance. Specifically, it assumes that each node has a trusted hardware for maintaining logs of consensus messages, therefore ensures that BFT protocols can achieve higher fault tolerance to avoid a single shard from being compromised. By contrast, Benzene support permissionless environment and do not require every miner to have a trusted hardware.

Some recent works concentrate on optimizing blockchain sharding systems in specific application scenarios. For instance, Chainspace [20] and Tao et al. [21] focus on supporting smart contracts upon the sharding infrastructure, while the latter one explores methods to minimize cross-shard overheads. CoSplit [24] maximizes the degree of parallelization when transactions in a sharding system manipulate on the same states of smart contracts. Ethereum proposed a Danksharding scheme based on Proof-of-Stake consensus, in which there are no disjoint shards. Validators check the availability of the same block through Data Availability Sampling, so they do not have to download the whole contents of the block to check its availability. In comparison, validators in Benzene are still partitioned into disjoint shards and verify the whole contents of blocks, but can still be adopted in a system, which mainly concentrates on checking the availability of the data.

However, most approaches mentioned above are non-cooperative sharding, in which the consensus in each shard is reached independently by disjoint sets of nodes. Although Monoxide [14] proposes Chu-ko-nu Mining based on PoW mechanism to allow nodes to participate in the consensus process of other shards, it requires more per-node storage overheads or centralized nodes, to guarantee its security. By contrast, our Benzene imposes fewer overheads and meanwhile does not rely on any centralized nodes to preserve security.

2.2 TEE in Blockchain

As an isolated environment in processor, a TEE can protect the privacy and ensure the integrity of both data and codes inside [30]. Typical examples of TEEs include Intel SGX [31], ARM Trustzone [32], etc. Numerous studies have introduced TEEs into the blockchain system. The functionalities of TEEs in blockchain systems include acting as random sources and supporting privacy-preserving applications, etc. Take BITE [33] an example, in which TEEs verify transactions for light clients without revealing privacy. Meanwhile, Dang et al. [29] utilize Intel SGX in a BFT-based sharding system to serve for shard formation and consensus. Teechain [34] leverages TEEs to eliminate misbehaving parties and establish a more performant payment channel network. Other proposals such as [35], [36], [37] leverage TEEs to support the execution of smart contracts. For instance, Ekiden [35] proposes an off-chain contract execution method, through which smart contracts are executed in TEEs with privacy protection. It is worth mentioning that TEEs in our Benzene are responsible for verifying blocks.

Recent studies have investigated approaches to address challenges when incorporating TEEs into blockchains [34], [38], [39]. Some attacks targeting at TEEs, e.g., rollback attacks [40], may threaten the security guarantees of TEE-blockchain systems [34]. Similar to TEE-based contract execution engines, Benzene needs to avoid security challenges like rollback attacks. In particular, Benzene designs the appropriate TEE deployment and blockchain-based supervision to protect TEE security. As a result, Benzene does not require participants to trust all TEEs, thereby eliminating the cost of preserving consistency among multiple TEEs.

3 System and Adversary Model

3.1 System Model

We assume an open peer-to-peer network that allows all nodes to join or leave at any time. The entire network is divided into s shards. Each joining node that initially generates an address is randomly assigned to a shard. The blockchain maintains *accounts* owned by users. Each account records some states, like the balance value. Each account belongs to a specific shard. Users can generate transactions to manipulate these accounts. We denote a transaction by $Tx\langle S, t, D, n\rangle$, which represents the nth transaction generated by source account S conducting operations t upon destination account D. The operations can be token transfer, state manipulation, etc. Assume that S belongs to Shard A. If account D also belongs to Shard A, then such a transaction is a *local transaction*; otherwise, it is a *cross-shard transaction*. Some of the nodes in the network are equipped with TEE hardware.

3.2 Adversary Model

Adversaries can initiate several typical attacks. Consider an adversary who occupies a specific fraction of the total mining power [41]. The adversary can decide what to do with the information received from the network [18] and how to make use of its own mining power (e.g., focusing its mining power on a specific shard to conduct 1% attack [14]). Moreover, the adversary can deploy TEEs on its nodes and offer verification services to miners in the same shard. Such TEEs can be correctly initialized. But the adversary may decide what messages received by TEEs, or restart TEEs and provide correctly-encrypted yet stale states (i.e., conducting rollback attacks) to disturb its execution results [39]. In conclusion, the adversary can mislead TEEs to generate execution results with states deviated from the newest consensus results.

4 Cooperation-Based Sharding System

4.1 System Overview

Benzene is a novel blockchain sharding system with a cooperation-based consensus protocol. Fig. 1 depicts key components of Benzene. Benzene is composed of a double-chain architecture, in which each shard maintains two types of blockchains to conduct two decoupled functions: 1) proposer chains record transactions and 2) vote chains conduct cooperation-based consensus. To realize cross-shard cooperation, each shard conducts cross-shard verification to verify proposer blocks of other shards. Tasks of authenticating block contents are delegated to TEEs, thereby incurring fewer storage and computation overheads at ordinary nodes. The cooperation-based consensus is realized through cross-shard voting. Each shard confirms proposer blocks according to vote blocks generated in all shards. We first elaborate on key components of Benzene as follows.

Authorized licensed use limited to: INDIAN INST OF INFO TECH AND MANAGEMENT. Downloaded on April 15,2024 at 07:10:20 UTC from IEEE Xplore. Restrictions apply.

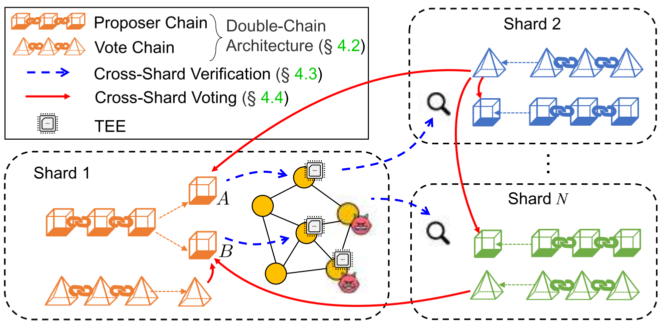

Proposer Chain
Vote Chain
Double-Chain Architecture (§ 4.2)
Cross-Shard Verification (§ 4.3)
Cross-Shard Voting (§ 4.4)
TEE
Shard 1
Shard 2
Shard N
A
B

Fig. 1. The architecture of Benzene.

TABLE 2
Blockchain Characteristics

PC^I/VC^I	The proposer chain / vote chain in the Ith shard.
PC^I_h/VC^I_h	The proposer block / vote block at height h on PC^I/VC^I.
$PC^I_{h,i}$	The ith proposer block at height h on PC^I (fork exists).
$\widehat{PC^I_h}/\widehat{PC^I}/\widehat{VC^I_h}/\widehat{VC^I}$	The header of the referred block or blockchain.
s	The total number of shards.

Double-Chain Architecture. Benzene enables cross-shard cooperation by function decoupling on top of the double-chain architecture. In particular, Benzene decouples the transaction-recording functions from the consensus-execution functions by *proposer chains* and *vote chains*, respectively. Each shard has one proposer chain and one vote chain. Proposer chains are composed of *proposer blocks* that record transactions related to the same shard. Vote chains are composed of *vote blocks*, which participate in cross-shard voting consensus protocol to confirm proposer blocks on proposer chains. We will further elaborate on the double-chain architecture in Section 4.2.

Cross-Shard Verification. To accomplish cooperation-based consensus, miners in a shard have to affirm the correctness of proposer blocks in other shards. Benzene designs a cross-shard verification mechanism integrated with TEEs deployed by nodes that are willing to provide authentication services. TEEs verify proposer blocks of the same shard and give validation proofs. Both headers of proposer blocks and validation proofs are broadcast among all shards. Miners in other shards conduct cross-shard verification through verifying proofs produced by TEEs before generating vote blocks. With the adoption of TEEs, miners in other shards are not necessary to verify block contents themselves but verify the validation proofs instead. Therefore, they do not need to store the complete historical transaction records of other shards so that storage overheads can be saved. Section 4.3 will describe the cross-shard verification mechanism in detail.

Cross-Shard Voting Consensus. The aim of cross-shard voting consensus is to confirm a unique proposer block in each shard according to vote blocks that have been mined in all shards. The cross-shard voting consensus consists of three phases: (1) *transaction-recording phase*, (2) *vote-generation phase*, and (3) *block-confirmation phase*. In the transaction-recording phase, proposer blocks recording transactions are generated by miners. More than one proposer blocks may be simultaneously generated in each shard (like Block A and Block B of Shard 1 in Fig. 1), thereby incurring forks, which will nevertheless be resolved by vote blocks later. TEEs verify newly-mined proposer blocks, and validation proofs generated by TEEs will be sent to the entire network, as depicted by blue dotted arrows in Fig. 1. In the vote-generation phase, miners verify validation proofs from all shards to check the validity of proposer blocks, without the need for downloading whole block contents, and mine vote blocks so as to vote for one proposer block in each shard, as depicted by red arrows in Fig. 1. Vote blocks are also broadcast across the entire network. In the block-confirmation phase, the proposer block containing the most number of votes within each shard is confirmed (e.g., the Block B in Shard 1). Each shard synchronizes vote chains from all shards to recognize the voting results at the same *height* (i.e., the distance from a block to the genesis block). A unanimous distributed ledger is thus generated out of the confirmed proposer block.

Cross-shard voting consensus enhances the fault tolerance of a single shard while preserving high performance. The enhanced fault tolerance mainly owes to the mining power brought by vote blocks into every single shard. Meanwhile, vote blocks are also generated in a parallel so as to greatly reduce the confirmation latency compared with Bitcoin that confirms a block with sequential blocks on a single chain [18], [42]. Section 4.4 will give a thorough description of cross-shard voting consensus protocol.

