RepChain: A Reputation based Secure, Fast and High Incentive Blockchain System via Sharding

Chenyu Huang*, Zeyu Wang*, Huangxun Chen, Qiwei Hu,
Qian Zhang†, *Fellow, IEEE*, Wei Wang, *Member, IEEE*, Xia Guan

***Abstract*—In today's blockchain system, designing a secure and high throughput blockchain on par with a centralized payment system is a difficult task. Sharding is one of the most worthwhile emerging technologies for improving the system throughput while maintain high security level. However, previous sharding related designs have two main limitations: Firstly, the security and throughput of their random-based sharding system is not high enough as they did not leverage the heterogeneity among validators. Secondly, to design an incentive mechanism that promote cooperation could incur a huge overhead on their system. In this paper, we propose RepChain, a reputation-based secure and fast blockchain system via sharding, which also provides high incentive to stimulate node cooperation. RepChain utilizes reputation to explicitly characterize the heterogeneity among the validators and lay the foundation for the incentive mechanism. We propose a new double-chain architecture - a transaction chain and a reputation chain. For transaction chain, an efficient Raft-based synchronous consensus has been presented. For reputation chain, the synchronous Byzantine fault tolerance consensus that combines collective signing has been utilized to prevent the attack on both reputation score and the related transaction blocks. It supports a high throughput transaction chain with moderate generation speed. Moreover, we propose a reputation-based sharding and leader selection scheme. To analyze the security of RepChain, we propose a recursive formula to calculate the epoch security within only $\mathcal{O}(km^2)$ time. Furthermore, we implement and evaluate RepChain on the Amazon Web Service platform. The results show our solution can enhance both throughout and security level of the existing sharding-based blockchain system.**

***Index Terms*—Blockchain, Reputation, Sharding**

I. Introduction

The Internet of Things (IoT) is a promising technique to enhance the connection between all physical objects in the world, the number of which was 4.9 billion in 2015 and expected to be 25 billion by 2020 [1]. However, IoTs still face many security issues such as point of failure, DDoS, Sybil attack and *etc* [2]. In recent years, the development of blockchain has shown great potential for its integration with IoTs to address the aforementioned security issues [3]–[5]. Technically, blockchain is a decentralized and public digital ledger which records data in distributed nodes. In ideal cases of such an intermediary-free system, consensus, *i.e.* an agreement on data is expected to be achieved efficiently by the honest majority (*i.e.*, high throughput) and will be resistant to retroactive modification by malicious users (*i.e.*, high security). Existing permissionless blockchain systems are still far from meeting the above expectations. For example, Bitcoin and Ethereum can only handle 7 and 15 transactions per second respectively [6], which barely meet the requirement of IoTs.

The fundamental limitation is attributed to a global consensus requirement, *i.e.*, all data should be validated by every validator. Thus, the validation workload increases with the growing number of validators, but the system capacity remains unchanged. Sharding is an intuitive solution to tackle this limitation, where the validators are separated into several groups so that transactions can be processed in parallel to boost throughput. In the state-of-the-art literature, several sharding-based protocols (RSCoin [7], Elastico [8], OmniLedger [9], RapidChain [10] and Monoxoide [11]) have been proposed to address the trade-off between throughput and security.

However, we argue that a practical blockchain system should take validator heterogeneity and incentive mechanisms into design consideration to fully exploit the potential of sharding techniques. Firstly, most existing works regard validators to be the same except for the distinction of honest/malicious attributes. However, validators in a practical system present such a difference in terms of computing capability, communication bandwidth and historic behaviors. Thus, under random sharding [7]–[9], [11] or simple balanced random sharding [10] protocols, those less-competent validators become bottlenecks and hamper system throughput. Secondly, an incentive mechanism is crucial for validator activation and retention. In a practical blockchain system, it is undesirable to rely on a trusted centralized bank [7] to allocate rewards. Simply allocating rewards to a shard leader or block miner ([8], [11]) would result in income variance, *i.e.*, the rewards of validators with modest capabilities have high variance, which may dampen validators' enthusiasm for participation.

In this paper, we introduce 'reputation' in the context of the sharding system to jointly address the above two issues in practical systems. A reputation established on historical behaviors is the cornerstone of many trust systems. For example, merchants in the markets build up their reputations on long-term fair trading to earn trust from customers [12], [13] and; nodes in Peer-to-Peer systems establish their reputations on active participation in file sharing to obtain other nodes'