4.2 Double-Chain Architecture for Function Decoupling

Benzene separates functions of transaction recording and consensus execution that are coupled together in traditional blockchains like Bitcoin so to achieve cross-shard cooperation. Decoupled functions are conducted by a double-chain architecture. In the double-chain architecture, each shard maintains two blockchains: a proposer chain and a vote chain. A proposer chain records transactions in its shard concurrently while a vote chain cooperates with other vote chains to confirm blocks on the proposer chain. The proposer chain of Shard I is denoted by PC^I with PC^I_h being the block at height h. Similarly, VC^I denotes the vote chain in Shard I with VC^I_h being its hth block, where a term with a widehat represents the header of a block or a chain of block headers (e.g., $\widehat{VC^I_h}$ and $\widehat{VC^I}$). Some related characteristics are summarized in Table 2.

4.2.1 *Proposer Chain*

Being responsible for recording transactions in each shard, a proposer chain is composed of proposer blocks linked by back-hashes. After receiving and verifying transactions in the same shard, a miner can pack transactions in the form of a Merkle tree and find a nonce that fulfills the PoW target to mine a valid proposer block. The PoW target is shared among diverse shards. Multiple proposer blocks may exist in a shard at the same time similar to the occurrence of forks in Bitcoin though they will later be resolved by the cooperation-based consensus. We specify an unconfirmed proposer

Authorized licensed use limited to: INDIAN INST OF INFO TECH AND MANAGEMENT. Downloaded on April 15,2024 at 07:10:20 UTC from IEEE Xplore. Restrictions apply.

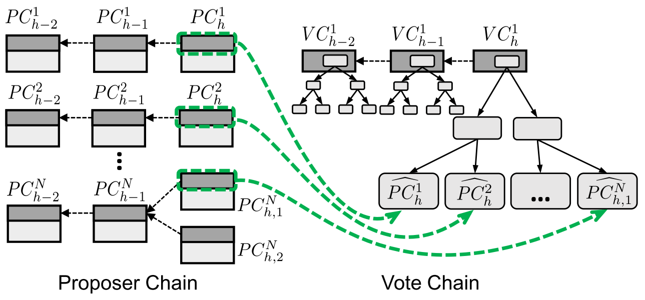


Fig. 2. Proposer chains and vote chain.

block as $PC^K_{h,i}$, with K being the shard index, h being the block height, and i being the index among proposer blocks with the same K and h. Take Fig. 2 as an example, in which two proposer blocks $PC^N_{h,1}$ and $PC^N_{h,2}$ (at the same height) are mined in Shard N. Proposer blocks are then verified by TEEs and their headers (e.g., $\widehat{PC^1_h}$ to $\widehat{PC^N_{h,2}}$) are broadcast to all remaining shards to receive votes from Shard 1 to Shard N. Each shard also maintains headers of the proposer chains of other shards to record consensus results.

4.2.2 Vote Chain

Vote chains enable cross-shard cooperation among all shards through voting upon proposer chains. A vote chain is composed of linked vote blocks, which are generated through the PoW process. A vote block represents a vote upon s headers of proposer blocks, each of which belongs to a unique shard among all s shards. Fig. 2 depicts that the latest vote block VC^1_h in Shard 1 casts votes upon headers of proposer blocks in all shards, i.e., from $\widehat{PC^1_h}$ of Shard 1 to $\widehat{PC^N_{h,1}}$ of Shard N. The vote block constructs a Merkle tree based on the hash values of headers of s proposer blocks. The Merkle root is included in the vote block header. If a fork occurs on a vote chain, it will be resolved according to the longest-chain-rule similar to Bitcoin [43], [44]. Although a single vote chain may be reversed by the attacker, it will have little effect on the proposer chain, since the proposer chain relies on the overall effects contributed by all vote chains. We will prove later in Section 5.2.1 that the overall effect of multiple vote chains can make each proposer chain have shorter confirmation latency compared with Bitcoin to achieve the same reversal probability when there are more than 40 shards. Vote blocks are broadcast to the whole network to help each shard acknowledge the consensus results. Since each vote block is negligible in size (i.e., a vote block is much smaller than a proposer block as in Section 5.2.3), it will not affect the scalability of the system. Each shard maintains all s vote chains to verify the latest vote blocks and make historical consensus results auditable.

4.3 Cross-Shard Verification Mechanism

In the cooperation-based architecture of Benzene, an honest miner has to guarantee the validation of proposer blocks in other shards before casting votes on them. In other words, a cross-shard verification is necessary. A trivial approach to achieve this goal is to store blockchain records of other shards and verify proposer block contents themselves while it causes considerable overheads since miners have to store all historical transactions and verify proposer blocks from all shards. To reduce both computation and storage overheads, we adopt TEEs to provide validation proofs for achieving the cross-shard block-verification.

4.3.1 Security Challenges of TEE

However, the direct incorporation of TEEs into blockchains also poses two challenges in assuring the correctness of the cross-shard verification: (C1) *The weak availability of TEEs* since a TEE can be crashed by its owner at any time; (C2) *The fragile guarantee of the latest blockchain state*. First, a TEE may suffer from the rollback attack due to the execution of the stale states [40], [45], consequently violating its execution results [39], [40]. Consider a case that a TEE has updated its state stored in the disk with successively confirmed proposer blocks. When the TEE is restarted by the adversary, the run-time memory is cleared and the TEE requires the latest state. The adversary then provides a stale state (though correctly sealed), thereafter making further verification results deviate from the current consensus results. Second, even for TEEs not being restarted by the adversary, they may also execute on an incorrect state. For instance, a TEE may be provided with an orphaned block so that it maintains an outdated or incorrect state, which is not consistent with the latest consensus result.

4.3.2 Design of Verification Mechanism

We design a cross-shard verification mechanism, which tackles the above challenges of TEEs. Our solutions can be summarized into the following aspects.

An Appropriate TEE Deployment to Address C1. We leverage Intel SGX [31] for the implementation of TEE. In particular, a miner deploys Intel SGX on its node. As an isolated container of codes, the *enclave* in Intel SGX is initialized with the blockchain states and verification codes. The key pair is generated for the further encryption process. Intel SGX ensures a correct initialization of given codes through remote attestation [38], [46], [47]. Both attestation results and public keys of TEEs are publicly available so that the execution of TEEs will not deviate from the pre-defined verification codes after verifying attestation results since the integrity of initial codes in TEEs is ensured (not tampered by the adversary). Meanwhile, all these TEEs can verify blocks independently with locally stored states, consequently allowing miners to obtain verification services from an arbitrary number of TEEs. Moreover, this design also guarantees the fault tolerance because of the following refinements. (a) Each TEE can be interchangeable (e.g., replacing a failed TEE) by adopting a stateless design. TEEs require the latest states of blockchain when verifying proposer blocks. Further, a proof-of-publication can ensure that the state has been confirmed by the consensus protocol [35]. (b) Miners can require the latest states of the related accounts from the targeted shard and then manually verify proposer block contents even when TEE services are not accessible.

Supervision-Based Verification to Tackle C2. The basic idea of supervision-based verification is to allow miners in other shards to distinguish suspicious validation proofs from

Authorized licensed use limited to: INDIAN INST OF INFO TECH AND MANAGEMENT. Downloaded on April 15,2024 at 07:10:20 UTC from IEEE Xplore. Restrictions apply.

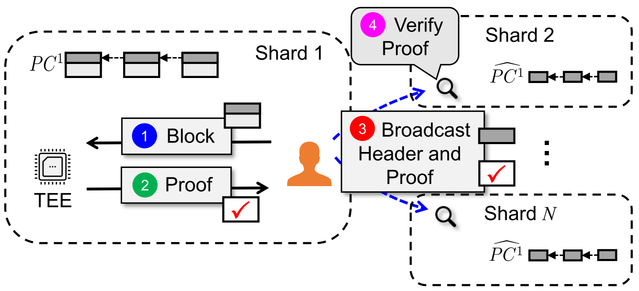


Fig. 3. TEE verification process.

correct ones even though TEEs cannot guarantee to always fetch the latest state. The process of supervision-based verification is depicted in Fig. 3. Each time a proposer block that is successfully mined will be forwarded to the TEE of the same shard for verification (Step ❶). The TEE checks whether a proposer block is valid according to both preloaded codes and locally maintained states. If such a block is valid, the TEE returns a digital signature as its validation proof (Step ❷). The miner then sends the header of its proposer block and the validation proof to all shards (Step ❸). Preloaded verification codes require the TEE to include the hash values of the latest proposer block headers in the validation proof. Since the header of the latest proposer block is broadcast and maintained by the whole network, all nodes can verify its hash value independently (Step ❹). If the hash value in the TEE signature is consistent with locally stored records, then miners will trust this block and vote for it. Otherwise, the block may be authenticated with the outdated or orphaned state and will consequently be rejected by honest miners.

4.4 Cross-Shard Voting Consensus Protocol

The cross-shard voting consensus protocol aims to confirm both local transactions and cross-shard transactions.

4.4.1 *Confirming Local Transactions*

The basic goal of consensus protocol is to decide one unique proposer block in each shard. For each Shard I ($I = 1, 2, \cdots, s$), the cooperation-based consensus decides a unique proposer block PC_h^I at the height h with given vote blocks VC_h^1, VC_h^2, $\cdots$, VC_h^s. Proposer block PC_h^I at height h in turn gives a unique order of transactions in Shard I. Algorithm 1 and Fig. 4 describes the three major phases of the consensus protocol.