*Co-primary Authors, †Corresponding author
Chenyu Huang and Qian Zhang are with Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, China (Email: chuangak@connect.ust.hk, qianzh@cse.ust.hk).
Huangxun Chen is with Theory Lab of 2012 Labs, Huawei, Shenzhen, China (Email: hchenay@connect.ust.hk).
Zeyu Wang is with TuSimple, Beijing, China (Email: uchihatmtk-inu@gmail.com).
Qiwei Hu and Wei Wang are with School of Electrical Information and Communication, Huazhong University of Science and Technology, Wuhan, 430074, China (Email: qwh@hust.edu.cn, weiwangw@hust.edu.cn).
Xia Guan is with Oasis Future (Shenzhen) Holdings, Shenzhen, China (Email: gx@oasisfuture.com)

cooperation [14], [15]. In recent years, such a concept has infiltrated the blockchain [5], [16], [17]. However, their systems are not designed for sharding protocols. Thus, to fill this gap, we propose RepChain to integrate reputation with a sharding-based blockchain to boost the throughput, enhance the security and also provide a desired incentive mechanism.

Reputation scores in our system are calculated based on the behaviors of validators, which indicate validators' capabilities and reliability. Reputation score serves three important purposes. Firstly, it boosts the system throughput by helping elect a highly capability leader. Secondly, it enhances the system's security by helping to balance multiple shards to have similar proportions of active and inactive, honest and malicious validators so that it is more difficult for attacker to take control of one shard. The basic idea to achieve this goal is enforcing different shards to have the same sized and similar total reputation scores during the sharding procedure. Generally, a reputation-based sharding system only degrades to a random sharding one with an advanced attacker in the worst case. Thirdly, it greatly provides great incentive for the validators to do their best by helping establish a reputation-based reward scheme. Therefore, every devoted contribution will be recognized and an honest and competent validator who contributes more will be allocated more rewards.

A strawman solution is to use Byzantine fault tolerance (BFT) consensus directly, but it is inefficient. To calculate the reputation score of a certain validator, all n validators should achieve consensus on this validator's behaviour - their votes for transactions, which requires transmitting $\mathcal{O}(n^2)$ messages. Thus, to calculate all validators' reputation scores, we need to transmit $\mathcal{O}(n^3)$ messages.

Thus, two challenges should be addressed for a practical system. Firstly, we should design an efficient consensus protocol for transactions to avoid those with lower processing capabilities and more security breaches becoming the bottlenecks. Secondly, we should design an efficient consensus protocol for reputation scores. On the one hand, such a consensus should be reached to improve throughput, enhance security level and provide incentive mechanism. On the other hand, we should keep the system overhead brought by reputation score at an acceptable level without impairing these benefits.

RepChain novelly proposes the double-chain architecture, *i.e.*, maintains transaction chain and reputation chain separately. For transaction chain, RepChain revises the traditional Raft to a synchronous consensus protocol. Compared to other sharding-based works, the combination of reputation-based leader election and Raft consensus eases the workload of the least capable validator gives incentives to the competent leader to contribute more so as to improve the system's throughput. The Raft protocol only needs $\mathcal{O}(n^2)$ message complexity for the consensus of all n validators' votes on the transactions. Thus, we can achieve transaction agreements efficiently in a sharding-based blockchain. However, the Raft protocol cannot prevent a Byzantine attacker. Thus, a synchronous Byzantine fault tolerance consensus, CSBFT, that combining the **C**ollective **S**igning [18] and the RapdiChain's **BFT** consensus [10] has been proposed for reputation chain. To balance the benefits and overheads of such a consensus, we pack the hashes of multiple confirmed transaction blocks, the reputation scores of all validators calculated based on their transaction votes and one collective signature into one reputation block. In this way, a reputation block provides proof of authenticity for multiple transaction blocks. Thus, RepChain can achieve efficient consensus on both reputation scores and transaction chains without incurring too much overhead, *i.e.*, a reputation chain with moderate generation speed can support a high throughput transaction chain. It is worth mentioning that existing sharding systems can also benefit from this idea with a little modification on their consensus and sharding scheme to gain the benefits brought by reputation.

To evaluate RepChain, we first provide analysis on the security, throughput and incentive mechanisms with a simple reputation scheme. Compared with previous works [9], [10] that only estimate the failure probability of one epoch, we propose a new recursive formula to give the exact solution in time complexity of $\mathcal{O}(km^2)$. Besides, we implement RepChain and evaluate it using the Amazon Web Service with 900 instances (450 from US West and 450 from US East). We simulate 1800 nodes and set the shard size to 225 to maintain a high security level. The results show that our system achieves the throughput of 6852 transactions per second (tps) and the user-perceived latency of 58.2 seconds on average. We also evaluate our system under three different threat models and various validator capabilities. The results present that our system can enhance the throughput and security level of a sharding-based blockchain system.

In summary, this paper makes the following contributions:

1) RepChain is the first to integrate reputation scores with sharding-based blockchain, which explicitly characterizes the heterogeneity among the validators and lays the foundations for the incentive mechanism.
2) RepChain is the first sharding-based system with double-chain architecture, which enables a reputation chain generated via the CSBFT consensus to support a high throughput transaction chain generated via Raft.
3) We propose the reputation-based sharding and leader selection scheme to boost the system throughput and enhance the security level.
4) We analyze the security and performance of the system, implement the proposed system and conduct extensive evaluations on Amazon Web Services to validate the effectiveness of our design.

II. Related Work

In this section, we introduce existing efforts to address blockchain scalability problem, and elaborate sharding-based blockchains and the reputation-based systems in state-of-the-art literatures.

A. Blockchain Scalability

It is well known that scalability is an unsolved issue in existing blockchain systems especially considering the integration of IoT and blockchain [2], [19]. Considerable efforts have been made to tackle this problem from various perspective. Bitcoin-NG [20] boosts system throughput by allowing the shard leaders in an epoch to append more blocks than those in Bitcoin. However, this method results in substantially

increasing computation overhead of each node. ByzCoin [18] leverages a collective signing technique to reduce communication complexity of PBFT, but it does not make fundamental changes on PBFT to solve the scalability issue. Proof-of-Stake (PoS) is a promising direction. Instead of computation power, validators use their stake to vote the leader. Ouroboros and its follow-up works [21], [22] propose to elect the leader based on coin-flipping in an epoch way. Algorand [23] proposes a new byzantine agreement scheme by forming a committee that the members are randomly selected from all the nodes via the verifiable random function (VRF). However, PoS may suffer from the monopoly problem that only the richest are permitted to have control of consensus. Directed Acyclic Graph (DAG) tries to use a finite diected graph with no directed cycles to replace the chain structure. IOTA [24] and SPECTRE [25] use the DAG for transactions and blocks respectively to enable the parallelism of blockchain. However, the valid transactions in DAG are possible to be falsely appended to parasite chains [26]. Lightning network [27] and Bolt [28] both resort to off-chain solution, which allows the participants to transfer micro-payments through a payment channel without appending it onto the main chain. However, the off-chains do not have the same security guarantees as the main chain, which results in vulnerability to various attacks [29], [30].

B. Sharding-based Blockchain

A sharding-based blockchain basically distributes transactions to different shards. It is an intuitive solution to improve scalability, as the total throughput is equal to the product of in-shard throughput and the number of shards. This idea attracts attentions of researchers and a few high throughput and secure systems have been built upon it. RSCoin [7] leverages sharding to build a centrally banked system, where a simple Two Phase Commit (2PC) has been utilized between the user and a set of mintettes from one shard. Elastico [8] proposed the first sharding-based permissionless blockchain with BFT consensus. It achieves a near-linear computational scalability and tolerates corruptions of 1/4 nodes. OmniLedger [9] also adopts sharding, which selects a leader via a verifiable random function (VRF) and utilizes a variant of ByzCoin [18] to improve throughput. RapidChain [10] considers sharding in the context of synchronous protocols, which improves throughput and is resilient to corruptions of 1/3 nodes. Monoxide [11] proposes asynchronous consensus zones that shard the Proof-of-Work (PoW) blockchain. They propose eventual atomicity to ensure transaction atomicity across zones, and Chu-ko-nu mining to ensure the effective mining power within one zone. Dang*et al.* [31] propose a sharding blockchain based on trusted hardware, Intel SGX, to achieve high performance for both consensus and shard formation protocol.

C. Reputation and Blockchain

In traditional P2P network, it is common to leverage reputation as an incentive mechanism [14], [15]. In recent year, reputation concept has penetrated into blockchain. CertChain [32] proposed a Dependability-rank based consensus and incentive mechanism, which takes the economic benefits and misbehavior into consideration. However, their design is tailored for certifying authorities (CA) but not to address scalability issue. B-IoT [5] and PoT [17] both propose credit-based PoW systems to reduce high computing cost of PoW. B-IoT [5] built on Directed Acyclic Graph (DAG) and thus suffers from the drawbacks in DAG. PoT [17] generates the reputation based on the trust network that reported by every nodes. However, to defend attacks, they assume a trust seeding where a set of trusted nodes are identified and the trust network evolves around it. RepuCoin [16] tried to integrate reputation with blockchain. They proposed a reputation-based weighting scheme consensus to avoid high computation cost as in PoW. However, their reputation scores are based on the total amount of valid works from the very beginning of the chain without decay. It could result in a severe monopoly problem and double spending attacks if high reputation users collude with each other. Moreover, it is extremely difficult for new validators to join the consensus group and get the rewards. More seriously, their system could be defeated if an attacker joins in the very beginning.

III. Model and Overview

This section introduces the system model, network model, threat models and the overview of RepChain respectively.