Algorithm 1. Cooperation-Based Consensus Protocol

```
1)  Transaction-Recording
    a)  Miner in Shard $I$ generates $PC_{h,i}^I$ based on $PC_{h-1}^I$;
    b)  Sends $PC_{h,i}^I$ to a TEE deployed in Shard $I$;
    c)  The TEE returns a validation proof;
    d)  Broadcasts the proposer block header $\widehat{PC}_{h,i}^I$ and its
        validation proof to all shards.
2)  Vote-Generation
    a)  Miner receives sets of proposer block headers and
        their validation proofs from all shards;
    b)  Verifies each proposer block header $\widehat{PC}_{h,i}^K$ according
        to its validation proof and header of the proposer
        chain $\widehat{PC}^K$ ($K = 1, 2, \cdots, s$);
    c)  Picks a valid proposer block from each shard to
        mines $VC_h^I$;
    d)  Broadcasts $VC_h^I$ among the entire network.
3)  Block-Confirmation
    a)  Miner receives vote blocks from all shards;
    b)  Verifies vote block $VC_h^K$ with locally stored proposer
        block headers and vote chain $VC^K$ ($K = 1, 2, \cdots, s$);
    c)  Calculates how many votes are received by proposer
        blocks in each shard, and the one with the most num-
        ber of votes will be confirmed;
    d)  Miners and TEEs in each Shard $I$ execute all related
        transactions ordered by proposer block $PC_h^I$.
```

Transaction Recording. In this phase, miners package transactions, mine proposer blocks, send these proposer blocks to TEEs for verification, and finally send information needed for cross-shard verification to other shards. After receiving transactions related to the same shard, a miner in Shard I generates proposer block $PC_{h,i}^I$ and sends it to a TEE for verification. The TEE returns a validation proof if the block is valid. The validation proof also contains the hash values of the latest headers of the confirmed proposer block (e.g., $\widehat{PC}_{h-1}^I$) stored in the TEE to prove that states in the TEE have been updated to the latest version. Both the proposer block header $\widehat{PC}_{h,i}^I$ and its proof are propagated among the entire network. Multiple proposer blocks may be authenticated and broadcast simultaneously since more than one TEEs are working in parallel. Fig. 4a shows that two proposer blocks $PC_{h,1}^1$ and $PC_{h,2}^1$ are authenticated in Shard 1. Both of them are then to be possibly confirmed by vote blocks.

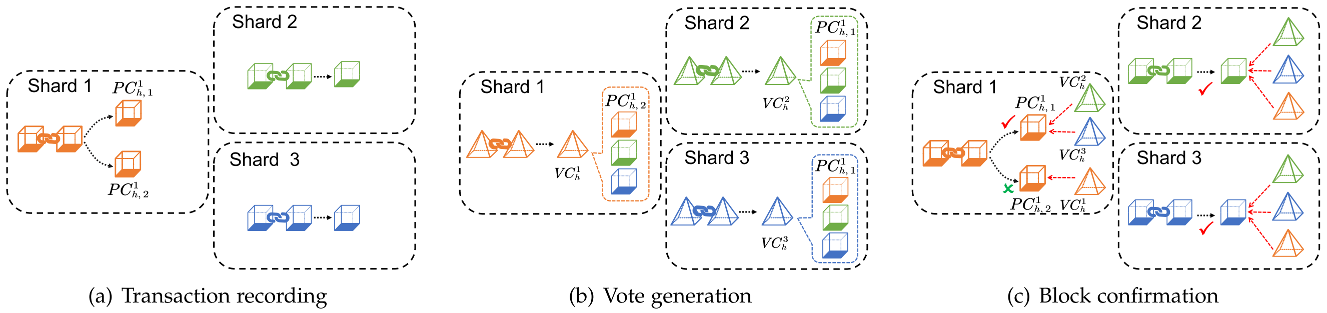


Fig. 4. The cooperation-based consensus protocol.

Authorized licensed use limited to: INDIAN INST OF INFO TECH AND MANAGEMENT. Downloaded on April 15,2024 at 07:10:20 UTC from IEEE Xplore. Restrictions apply.

Vote Generation. In this phase, miners verify proofs of proposer blocks from all shards and generate vote blocks to cast votes on these proposer blocks. After the transaction-recording phase, miners will receive headers of valid proposer blocks $\widehat{PC^K_{h,i}}$ $(K = 1, 2, \cdots, s)$ and their validation proofs from all shards. Each miner can independently verify $\widehat{PC^K_{h,i}}$ and its validation proof with the locally-stored headers of proposer chains and the locally-stored TEE key. Specifically, the miner will check whether the hash values of the latest confirmed proposer blocks which are included in the validation proof is consistent with its local records. Each miner picks one proposer block header from each shard. A set of chosen headers are formed and a new vote block VC^I_h is mined accordingly. Such a vote block casts votes upon totally s proposer block headers, each of which comes from a different shard. Consider Fig. 4b as an example, in which the vote block in Shard 1 (i.e., VC^1_h) votes for $\widehat{PC^1_{h,2}}$ while both VC^2_h in Shard 2 and VC^3_h in Shard 3 vote for $\widehat{PC^1_{h,1}}$ instead. Vote blocks will then be broadcast globally.

Block Confirmation. In this phase, miners receive vote blocks from all shards, and confirm the proposer block, which has gained the most number of votes. After receiving vote blocks VC^K_h $(K = 1, 2, \cdots, s)$ from each shard, miners verify it through locally-stored headers of proposer blocks and the vote chain VC^K. For example, miners check whether proposer blocks' headers included in the vote block VC^K_h are authenticated and whether the predecessor of VC^K_h is valid. Upon receiving headers of proposer blocks and vote blocks, every node can calculate how many votes are gained by each proposer block header; the number of votes is at most s. For the set of all proposer block headers in Shard K $(K = 1, 2, \cdots, s)$ at height h, the header with the most number of votes will be confirmed and appended to the proposer chain PC^K. For instance, as shown in Fig. 4c, $PC^1_{h,1}$ gets two votes while $PC^1_{h,2}$ gets only one, thereby $PC^1_{h,1}$ being confirmed and appended to PC^1. If multiple proposer blocks have the same number of votes, then the proposer block with the smallest hash value will be determined as the consensus result. Miners and TEEs can then update locally-stored ledgers by executing transactions ordered by the proposer block of the same shard.

Fork Resolution. Another issue is fork resolution. The confirmed proposer block at each height can always be unique if no fork occurs on vote chains. However, the occurrence of forks on vote chains can affect the consensus results. For example, in Fig. 5, a fork exists on the vote chain VC^2 and influences the consensus result on the proposer chain PC^1. $PC^1_{h,1}$ gains two votes: VC^1_h and $VC^2_{h,1}$. $PC^1_{h,2}$ also gains two votes: $VC^2_{h,2}$ and VC^3_h. We have to solve the fork on VC^2 and then decide which proposer block gains the most number of votes. We first separate the *first confirmation* from the *final confirmation*. The first confirmation is the status that vote blocks of all shards at the same height have been generated though a fork may exist on vote chains. Thus, a proposer block has the probability to be orphaned later even if it gains more votes than others at present. But the system will not halt. After the first-confirmation, miners will continue to generate new proposer blocks and vote blocks, even if forks may exist on vote chains. We adopt the longest-chain rule to address forks similar to Bitcoin and Prism [18]. For example, in Fig. 5, $VC^2_{h,1}$ is on the longest chain. As a result, $VC^2_{h,2}$ is discarded and

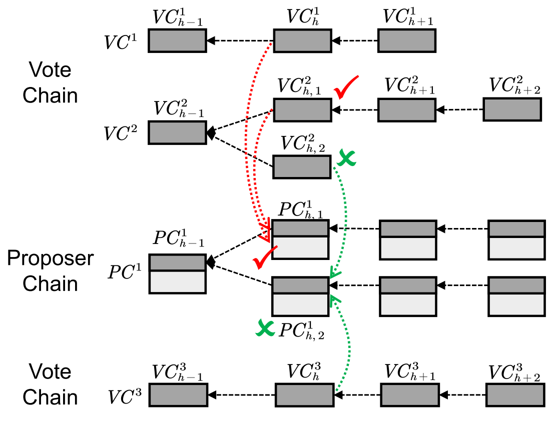


Fig. 5. Fork resolution.

$PC^1_{h,1}$ has the highest number of votes. We define that a proposer block is *final-confirmed* when each vote block at the same height is followed by at least z sequential blocks. The longest chain followed by z sequential blocks ensures that the vote block will not be reversed with a high probability, thereby in turn limiting the reversal probability of proposer chains. Transaction confirmation is also advised not to be earlier than the final confirmation of the related proposer block. The selection of z value will be given in Section 5.2.1.

4.4.2 Confirming Cross-Shard Transactions

In Benzene, the cross-shard transaction processing aims at achieving *eventual atomicity* with a two-phase confirmation. The confirmation of a cross-shard transaction can be seen as the execution of two phases: the first phase manipulates states in the source shard and the second phase manipulates states in the destination shard. Eventual atomicity indicates that such two phases will be finished by incentivized miners eventually and therefore atomicity is preserved [14].

Fig. 6 depicts the procedure of confirming cross-shard transactions. During the first phase, cross-shard transactions are recorded in a proposer block (e.g., PC^1_h in Fig. 6) in the source shard and are confirmed for the first time (Steps ❶-❷). After that, cross-shard transactions, along with their Merkle tree paths in the proposer block are sent to the destination shard (Step ❸). The miner in Shard 2 affirms that those cross-shard transactions have been collected in a confirmed proposer block and then collects them in a new proposer block (Step ❹). If such proposer block (e.g., PC^2_{h+n} in Fig. 6) is confirmed in the consensus protocol (Step ❺), then cross-shard transactions are also confirmed.

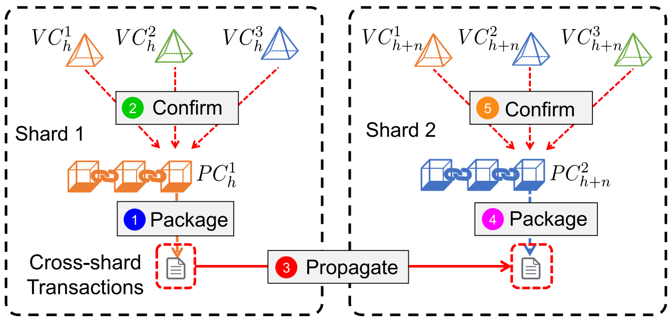


Fig. 6. Processing cross-shard transactions.

Authorized licensed use limited to: INDIAN INST OF INFO TECH AND MANAGEMENT. Downloaded on April 15,2024 at 07:10:20 UTC from IEEE Xplore. Restrictions apply.

Such two phases are accomplished asynchronously. We can optimistically expect that cross-shard transactions can finally be confirmed in the destination shard and therefore all two phases are finished, since all miners are motivated by incentives to package all transactions, which are valid but are not yet included on chain, similar to other PoW-based systems like Bitcoin [14], [43]. Even when the proposer block containing a cross-shard transaction is orphaned in the destination shard, such a cross-shard transaction may be packaged later by other proposer blocks and hence may be confirmed again afterwards, which still satisfies the Eventual Atomicity.

If a fork leads to the failure of the first phase (i.e., a corresponding proposer block recording the cross-shard transaction is orphaned), the second phase will also fail, which is accomplished by invalidating the previous block and creating a new fork. Miners can check whether a fork occurs in other shards and whether cross-shard transactions are still valid, since they can record block headers of other shards to check the longest chain and can verify whether TEE proofs which authenticated cross-shard transactions are still based on the valid states. Miners will rollback their ledger states, which can be accelerated by checkpoint blocks similar to [14]. Such a rollback rarely occurs as long as we require that a cross-shard transaction has to be embedded in the source shard deep enough before confirming it in the destination shard.

We can optimize the confirmation process of cross-shard transactions by processing them in batches. We design a new type of block called a *transaction block*. A transaction block packages transactions with the same source shard and the same destination shard. For instance, a transaction block may record cross-shard transactions from Shard A towards Shard B. Transaction blocks are not organized in a chain structure. Instead, transaction blocks are organized in a Merkle Tree and included in a proposer block. Transaction blocks can be verified by TEEs to accomplish cross-shard verification. Each time a proposer block is confirmed in the source shard, transaction blocks included in it will be sent to their destination shard, along with their paths in the Merkle Tree and validation proofs so as to accomplish the verification and confirmation process of cross-shard transactions.

5 System Analysis

5.1 Security Analysis

5.1.1 Fault Tolerance Analysis

We assume that an adversary intends to reverse the consensus results. In the non-cooperative sharding system, the adversary will attempt to mine a longer fork on the chain of a single shard. In Benzene, the adversary has to confirm another proposer block by mining longer vote chains among half of the shards. We prove that Benzene achieves a fixed fault tolerance while the non-cooperative sharding suffers from the decreased fault tolerance with the increased number of shards.

Theorem 1. *The fault tolerance of the non-cooperative sharding system is $\frac{1}{s+1}$ with s being the number of shards, while Benzene achieves $\frac{1}{3}$ with any number of shards.*

Proof. The system fault tolerance of the non-cooperative sharding system depends on the relative mining power of the adversary distributed on a single chain in a shard. Assume that the mining power controlled by the adversary M_A occupies β fraction of the total mining power M in the entire network, i.e., $M_A = \beta M$. Denoting the mining power of honest miners by M_H, we have $M_H = (1-\beta)M$. We then calculate the average mining power imposed on a chain by the adversary and honest miners with equations given as follows,

$$M_{\overline{H}} = \frac{M_H}{s} = \frac{(1-\beta)M}{s}, \tag{1a}$$

$$M_{\overline{A}} = \frac{M_A}{1} = \beta M. \tag{1b}$$

□

Let p be the probability that an honest miner mines the latest block on the chain of a specific shard, and q be the probability that the adversary mines it. On a specific chain, $\frac{q}{p}$ is equal to the proportion of $M_{\overline{A}}$ to $M_{\overline{H}}$. We then have:

$$\frac{q}{p} = \frac{M_{\overline{A}}}{M_{\overline{H}}} = \frac{\beta s}{1-\beta}. \tag{1c}$$

When $\frac{q}{p} > 1$, the attacker can always reverse the consensus results from any historical block in a period that is long enough. When $\frac{q}{p} \leq 1$, we get $\beta \leq \frac{1}{s+1}$, which apparently decreases with the increased number of shards.

By contrast, the system fault tolerance of Benzene depends on the relative mining power of the adversary distributed on vote chains. Denoting the mining power controlled by the adversary by M'_A, we have $M'_A = \beta M'$. Denoting the mining power of honest miners by M'_H, we have $M'_H = (1-\beta)M'$. Assume that the honest mining power is equally distributed among s vote chains, while malicious mining power is equally distributed among $s/2$ vote chains. On each vote chain targeted by the adversary, there is M'_H/s honest mining power and $\frac{M'_A}{s/2}$ malicious mining power. We can calculate the mining power proportion by following the same methods. Particularly, we have:

$$M'_{\overline{H}} = \frac{M'_H}{s} = \frac{(1-\beta)M'}{s}, \tag{2a}$$

$$M'_{\overline{A}} = \frac{M'_A}{s/2} = \frac{2\beta M'}{s}, \tag{2b}$$

$$\frac{q'}{p'} = \frac{M'_{\overline{A}}}{M'_{\overline{H}}} = \frac{2\beta}{1-\beta}. \tag{2c}$$

When $\frac{q'}{p'} \leq 1$, we have $\beta \leq \frac{1}{3}$, indicating that Benzene can tolerate at most $\frac{1}{3}$ adversarial mining power, which does not depend on the number of shards. □

5.1.2 TEE Security

We now prove that the integrity of the distributed ledger of Benzene is ensured even with attacks aiming at TEEs. Attackers can be any nodes in the network attempting to compromise any deployed TEE. We mainly consider that the adversary attempts to compromise the TEE deployed on its own node since it is the most possible way to successfully launch an attack. The owner of the targeted TEE can launch

Authorized licensed use limited to: INDIAN INST OF INFO TECH AND MANAGEMENT. Downloaded on April 15,2024 at 07:10:20 UTC from IEEE Xplore. Restrictions apply.

TABLE 3
Vote Block Confirmation Latency

Number of shards	20	30	40	50	100	200	500
Minimized z	17	15	14	13	11	9	8

two types of attacks: (A1) The adversary tries to crash the TEE or hold the communication messages to make verification proofs be inaccessible; (A2) The adversary conducts the rollback attack or sends invalid blocks to the TEE with an expectation that the TEE will generate a verification proof incompatible with the current consensus results.

To address (A1), miners can fetch verification results from other accessible TEEs as mentioned in Section 4.3.2. Meanwhile, consensus participants in other shards can also download historical transactions to verify proposer block contents. To tackle (A2), miners can pick out invalid proofs through verifying the included hash values of the latest states. After synchronizing vote chains and headers of proposer chains from all shards, miners can keep tracking the consensus results in all shards. Since the attacker cannot tamper with verification codes encapsulated in the enclave, the hash values of states that have been stored in the compromised TEE and been deviated from the consensus results will be discovered in the proof, as mentioned in Section 4.3.2. As a result, proposer blocks with invalid proofs will not be voted by honest participants. Thus, the distributed ledger will not be affected.

5.2 Performance Analysis

5.2.1 Confirmation Latency

In Benzene, forks on vote chains can lead to the orphaned proposer blocks. Forks can be resolved by the longest chain rule as mentioned in Section 4.4. To make the probability of a proposer block being orphaned in the future be as low as ϵ, we wait until all the vote blocks at the same height are followed by successive z blocks. The time period of z block intervals is referred to as the *confirmation latency*. We now measure the selection of z with the given proportion of adversarial mining power β and the number of shards s. Assuming $\beta = 0.25$, we can calculate the reversal probability P for the vote block that is z-deep according to the following equation as given by [43]:

$$P = 1 - \sum_{k=0}^{z} \frac{\lambda^k e^{-\lambda}}{k!}\left(1 - \left(\frac{q}{p}\right)^{(z-k)}\right). \tag{3}$$

The reversal probability of a proposer block when all related vote blocks in s shards are z-deep is as follows:

$$\epsilon = \sum_{i=\frac{s}{2}}^{s} \binom{s}{i} P^i (1-P)^{s-i}. \tag{4}$$

Table 3 presents the results of the minimized z with $\epsilon \leqslant 0.001$. We observe from Table 3 that z drops as s raises. The value of z in Bitcoin with the same parameters (i.e., $\beta = 0.25$, $\epsilon = 0.001$) is 15, which is larger than z in Benzene when $s > 40$. Although sharding weakens the security of each shard and in turn affects the confirmation latency, the cross-shard cooperation provides a comparable and even lower confirmation latency compared with non-sharding systems.

5.2.2 Bandwidth Consumption

The TEE verification reduces the network-bandwidth consumption in the cooperation-based protocol. Let PC^A be a proposer block in Shard A and $\widehat{PC^A}$ be its header. Proposer block PC^A is propagated within Shard A, consequently consuming the inner-shard bandwidth. After PC^A being verified by a TEE and attached with a validation proof, both its header $\widehat{PC^A}$ and the validation proof are propagated to the whole network. This process thus consumes the inter-shard bandwidth. The vote block is also globally broadcast. Since $\widehat{PC^A}$, validation proof, and vote blocks are small in terms of size, they consume less bandwidth compared with proposer blocks. Meanwhile, cross-shard transactions and their Merkle proofs are only sent to their destination shards rather than being broadcast among the whole network, thereby reducing the consumption of inter-shard bandwidth. For example, a proposer block with 4,000 transactions is about 1MB. In comparison, broadcasting cooperation-related data, including TEE proofs and vote blocks, consumes about 87KB with 50 shards. Therefore, lightweight cooperation-based data will not significantly consume cross-shard bandwidth.