A. System Model

An *epoch* in RepChain denotes the time interval between events of validator assignment to shards as other sharding blockchains [8]–[10]. In RepChain, we assume a permissionless model similar to that in RapidChain [10], *i.e.*, any node can join the system at epoch e by submitting a PoW puzzle solution. However, the newly joined validator should start working at epoch $e+1$. Thus, the number of validators within an epoch is static. Without loss of generality, we assume there are n validators $V^n = \{v_1, v_2, ..., v_n\}$ and k shards $C^k = \{C_1, C_2, ..., C_k\}$ in an epoch e. Thus, there are $m = n/k$ validators in each shard, including one leader and $m-1$ members. The reputation score of each validator in epoch e is r_i^e. Clients will send transaction tx to validators, who will generate both transaction chain and reputation chain via consensus.

B. Network Model

Our network model is similar to that in [10]. We assume all messages delivered in the network are authenticated with the sender's private key. Moreover, we assume that the connections between honest nodes are well established and the transmission can be finished within duration Δ in the intra-shard consensus, *i.e.*, the communication channel is synchronous within one shard. To address the poor responsiveness of synchronous consensus, we adopt the method in RapidChain [10], where all the members would agree on a new Δ every week to achieve a long-term responsiveness. Except intra-shard consensus, RepChain are built on partially-synchronous channels with optimistic exponentially increasing time-outs.

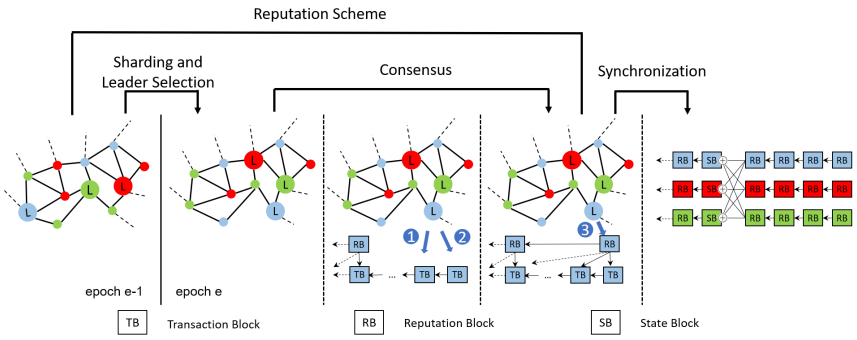


Fig. 1. RepChain Overview. Different color represents different shard. Each shard will generate its own transaction chain and reputation chain. The order of the arrow shows the leader will first build the TB and then generate RB after severl TBs.

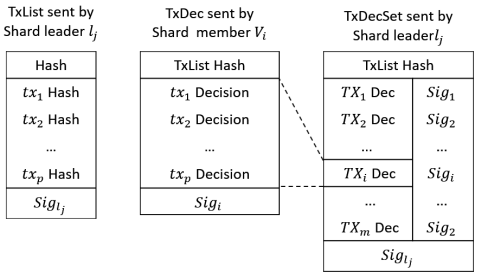


Fig. 2. Data structure of TxList, TxDec and TxDecSet.

C. Threat Model

We assume a Byzantine adversary who corrupts less than $f = n/3$ nodes. The corrupted nodes can collude with each other and conduct arbitrary behaviours such as sending invalid information or remaining silent. It is assumed that the adversary corrupts the fixed part of the nodes, which is different from the slowly adaptive attackers in [8]–[10]. We argue that in a practical system, validators rarely have the same capabilities and secure protection, *i.e.*, some nodes are easily corrupted while others are more robust. Besides, our system only kicks out a node when it misbehaves as the shard leader. Thus, for the adversary's benefit, it is more realistic to control a fixed part of nodes and pretend to be good nodes until they get high reputations or even are selected as shard leaders, then suddenly launch an attack to the system. Thus, this paper considers three attacker strategies as follows.

Simple Attack: The malicious nodes do bad things continuously.

Camouflage Attack: The malicious node pretends to be a normal node until it becomes a shard leader. Other malicious nodes within the same shard will support the malicious leader.

Observe-Act Attack: The attacker observes the reputation score distribution of the normal nodes. Then the attacker controls the malicious nodes to act and have the same reputation score distribution as the normal one. The purpose of this strategy is to increase the probability that most malicious nodes are grouped into one shard.

Moreover, we also consider several attacks on reputation scores as follows.

Self-Promoting Attack: The malicious leader tries to increase the reputation scores of the malicious validators.

Slandering Attack: The malicious leader tries to decrease the reputation scores of honest validators.

D. System Overview

Our system maintains a double-chain architecture, *i.e.*, a transaction chain and a reputation chain. It has four main components as in Fig. 1: sharding and leader selection, consensus, reputation scheme, and synchronization .

Sharding and Leader Selection: All the nodes are assigned into different shards at the beginning of each epoch, then each shard selects its shard leader.

Consensus: The clients send transactions txs to the shards which are responsible for the input UTXOs (unspent transaction output). Then these shards run intra-shard synchronous consensus to extend the transaction blockchain at high speed and the reputation blockchain at moderate speed. For cross-shard transactions, our system adopts Atomix protocol from OmniLedger.

Reputation Scheme: Given behaviors of all validators by the consensus, all validators can calculate and reach a consensus on the reputation scores. The reputation scores enhance previous two components by helping elect a high capability leader in the shard leader selection, and helping balance multiple shards to have similar proportions of active/inactive, honest/malicious validators. Therefore, all three components work together to provide a high throughput, secure and high-incentive blockchain.

Synchronization: At the end of each epoch, each shard generates a state block to conclude the transaction blockchain and the reputation blockchain. The nodes synchronize and update the local data based on these state blocks from all the shards, and then the system starts a new epoch.

IV. System Design

This section presents the detailed design of RepChain.

A. Sharding and Leader Selection

In sharding and leader selection, our system maintains four properties as follows:

- *Randomness*: It is hard to predict the results of the sharding and leader selection.
- *Balance*: Each shard has the similar total reputation score, i.e., similar proportions of active/inactive, honest/malicious validators. It is difficult for malicious users to take control of one shard.
- *Uniformity*: The results can be validated by each validator locally without too much communication overhead.
- *Incentive*: For the validators, the higher the reputation, the higher the probability to be a shard leader.

When an epoch e begins, all the validators will be sharded into different groups. For each group, a shard leader will be selected based on their reputation as Alg. 1 shows. Decaying is an important feature in reputation scheme, thus we adopted a sliding window w to calculate the cumulative reputation

score R^w. Specifically, the validators use the random generator RNG from the random $seed^e$ to generate random numbers. Such seed could be generated by the secure distributed bias-resistant randomness generation protocol that adopted by OmniLedger [9] and RapidChain [10]. R^{sort} denotes the validators' reputation scores sorted in descending order (Line 3). Next, each validator is randomly assigned to a shard with minimum size to maintain the balance property (Line 4-7). For shard leader selection, we adopt the following strategy to maintain the incentive, randomness and uniformity property simultaneously (Line 11-22). First, the validators with reputation scores higher than the median have chances of being selected as the leader (Line 12-14). It is worth mentioning that RepChain is not allowed the new arrival to be the leader candidate with the reputaiton limitation rule above. Because the new arrival could be attacker which occurs with the probability of 1/3. Then each validator will calculate a number $p_{i,j}$ comes from dividing a random number generated based on the same seed $Seed^e$ by its reputation score. (Line 15-16) Finally, the validator with the minimum $p_{i,j}$ among $(p_{i,0}, pi, 1, ..., p_{i,m})$ is selected as the leader of shard i (Line 21).

Algorithm 1 Sharding and Leader Selection Algorithm

Input:
A random $Seed^e$ and the cumulative reputation score $R^w = r_1^w + r_2^w + ... + r_n^w$ over previous w epochs;
Output:
The k shards $C = \{C_1, C_2, ..., C_k\}$;
The k leaders $L = \{l_1, l_2, ..., l_k\}$.

1: Initialize $C_i = \emptyset$, $L_i = \emptyset$ for each $1 \leq i \leq k$.
2: Set the seed of random generator RNG as $Seed^e$.
3: R^{sort} = sort(R^w)
4: **for** each r_i^{sort} in R^{sort} (r_i^{Sort} is validator v_g's score) **do**
5: Find a shard C_t that has the minimum cardinality.
6: **if** multiple C_t satisfied the requirements **then**
7: Randomly select one based on RNG
8: **end if**
9: Assign validator v_g to C_t, $C_t = C_t \cup \{v_g\}$.
10: **end for**
11: **for** each shard $C_i \in C$ **do**
12: rm = median of the subset of R^w that belongs to C_i
13: **for** each validator $v_j \in C_i$ **do**
14: **if** $rm \leq r_j^w$ **then**
15: Generate a random float $0 \leq y \leq 1$ from RNG.
16: $p_{i,j} = y/r_j^w$
17: **else**
18: $p_{i,j} = +\infty$
19: **end if**
20: **end for**
21: $l_i = v_j$ where $p_{i,j} = \min(p_{i,1}, p_{i,2}, ...p_{i,m})$
22: **end for**

B. Consensus

Similar to Bitcoin, each transaction tx will have a unique identity, a list of input UTXOs and a list of output UTXOs. UTXO, short for unspent transaction output, is the unused coin from previous tx and contains the signature. The shard i will only store the tx with specific identity, *i.e.*, the prefix of the identity is equal to i [10], [11]. The input UTXOs may come from different shards. Thus, our system also needs to handle cross-shard transactions. We denote the shard be responsible for the input UTXO as the input shard, and the shard for the output UTXO as the output shard.