5.2.3 Storage Consumption

Most of the storage in each shard is consumed by shard-specific contents, such as the state of accounts, vote chain VC^I, and proposer chain PC^I. Fewer contents are required to be duplicated among all shards compared with shard-specific ones. Considering the storage in Shard I, nodes have to store vote chains VC^K in remaining $(s-1)$ shards to validate consensus results, and headers of $(s-1)$ proposer chains $\widehat{PC^K}$ to confirm cross-shard transactions. Blockchain storage can be further reduced to $2s$ blocks by adopting a similar design to [48], in which each chain is compacted to a single latest block. Key values of TEEs in all shards are also stored to verify validation proofs of proposer blocks. According to our analysis, proposer blocks containing all referred transactions consume more than 13,000 KB, while vote blocks consume less than 10 KB when there are 20 shards, much smaller than proposer blocks. Although the vote block size also grows with the increased number of shards, its storage overhead can be ignored compared with proposer blocks.

6 Evaluation

We conduct both prototype experiments and large-scale simulations to evaluate the performance of the proposed Benzene in a wide range of the network size.

6.1 Experimental Setup

We first implement a prototype of our Benzene and conduct experiments to evaluate its performance. In particular, we construct a blockchain prototype system based on four high-performance servers with up to 100 node instances. Meanwhile, we implement TEE verification codes in Intel SGX, which is deployed at a computer with 2.80 GHz CPU with SGX-enabled BIOS support. Typically, a blockchain

Authorized licensed use limited to: INDIAN INST OF INFO TECH AND MANAGEMENT. Downloaded on April 15,2024 at 07:10:20 UTC from IEEE Xplore. Restrictions apply.

system can be abstracted and divided into four layers: application layer, data layer, consensus layer and network layer [49]. In the application layer, we implement functions of maintaining accounts. Each account stores balance values of users, and are manipulated according to confirmed transactions transferring tokens between accounts. In the data layer, we implement data structures of transactions and two types of blockchains. Each transaction includes an input address, an output address, and the value of transferred tokens. The size of each transaction is 250 bytes. We choose ECDSA to create digital signatures and SHA-256 in the hashlib package as the hash function linking blocks. Each proposer block contains 20,000 transactions on average. Each vote block packages proposer block headers and generates a Merklre Tree with the merkletools package. In the consensus layer, we implement the cooperation protocol based on the PoW consensus. Participants perform mining process to mine the valid proposer blocks and vote blocks, and decide the consensus results according to the voting-based algorithm. In the network layer, we implement blockchain network. We use multi-threading in Python to implement nodes working in parallel and we use MPI (Message Passing Interface) for Python to ensure the communication between miners. Nodes are randomly assigned to different shards according to their addresses. Each node mines blocks and requires TEE verification services to conduct cross-shard verification. Moreover, each node maintains a memory pool to store valid but yet unconfirmed blocks to accelerate the verification process, similar to Bitcoin [43]. We follow some typical configurations. The bandwidth of each node is 30Mbps and the end-to-end latency is 100 ms. The block interval for each block is about 15 seconds [14].

We implement two blockchain system prototypes for comparing with our Benzene. In particular, we implement a non-cooperative sharding scheme, namely *Non-Coop Sharding*, which is a PoW-based sharding system with the exclusion of the cross-shard cooperation. In Non-Coop Sharding, miners only process transactions of the same shard and maintain a single chain in each shard. Meanwhile, we also implement a non-sharding scheme, namely *Non-Sharding*, which confirms transactions with multiple vote chains mining in parallel, similar to the consensus protocol in Prism [18]. In Non-Sharding, there are one proposer chain and multiple vote chains in the system. The number of vote chains is equal to the number of shards in Benzene for fair comparison. In prototype experiments, each shard of Benzene contains 5 nodes and 1 TEE. Regarding Non-Coop Sharding, the 1/4 fault tolerance limitation causes only 2 shards being allowed according to our analysis in Section 5.1.1.

We also conduct simulations to further evaluate the performance of Benzene and other baselines with up to 2,500 nodes and 500 shards. We simulate three baseline schemes: Monoxide, Non-Coop Sharding and Non-Sharding (Prism). Benzene and Monoxide distribute 5 nodes in each shard while Non-Coop Sharding has two shards to achieve the same 1/4 fault tolerance as Benzene. The simulated network has the same settings (e.g., bandwidth, latency, and block size) as the prototype experiments. We simulate the random block creation and TEE verification processes to get a precise estimation of the execution time. The block interval is

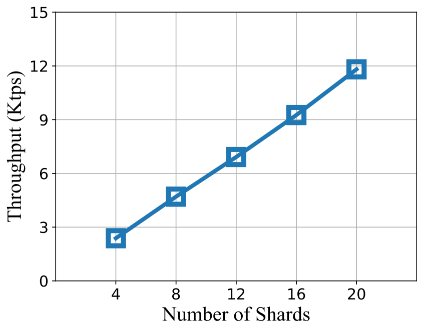


Fig. 7. Throughput in prototype experiments.

re-tuned to 20 seconds to cover the time of both block generation and TEE verification. The end-to-end latency also follows random distribution with its mean being 0.1 sec [14], and the latency for transactions being broadcast to all shards increases logarithmically with the number of nodes [50].

6.2 Scalability Evaluation

We want to evaluate the scalability of Benzene and answer the following question:

- Can Benzene achieve better performance as the number of shards increases?

To answer this question, we measure the throughput and latency in both prototype experiments and simulations, with the increased number of shards.

Throughput. We first evaluate the system throughput in prototype experiments. We measure TPS of Benzene with 4, 8, 12, 16, and 20 shards; each shard contains 5 nodes and 1 TEE. When measuring the practical throughput, we avoid repeatedly counting on the duplicated transactions in proposer blocks and cross-shard transactions. Fig. 7 shows that TPS of Benzene can linearly scale with the increased number of shards. For example, Benzene achieves 11,810 TPS with 20 shards. To evaluate the scalability of Benzene in a large scale network, we evaluate its throughput by simulations. The number of shards is increased from 4 to 500 with each shard containing 5 miners and 1 TEE. Results in Fig. 8 demonstrate that the throughput of Benzene can scale almost linearly with the enlarged network, e.g., achieving 40× increment from 4 shards to 200 shards. This further demonstrates that Benzene can achieve a highly-scalable throughput despite the slightly-increased cross-shard communication overhead. Moreover, we measure the performance of Benzene beyond 200 shards and observe that the increment of throughput starts to stop at around 400 shards.

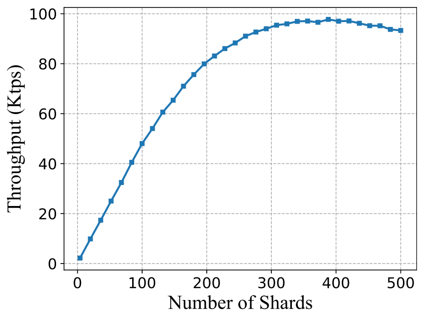


Fig. 8. Throughput evaluated by simulations.

Authorized licensed use limited to: INDIAN INST OF INFO TECH AND MANAGEMENT. Downloaded on April 15,2024 at 07:10:20 UTC from IEEE Xplore. Restrictions apply.

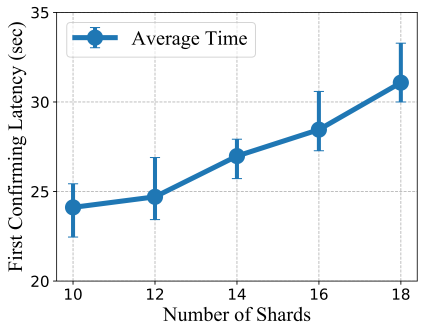


Fig. 9. First confirming latency in prototype experiments.

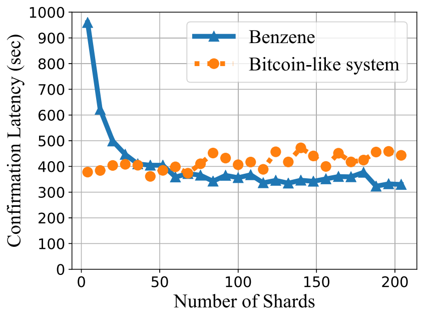


Fig. 11. Confirmation latency in simulations.

This is because we assume that the number of deployed TEEs is limited in simulations, therefore leading to a maximal number of blocks that can be verified per second.

First-Confirming Latency. In Benzene, each proposer block has to be confirmed by vote blocks, each of which has to wait for sequential vote blocks before being finally confirmed. The first-confirming latency is the time interval from the time when a proposer block is successfully mined to the time when it receives enough vote blocks at the same height. Fig. 9 shows the average first-confirming latency in prototype experiments and error bars show the maximal and minimal first-confirming latency. First-confirming latency increases with the increased number of shards. This is reasonable since more shards lead to a larger network and in turn require more time in block propagation.

Confirmation Latency. The confirmation latency is the time interval from the time when a proposer block is mined to the time when it is finally confirmed by sequential z vote blocks. Giving a fixed low probability ϵ of the proposer chain being reversed, the value of z depends on the number of shards s as well as the fault tolerance β where the calculation of z value can be referred to Section 5.2.1. We measure the confirmation latency with $\beta = 1/4$, $\epsilon = 0.001$ and the different number of shards. Fig. 10 shows average confirmation latency in prototype experiments and error bars show the maximal and minimal confirmation latency. The confirmation latency shows fluctuation since the value of z decreases while the network propagation latency increases. Moreover, to reveal how the latency varies in a broader range of the network size, we measure the confirmation latency from 4 to 200 shards through simulations. We also compare the confirmation latency of Benzene with that of Bitcoin in simulations. Fig. 11 shows that the confirmation latency drops at first while maintaining a stable level when the number of shards increases to hundreds. This is mainly owed to the decreased z value. Moreover, when there are more than 50 shards, Benzene can achieve a lower confirmation latency compared with the Bitcoin-like system. This is because Bitcoin requires $z = 15$ while Benzene with 50 shards achieves $z = 13$ and further drops to about 9 as the number of shards increases to 200 when $\epsilon = 0.001$ and $\beta = 0.25$.

Takeaway. The throughput of Benzene will increase linearly from 4 shards to 200 shards. Besides, the confirmation latency will remain at a stable level.

6.3 Performance Comparison

We want to compare the performance of Benzene with other baselines and answer the following question:

- Can Benzene achieve better performance than other baselines while preserving the same fault tolerance?

Throughput Comparison in Prototype Experiments. We compare our Benzene with Non-Coop Sharding, and Non-Sharding schemes in terms of TPS with the same fault-tolerance value 1/4 and the same number of nodes. We conduct experiments on Benzene with 8, 10, 12, and 14 shards. Fig. 12 plots the results of prototype experiments. Due to the latency caused by the TEE verification, TPS of Benzene is slightly lower than that of Non-Coop Sharding at the beginning. But, Benzene obtains a more rapid improvement when the number of nodes increases, e.g., achieving a higher TPS with 50 nodes than both Non-Coop Sharding and Non-Sharding. This is because Benzene can scale to more shards while preserving security, therefore achieving better performance. In comparison, Non-Coop Sharding cannot be fully scaled due to its decreasing fault tolerance. Although the performance of Non-Sharding may be slightly increased with more nodes, such increase is not comparable with the performance enhancement brought by sharding.

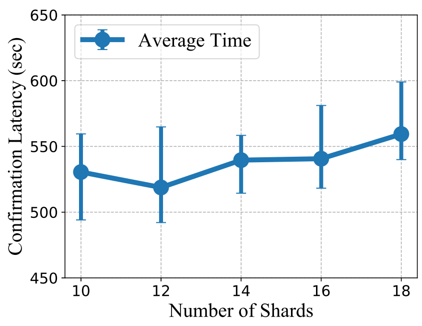


Fig. 10. Confirmation latency in prototype experiments.

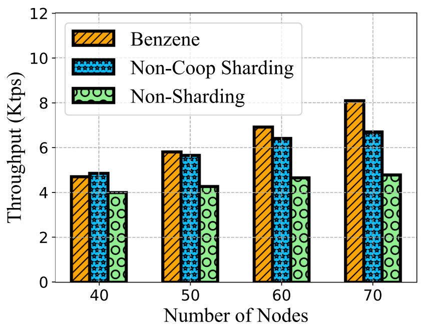


Fig. 12. Throughput comparison in prototype experiments.

Authorized licensed use limited to: INDIAN INST OF INFO TECH AND MANAGEMENT. Downloaded on April 15,2024 at 07:10:20 UTC from IEEE Xplore. Restrictions apply.

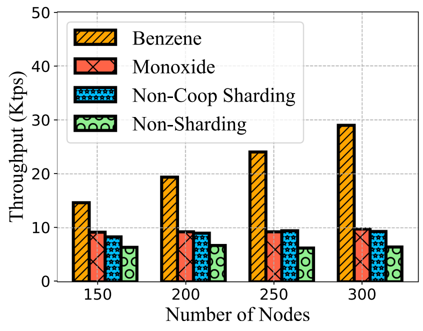


Fig. 13. Throughput comparison in simulations.

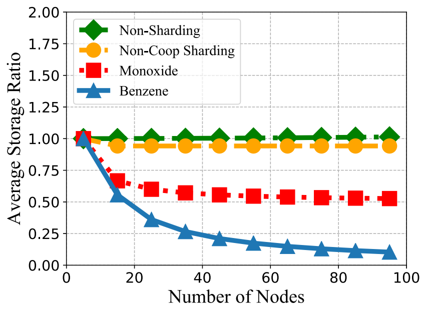


Fig. 15. Average storage consumption reduction.

Throughput Comparison in Simulations. Moreover, we also evaluate Benzene with a comparison with other baselines in a larger scale network with a fixed fault tolerance through simulations. All approaches are tested under 150, 200, 250, and 300 nodes while preserving 1/4 fault tolerance. As shown in Fig. 13, the TPS of Benzene scales up much faster than those of other approaches, whose TPS just slightly increases with the increased number of nodes. This is mainly because Benzene can scale up to more shards without compromising security while Non-Coop Sharding cannot. Monoxide can scale to more shards, but it requires miners to download and verify blocks from other shards, making its performance similar to Non-Coop Sharding. Although Non-Sharding adopting the voting-based consensus can increase the throughput and reduce the latency, such performance improvement is not comparable with the improvement brought by sharding schemes. Benzene combines the advantages of both voting-based consensus and sharding schemes at the same time, thereby achieving a better performance.

Average Waiting Time for Transactions. We compare the average latency of a transaction from the generation (i.e., being submitted by users to the network) to the first confirmation among four simulated approaches: Benzene, Monoxide, Non-Coop Sharding, and Non-Sharding. Such latency can reflect the system efficiency of processing transactions. All four approaches are tested with 50 nodes. We generate transactions under the varied transaction frequency, i.e., from 1,000 transactions per second to 10,000 transactions per second. Fig. 14 shows that the latency in all four approaches remains stable initially, and continuously increases when the transaction-generation frequency exceeds a specific level. This is because when the transaction-generation frequency is higher than the system processing capacity, a group of the generated transactions have to wait in transaction pools, thereby prolonging the latency of being packed. Further, Non-Sharding first reaches its performance bottleneck, followed by Non-Coop Sharding and Monoxide. By contrast, Benzene is the last one reaching the bottleneck owing to its fully-utilized capacity even when more than 6,000 transactions are submitted to the network per second.

Storage and Bandwidth. We compare the average storage and bandwidth consumption per transaction and per node of Benzene with those of other baselines. In simulations, we divide the total storage and bandwidth consumption among the whole network by the total amount of transactions and the number of nodes. Such a metric can describe how many additional resources are required when a new transaction is submitted to the system. Moreover, we divide such average consumption to that of a non-sharding scheme to illustrate how many times of reduction a sharding system can achieve by distributing storage and bandwidth overheads among shards. All baselines are tested from 5 nodes to about 100 nodes. Both Figs. 15 and 16 demonstrate that Benzene requires less than 1/4 of the average storage and bandwidth consumption compared with non-sharding schemes, and achieves lower resource consumption compared with other sharding schemes. Reasons behind these observations are as follows. In non-sharding Prism, the storage and bandwidth consumption are equally imposed on all nodes. When the number of nodes increases, the average storage and bandwidth consumption remain stable. Non-Coop Sharding partitions such workloads into shards. However, such reduction is limited since it cannot scale to more shards when the number of nodes increases due to its weak fault tolerance. In Monoxide, miners in one shard store, verify, and pack transactions in other shards to protect the per-shard security. In other words, Monoxide increases the storage and bandwidth overheads to preserve security. By contrast, Benzene can partition storage and bandwidth workloads into shards as far as possible when the number of nodes increases, without adding considerable overheads to preserve security.

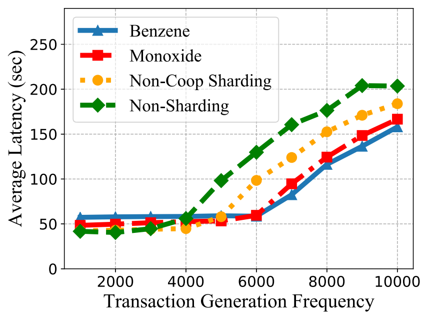


Fig. 14. Average latency of submitted transactions.

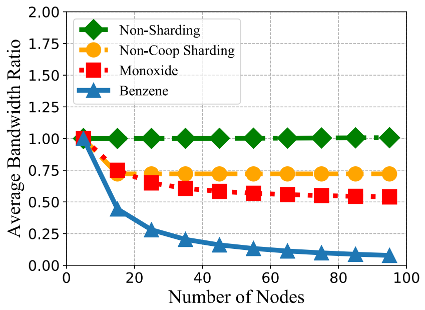


Fig. 16. Average bandwidth consumption reduction.

Authorized licensed use limited to: INDIAN INST OF INFO TECH AND MANAGEMENT. Downloaded on April 15,2024 at 07:10:20 UTC from IEEE Xplore. Restrictions apply.

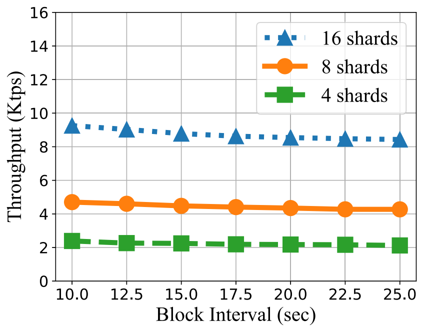


Fig. 17. Throughput under varied block interval.

Takeaway. Users can scale Benzene to numerous shards while preserving a fixed fault tolerance. Moreover, Benzene can effectively distribute the storage and bandwidth overhead among shards. Therefore, Benzene can achieve better performance with less resource consumption compared with other baselines.

6.4 Parameter Evaluation

We want to answer the following question:

- How will the performance metric of Benzene vary when we change some parameter settings?

Block Interval. To measure the impact of the block interval on TPS, we measure the TPS according to different intervals of the proposer block creation in prototype experiments. We simulate the block creation process to fulfill the adjustment of block creation interval. Our experiments are conducted under 4, 8 and 16 shards. Fig. 17 plots the TPS versus the block interval. The experimental results show that the throughput slightly decreases when the block interval extends. This is because miners have to spend more time on generating a valid proposer block for a longer block interval, thereby limiting the computation capacity consumed on packing transactions. However, such impact can almost be ignored when the interval increases from 10 seconds to 25 seconds. Moreover, more shards lead to a higher TPS; this is consistent with our observation mentioned above.