Inspired by Raft [33], OmniLedger [9], the protocol proposed by Ren *et al.* [34] and RapidChain [10], we propose a synchronous consensus to achieve high throughput and 1/2 resilience within a shard. Our consensus constructs two chains within each shard: a Raft consensus to generate a transaction chain and a Byzantine fault tolerance consensus, CSBFT, to generate a reputation chain. It is worth mentioning we can also use Raft consensus for reputation chain and CSBFT for transaction chain. In this scheme, the reputation chain is now vulnerable to the Byzantine attack. Thus, the TB should contain the hash of the RB to prevent Byzantine attack. Such scheme will have the same security level and function as RepChain but be less efficient than RepChain because: (i) the size of a TB (4MB) is much larger than that of a RB (<200KB); (ii) RepChain only generates one RB after generating several TBs.

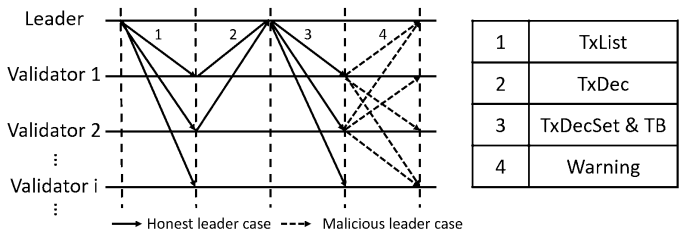


Fig. 3. The basic communication pattern of the Raft consensus under honest and malicious leader cases.

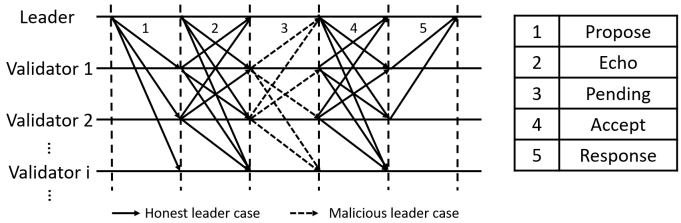


Fig. 4. The basic communication pattern of the CSBFT under honest and malicious leader cases.

1) Intra-shard Consensus: Firstly, the leader generates transaction chain via Raft consensus. The related data structures are shown in Fig. 2. Transaction list (TxList) orders the transactions so that every validator can calculate the reputation based on the same batch of transactions. Transaction decisions (TxDec) given by the member contains the decisions in the same order as the TxList. Transaction decision set (TxDecSet) includes all the TxDecs from shard members. The Fig. 4 Illustrates the communication pattern of the Raft consensus. (1) The leader will send the TxList to all the validators; (2) Each validator will verify all the txs in TxList and make a decision, *Accept*, *Reject* or *Unknown* for every tx. *Unknown* is

used when validator cannot handle too many transactions due to the hardware limitations. Then the validators will send the TxDec back to the leader; (3) The leader packs all the TxDecs to the TxDecSet. Meanwhile, the leader can generate the TB based on TxDecSet. Then leader will send both TxDecSet and TB to all the validators; (4) If the leader launch the self-promoting attack or slandering attack by modifying the TxList, TxDecSet or TB (see next subsection), the validators will send *warning* to each other and lead to the view-change. After this protocol, the validators can calculate the reputation scores. The Raft consensus can efficiently generate TB and fully leverage a high capability leader than the BFT which adopts all-to-all communication. However, the Raft consensus cannot prevent Byzantine attack. Thus, these TBs can only be viewed as candidate TBs and will finally achieve consensus in the second step.

Secondly, RepChain generates RB via a Byzantine fault tolerance consensus - CSBFT. The structure of RB can be viewed in Fig. 5 and it includes the hashes of candidate TBs as well as the reputation of all the validators. CSBFT is modified from RapidChain [10] which can achieve 1/2 resiliency within a shard. However, RapdiChain's consensus is not efficient in cross-shard transaction since they use the multiple signatures of honest validators as a proof for the transaction validation. Thus, CSBFT combines it with Collective Signing (CoSi) [18] which could aggregate multiple signatures into one signature as Fig. 4 shows. Specifically, the consensus of RapidChain contains four synchronous round: *propose*, a leader proposes a hash digest H for consensus; *echo*, validators send the H received from the leader to all other nodes with tag echo iff the H is correct; *pending*, if an honest validator receives different hashed H and H', it will send the H' with tag pending; *accept*, if an honest validator receives $f+1$ echo of the same and the only H, it will accept and send H with tag accept. After the consensus, the honest validator can know whether the leader is corrupted or not. CoSi also has four rounds: announce, commit, challenge and response. We refer to read the RapidChain [10] and ByzCoin [18] for the technical detail. The propose and echo round in the consensus can combined with announce and commit round respectively in the CoSi. Specifically, the leader sends announcement of signing H in $propose$ round. The validators will echo the leader the H and a random secret in $echo$ round. In the *accept* round, the leader send a collective Schnorr challenge and the H with $f+1$ echos to all the validators. After *accept*, the honest can determine whether the leader is corrupt or not. Thus, another round of $response$ is needed: validators can accept or reject to send the aggregate response to the leader and leader will generate the collective signature on the RB.