Bandwidth. We also measure the varied throughput under different bandwidth configurations through simulations. Fig. 18 shows that the throughput of Benzene with 100 shards increases from about 20,000 TPS to 40,000 TPS when the per-node bandwidth raises from 5 Mb/s to 30 Mb/s.

Takeaway. The throughput of Benzene can be increased when the block interval decreases or the bandwidth increases.

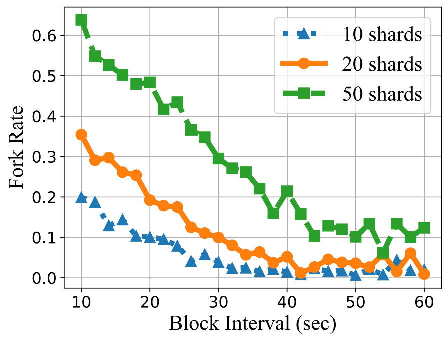


Fig. 19. Fork rate with varied block interval.

Users can select parameters to balance the trade-off between the performance and the security/costs.

6.5 Security Evaluation

We evaluate the security of Benzene with regard to forks on blockchains [51]. We aim to evaluate the security of Benzene by answering the following question:

- Can Benzene preserve security by ensuring a low fork rate?

Fork Rate Without Attackers. First, we evaluate the fork rate on proposer chains with varied block intervals in simulations. The fork rate is calculated by dividing the number of forks to the number of block intervals [52]. Fig. 19 illustrates that the fork rate decreases when the block interval prolongs. Fewer than 20% of consensus epochs will have forks with 20 shards when the block interval is 20 seconds. Moreover, more shards lead to a higher fork rate since it takes more time to broadcast blocks and thus increases the probability of another valid proposer block being successfully mined. But this ephemeral phenomenon is acceptable since forks will soon be resolved by vote blocks.

Probability of Successful Attack. Moreover, we evaluate the fork on proposer chains caused by adversarial attacks with varied fraction of malicious mining power in simulations. Malicious nodes can focus their mining power on half of the vote chains to cause a fork on the proposer chain. We measure the probability that such an attack occurs with different values of attacker fraction and the varied number of shards. Each vote block has to wait for sequential 6 blocks before being confirmed, i.e., the same as Bitcoin. An attack is successfully conducted only when more than half of the vote blocks are orphaned since the attacker mines a longer chain. Fig. 20 shows that the attack probability is limited to a low level (almost 0 when less than 22% mining power is

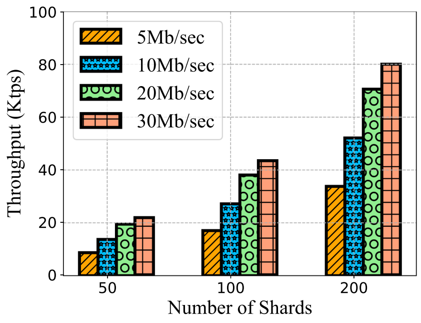


Fig. 18. Throughput under different bandwidth.

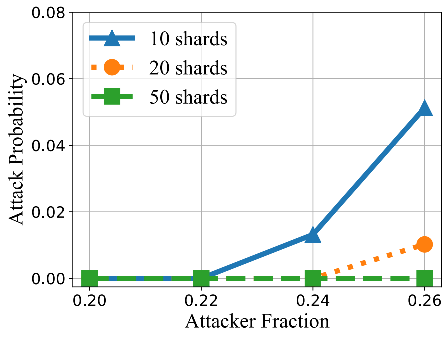


Fig. 20. Attack rate with varied attacker fraction.

Authorized licensed use limited to: INDIAN INST OF INFO TECH AND MANAGEMENT. Downloaded on April 15,2024 at 07:10:20 UTC from IEEE Xplore. Restrictions apply.

malicious, and only 0.05 when the attacker fraction is 0.26) and increases when approaching 1/3 fault tolerance. Moreover, the existence of more shards leads to a lower probability of Benzene being attacked; this is consistent with our previous analysis in Section 5.

Takeaway. For honest participants, the fork rate can be reduced as the block interval increases. Besides, the probability of an adversary successfully launching attacks by reversing consensus results is low when less than 26% mining power is malicious.

7 Conclusion

We propose Benzene, a sharding protocol based on cross-shard cooperation to enhance system security and performance. Unlike previous sharding approaches, Benzene allows multiple shards to cooperate with each other in the consensus protocol, consequently enhancing the system fault tolerance. Benzene has three key novel designs: i) a *double-chain architecture* to separate the transaction-recording function from the consensus-execution function, ii) a *cross-shard verification mechanism* with TEEs to minimize per-node overhead, iii) a *voting-based consensus* to realize cross-shard cooperation upon the task of consensus-execution. With double-chain architecture, each shard can record transactions independently while participating in consensus execution of other shards. Miners in each shard verify blocks from other shards with the minimized overhead, and cast votes to confirm those blocks in the voting-based consensus. Such a design enables a fully scaling of the system while achieving a fixed 1/3 fault tolerance. Moreover, voting-based consensus provides a shorter confirmation latency compared with previous single-chain PoW-based protocols (e.g., Nakamoto protocol). Experiments show that the throughput of Benzene can be linearly improved (i.e., 32,370 TPS with 50 shards) with the number of shards increasing to 200. Moreover, Benzene can achieve a lower confirmation latency than non-sharding Bitcoin while achieving the same security guarantee, even when the sharding technique weakens the system fault tolerance. Finally, Benzene requires less storage and bandwidth consumption on defending per-shard security, and maintains a low fork probability since voting-based consensus limits the occurrence of forks.

References

[1] M. J. Amiri, D. Agrawal, and A. E. Abbadi, "Caper: A cross-application permissioned blockchain," *Proc. VLDB Endow.*, vol. 12, no. 11, pp. 1385–1398, 2019.

[2] 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*, 2017, pp. 557–564.

[3] G. Danezis and S. Meiklejohn, "Centrally banked cryptocurrencies," in *Proc. Symp. Netw. Distrib. Syst. Secur. Symp.*, 2016, pp. 1–14.

[4] H. A. Kalodner, S. Goldfeder, X. Chen, S. M. Weinberg, and E. W. Felten, "Arbitrum: Scalable, private smart contracts," in *Proc. 27th USENIX Secur. Symp.*, 2018, pp. 1353–1370.

[5] I. Eyal, A. E. Gencer, E. G. Sirer, and R. V. Renesse, "Bitcoin-NG: A scalable blockchain protocol," in *Proc. 13th Usenix Conf. Netw. Syst. Des. Implementation*, 2016, pp. 45–59.

[6] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proc. 26th Symp. Operating Syst. Princ.*, 2017, pp. 51–68.

[7] T. Crain, C. Natoli, and V. Gramoli, "Red belly: A secure, fair and scalable open blockchain," in *Proc. IEEE 42nd Symp. Secur. Privacy*, 2021, pp. 466–483.

[8] Y. Buchnik and R. Friedman, "Fireledger: A high throughput blockchain consensus protocol," in *Proc. VLDB Endowment*, vol. 13, no. 9, pp. 1525–1539, 2020.

[9] H. Huang, W. Kong, S. Zhou, Z. Zheng, and S. Guo, "A survey of state-of-the-art on blockchains: Theories, modelings, and tools," *ACM Comput. Surv.*, vol. 54, no. 2, pp. 1–42, 2021.

[10] G. Wang, Z. J. Shi, M. Nixon, and S. Han, "SoK: Sharding on blockchain," in *Proc. 1st ACM Conf. Adv. Financial Technol.*, 2019, pp. 41–61.

[11] J. Hellings and M. Sadoghi, "Byshard: Sharding in a byzantine environment," *Proc. VLDB Endowment*, vol. 14, pp. 2230–2243, 2021.

[12] 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*, 2018, pp. 583–598.

[13] M. Zamani, M. Movahedi, and M. Raykova, "RapidChain: Scaling blockchain via full sharding," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 931–948.

[14] J. Wang and H. Wang, "Monoxide: Scale out blockchains with asynchronous consensus zones," in *Proc. 16th USENIX Symp. Netw. Syst. Des. Implementation*, 2019, pp. 95–112.

[15] V. Buterin, "Ethereum sharding faq," 2019. Accessed: April 01, 2021. [Online]. Available: https://github.com/ethereum/wiki/wiki/ShardingFAQ

[16] F. McKeen et al., "Innovative instructions and software model for isolated execution," in *Proc. 2nd Int. Workshop Hardware Architectural Support Secur. Privacy*, 2013, Art. no. 10.

[17] V. Costan, I. A. Lebedev, and S. Devadas, "Sanctum: Minimal hardware extensions for strong software isolation," in *Proc. 25th USENIX Secur. Symp.*, 2016, pp. 857–874.

[18] V. Bagaria, S. Kannan, D. Tse, G. Fanti, and P. Viswanath, "Prism: Deconstructing the blockchain to approach physical limits," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2019, pp. 585–602.

[19] Z. Hong, S. Guo, P. Li, and C. Wuhui, "Pyramid: A layered sharding blockchain system," in *Proc. IEEE Int. Conf. Comput. Commun.*, 2021, pp. 1–10.

[20] M. Al-Bassam, A. Sonnino, S. Bano, D. Hrycyszyn, and G. Danezis, "Chainspace: A sharded smart contracts platform," in *Proc. 25th Annu. Netw. Distrib. Syst. Secur. Symp.*, 2018, pp. 1–15.