2) Warning in Intra-shard Consensus: In intra-shard consensus, we use $warning$ to detect the malicious behavior of the leader. Specifically, if the leader sends invalid TxList, TxDecSet or send the TB that contains tx less than half of the votes, the $f+1$ honest validator will send $warning$ to others. Once $f+1$ *warning* are received, view-change happens. Malicious leader can also choose to remove a TxDec of an honest validator from TxDecSet so as to decrease its reputation. The victim can send *warning* with its TxDec to all other validators. All the honest validators will increase the reputation score of the victim and relay such information to leader. If the malicious leader refuses to add the reputation score of the victim, the hash of RB, H, will be different between malicious leader and honest validators.

3) Cross-shard Protocol: For cross-shard transactions, we used Atomix protocol from OmniLedger, *i.e.*, the client will help to relay the transactions across shard with the proof-of-acceptance. In our system, the proof-of-acceptance is the collective signature on RB.

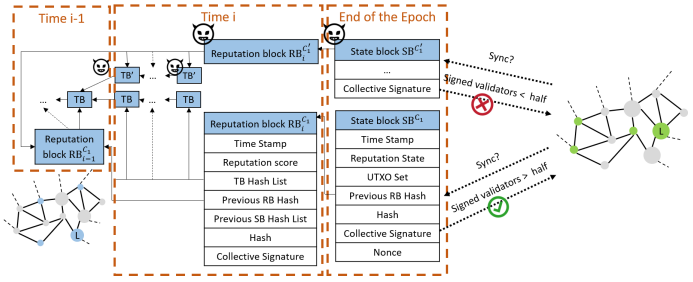


Fig. 5. This figure shows the structure of the reputation block (RB) and state block (SB) as well as the working procedure of collective signing in the synchronization process. At the end of the epoch, the validators will generate a state block that contains the collective signature. The validators in other shards can verify the legality of the state block by checking whether the collective signature is signed by enough validators.

C. Reputation Scheme

In this section, we illustrate the design of the reputation scheme which can enhance the security, incentive property and throughput of RepChain.

1) Reputation Score Calculation: At the end of one intra-shard consensus, the reputation scores within the shard can be calculated by all shard members individually and uniformly based on TxDecSet and TB. Since reputation scheme is not our core contribution we adopt a simple scheme that inspired by PeerTrust [14]. The reputation score r_i of validator i are calculated as follows:

$$r_i = \sum_{j=1}^{l} S(j) * T(j)$$

Where l is the number of transactions generated after the previous RB. We introduce $T(j)$ (the value of transaction j) to prevent the case where some validators are honest on low-value transactions but dishonest on a high-value transaction. The scaling factors $S(j)$ is used to reward or penalize different behaviors differentially. eBay [35] simply gives a reputation score of {-1, 0, 1} to correct, unknown and incorrect decisions. However, a validator can still gain a lot of profits even it is dishonest from time to time. Thus, our system sets different scaling factors for different behaviors. The standard of the setting is as follows. (1) The punishment for the wrong decision should be larger than that for the correct decision, because the former is more likely to be a malicious attack. (2) The punishment for agreeing to a wrong transaction should be larger than that for denying a correct one, because the former can bring more damage to the system than the latter. For *"Unknown"* decision, the validators neither earn nor lose their reputation scores, since it is believed that these

validators generally have lower capacities (CPU, hard disk, and bandwidth etc).

2) Reputation Blockchain: After the reputation score calculation, the validators utilize collective signing to generate the RB, as shown in Fig. 5. RB contains the reputation score that the validator owned in this epoch, the confirmed TBs, the previous RB and the collective signature. Recalling that in previous section, the consensus ensures the honest validators will sign on the RB iff the linked transaction blocks and the reputation scores are correct. Other shards can check the collective signature, and accept it if more than half of the validators sign on the block. Moreover, the malicious validators can fork the transaction blockchain and the reputation blockchain. However, as shown in Fig. 5, if a malicious validator in shard C_1 (blue) forks the blockchain by generating the malicious TBs and RBs, the shard C_2 (green) can easily check the collective signature to pick the correct blockchain signed by the honest majority.

D. Synchronization

In the sharding system, the transactions are separated into different shards. Thus at the end of one epoch, the validators of all shards need to synchronize their reputation blockchain and transaction blockchain to prepare for the next epoch. To prevent the huge cost of sending the whole blockchain, the validators launch a round of CSBFT to generate a state block (SB).