[21] Y. Tao, B. Li, J. Jiang, H. C. Ng, C. Wang, and B. Li, "On sharding open blockchains with smart contracts," in *Proc. IEEE 36th Int. Conf. Data Eng.*, 2020, pp. 1357–1368.

[22] S. Li, M. Yu, C.-S. Yang, A. S. Avestimehr, S. Kannan, and P. Viswanath, "Polyshard: Coded sharding achieves linearly scaling efficiency and security simultaneously," in *Proc. IEEE Int. Symp. Inf. Theory*, 2020, pp. 203–208.

[23] M. J. Amiri, D. Agrawal, and A. El Abbadi, *SharPer: Sharding Permissioned Blockchains Over Network Clusters*. New York, NY, USA: Association for Computing Machinery, 2021, pp. 76–88.

[24] G. Pîrlea, A. Kumar, and I. Sergey, "Practical smart contract sharding with ownership and commutativity analysis," in *Proc. 42nd ACM SIGPLAN Int. Conf. Program. Lang. Des. Implementation*, 2021, pp. 1327–1341.

[25] P. Zheng, Q. Xu, Z. Zheng, Z. Zhou, Y. Yan, and H. Zhang, "Meepo: Sharded consortium blockchain," in *Proc. IEEE 37th Int. Conf. Data Eng.*, 2021, pp. 1847–1852.

[26] 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.*, 2016, pp. 17–30.

[27] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *Proc. 3rd Symp. Operating Syst. Des. Implementation*, 1999, pp. 173–186.

[28] M. Vukolić, "The quest for scalable blockchain fabric: Proof-of-work versus. bft replication," in *Proc. Int. Workshop Open Problems Netw. Secur.*, 2015, pp. 112–125.

[29] H. Dang, T. T. A. Dinh, D. Loghin, E.-C. Chang, Q. Lin, and B. C. Ooi, "Towards scaling blockchain systems via sharding," in *Proc. Int. Conf. Manage. Data*, 2019, pp. 123–140.

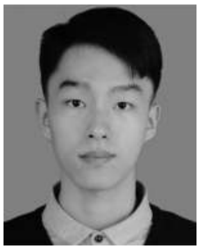
[30] J. Behl, T. Distler, and R. Kapitza, "Hybrids on steroids: SGX-based high performance BFT," in *Proc. 12th Eur. Conf. Comput. Syst.*, 2017, pp. 222–237.

[31] V. Costan and S. Devadas, "Intel SGX Explained," *IACR Cryptol. ePrint Arch.*, vol. 2016, 2016, Art. no. 86.

[32] T. Alves, "Trustzone : Integrated hardware and software security," *Inform. Quarterly*, vol. 3, pp. 18–24, 2004.

Authorized licensed use limited to: INDIAN INST OF INFO TECH AND MANAGEMENT. Downloaded on April 15,2024 at 07:10:20 UTC from IEEE Xplore. Restrictions apply.

[33] S. Matetic, K. Wüst, M. Schneider, K. Kostiainen, G. Karame, and S. Capkun, "BITE: Bitcoin lightweight client privacy using trusted execution," in *Proc. 28th USENIX Secur. Symp.*, 2019, pp. 783–800.
[34] J. Lind, O. Naor, I. Eyal, F. Kelbert, E. G. Sirer, and P. Pietzuch, "Teechain: A secure payment network with asynchronous blockchain access," in *Proc. 27th ACM Symp. Operating Syst. Princ.*, 2019, pp. 63–79.
[35] R. Cheng et al., "Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contracts," in *Proc. IEEE Eur. Symp. Secur. Privacy*, 2019, pp. 185–200.
[36] P. Das et al., "Fastkitten: Practical smart contracts on bitcoin," in *Proc. 28th USENIX Secur. Symp.*, 2019, pp. 801–818.
[37] M. Brandenburger, C. Cachin, R. Kapitza, and A. Sorniotti, "Trusted computing meets blockchain: Rollback attacks and a solution for hyperledger fabric," in *Proc. IEEE 38th Symp. Reliable Distrib. Syst.*, 2019, pp. 324–333.
[38] Y. Yan, "Confidentiality support over financial grade consortium blockchain," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2020, pp. 2227–2240.
[39] M. Brandenburger, C. Cachin, R. Kapitza, and A. Sorniotti, "Blockchain and trusted computing: Problems, pitfalls, and a solution for hyperledger fabric," 2018, *arXiv:1805.08541*.
[40] S. Matetic et al., "Rote: Rollback protection for trusted execution," in *Proc. 26th USENIX Secur. Symp. USENIX Secur.*, 2017, pp. 1289–1306.
[41] H. Yu, I. Nikolic, R. Hou, and P. Saxena, "OHIE: Blockchain Scaling Made Simple," in *Proc. IEEE Symp. Secur. Privacy*, 2020, pp. 90–105.
[42] G. Wang, S. Wang, V. K. Bagaria, D. Tse, and P. Viswanath, "Prism removes consensus bottleneck for smart contracts," in *Proc. IEEE Crypto Valley Conf. Blockchain Technol.*, 2020, pp. 68–77.
[43] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: http://bitcoin.org/bitcoin.pdf
[44] A. Dembo et al., "Everything is a race and nakamoto always wins," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2020, pp. 859–878.
[45] M. Brandenburger and C. Cachin, "Challenges for combining smart contracts with trusted computing," in *Proc. 3rd Workshop Syst. Softw. Trusted Execution*, 2018, pp. 20–21.
[46] G. Chen, Y. Zhang, and T.-H. Lai, "Opera: Open remote attestation for intel's secure enclaves," in *Proc. Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2019, pp. 2317–2331.
[47] I. Anati, S. Gueron, S. Johnson, and V. Scarlata, "Innovative technology for cpu based attestation and sealing," in *Proc. 2nd Int. Workshop Hardware Architectural Support Secur. Privacy*, 2013, Art. no. 7.
[48] B. Bunz, L. Kiffer, L. Luu, and M. Zamani, "Flyclient: Super-light clients for cryptocurrencies," in *Proc. IEEE Symp. Secur. Privacy*, 2020, pp. 928–946.
[49] H. Chen, M. Pendleton, L. Njilla, and S. Xu, "A survey on ethereum systems security: Vulnerabilities, attacks, and defenses," *ACM Comput. Surv.*, vol. 53, no. 3, pp. 1–43, 2020.
[50] A. Montresor, "Gossip and epidemic protocols," *Wiley Encyclopedia Elect. Electron. Eng.*, vol. 1, pp. 1–15, 2017.
[51] G. Bissias and B. N. Levine, "Bobtail: Improved blockchain security with low-variance mining," in *Proc. Symp. Netw. Distrib. Syst. Secur.*, 2020, pp. 1–16.
[52] C. Decker and R. Wattenhofer, "Information propagation in the bitcoin network," in *Proc. IEEE P2P Proc.*, 2013, pp. 1–10.



Zhongteng Cai received the BS degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2020. He is currently working toward the MS degree with Sun Yat-sen University, Guangzhou, China. His current research interests include blockchain technologies and consensus algorithms.

Junyuan Liang received the MS degree from Sun Yat-sen University, Guangzhou, China, in 2022. He is currently working toward the PhD degree with Sun Yat-sen University. He is working on edge computing and blockchain system, in particular on storage, payment channel network and sharding in blockchain.

Wuhui Chen (Member, IEEE) received the bachelor's degree from Northeast University, Shenyang, China, in 2008, and the master's and PhD degrees from the University of Aizu, Aizu–Wakamatsu, Japan, in 2011 and 2014, respectively. From 2014 to 2016, he was a Research Fellow with the Japan Society for the Promotion of Science, Japan. From 2016 to 2017, he was a Researcher with the University of Aizu. He is currently an associate professor with Sun Yat-Sen University, Guangzhou, China. His research interests include edge/cloud computing, cloud robotics, and blockchain.

Zicong Hong received the BEng degree in software engineering with the School of Data and Computer Science, Sun Yat-sen University. He is currently working toward the PhD degree in the Department of Computing, Hong Kong Polytechnic University. His current research interest includes Blockchain, Game Theory, Internet of Things, and Edge/Cloud Computing.

Hong-Ning Dai (Senior Member, IEEE) received the PhD degree in computer science and engineering from the Department of Computer Science and Engineering, Chinese University of Hong Kong. He is currently with the Department of Computer Science, Hong Kong Baptist University, Hong Kong, as an Associate Professor. His current research interests include blockchain and the Internet of Things. He has served as associate editors of *IEEE Transactions on Intelligent Transportation Systems*, *IEEE Transactions on Industrial Informatics*, *IEEE Systems Journal*, and *IEEE Access*. He is a member of the ACM.

Jianting Zhang received the bachelor's and master's degrees from Sun Yat-Sen University, China, in 2019 and 2022, respectively. He is currently working toward the PhD degree with Purdue University. His research interest focus on blockchain, smart contracts, and consensus algorithms.

Zibin Zheng (Fellow, IEEE) is currently a professor and the Deputy Dean with the School of Software Engineering, Sun Yat-sen University, Zhuhai, China. He authored or coauthored more than 200 international journal and conference papers, including one ESI hot paper and six ESI highly cited papers. According to Google Scholar, his papers have more than 15 000 citations. His research interests include blockchain, software engineering, and services computing. He was the BlockSys'19 and CollaborateCom16 General Co-Chair, SC2'19, ICIOT18 and IoV14 PC Co-Chair. He is a Fellow of the IET. He was the recipient of several awards, including the Top 50 Influential Papers in Blockchain of 2018, the ACM SIGSOFT Distinguished Paper Award with ICSE2010, the Best Student Paper Award with ICWS2010.

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

Authorized licensed use limited to: INDIAN INST OF INFO TECH AND MANAGEMENT. Downloaded on April 15,2024 at 07:10:20 UTC from IEEE Xplore. Restrictions apply.