The structure of SB is shown in Fig. 5. It contains the overall reputation scores of the validators over the past w epochs and the UTXO set of the clients by this epoch. We also combine the UTXOs that belong to one public key. For example, if two UTXOs have the values of a and b respectively and both belong to the public key PK_x. The validators will drop the UTXO that has the value of b and adds the value onto the remaining UTXOs. The drop principle is to reserve the UTXO with the smaller address. Thus, a validator only needs to download the state block of the other shard instead of downloading all of the data in TB and RB.

E. Double-Chain Architecture

As we mentioned before, we utilize the double-chain architecture to enhance security, throughput and incentive mechanism. In this section, we put all the pieces together to show how double-chain architecture works.

The double-chain architecture includes the transaction chain and the reputation chain. RepChain uses Raft consensus to generate transaction chain in a fast speed. To prevent Byzantine fault, the CSBFT has been used to finally confirm the reputation block and its relevant transaction blocks. The attacks on TxList, TxDecSet and TB can be early detected by *Warning*. Thus, the TB can only be achieved the consensus iff the corresponding RB that links to it has achieved consensus.

V. System Analysis

In this section, we will analyze the security, the performance and the incentive mechanism of RepChain.

A. Epoch Security

For epoch security, previous work [10] only estimates the upper bound of the failure probability via hypergeometric distribution since the straightforward calculation for the exact solution requires $\mathcal{O}(m^k)$ time. We propose a novel recursive formula to calculate exact solution in the time complexity of $\mathcal{O}(km^2)$, and prove RepChain only degrades to the random-based sharding scheme in the worst case.

Firstly, we provide the recursive formula for random-based sharding. We denote $F(x, y)$ as the number of the safe allocations, where x malicious nodes are assigned to y shards. An unsafe shard refers that half of the shard members are malicious nodes. If the above assignment results in any one unsafe shard, we regard such assignment as a failure, and the failure probability is $P(failure) = 1 - F(g, k)/C_n^g$. If the y-th shard contains s malicious nodes, then the original question is reduced to the number of safe allocations, where $g - s$ malicious nodes are distributed to $y - 1$ shards. Thus, the equation can be calculated recursively as follows:

$$F(x, y) = \sum_{s=0}^{d} F(x - s, y - 1)C_m^s \tag{1}$$

where $d = \lfloor \frac{m-1}{2} \rfloor$ and $F(x, 1) = C_m^x \mathbb{1}_{x \le d}$. For the case of random-based sharding, given $n = 1800$ and $k = 8$, $P(failure) = 1.25553\text{e}{-07}$.

Secondly, we prove that the *observe-act attack* degrades the security level as well as the performance of RepChain to that of the random-based sharding. Since the malicious validators maintain the same reputation score distribution as the honest nodes, the possibility of a validator being an attacker or a honest node is independent of the reputation scores. However, as the calculation above, it is still secure enough.

Thirdly, we prove RepChain is more secure under *Camouflage Attacker* than random sharding. Given all malicious validators support the malicious leader under *Camouflage Attacker*, they expose themselves due to rolling scheme and their reputation scores will decrease. Their cumulative reputation score will be lower than honest validators the following w epochs. Assuming a ($a = pk + q, 0 \le q \le k - 1$) malicious nodes are exposed, they will be allocated into k shard equally according to our sharding scheme. The allocation is equivalent to distribute $g - pk$ malicious nodes to k shards where each shard already contains p nodes. For clear elaboration, we first consider the rest q exposed validators will be assigned to k-th to $(k - q + 1)$-th shard. We denote $F(x, y, t)$ as the number of the safe allocations, where x is the number of malicious validators except the exposed ones, y denotes the number of shard and t is the number of exposed malicious nodes. It can be calculated similar to Equation 1 as follows:

$$F(x, y, t) = \sum_{s=0}^{d-u_l} F(x - s, y - 1, max(t - 1, 0))C_{m-p-u_l}^s \tag{2}$$

where $l = k - y$, $u_l = \mathbb{1}_{l \in [0, q-1]}$, $v = max\{0, q - l - 1\}$ and $F(x, 1, 0) = C_m^x \mathbb{1}_{x \le d}$. The above recursive equation can be calculated within $\mathcal{O}(km^2)$ time. Since we only consider the situation that the q validators are assigned to the last q shards, the total number of safe allocation should be $C_k^q F(g - a, k, q)$ and the failure probability is $(C_k^q F(g - a, k, q))/(C_k^q C_{n-a}^{g-a})$

when considering all situations. Fig 6 shows the failure probability with different a when $n = 1800$. The failure probability decreases with a increases, *i.e.*, the system are more secure when more malicious nodes are exposed. Thus, the security level of RepChain under such attack will be higher than random sharding scheme. For *Simple Attack*, the attacker can never success since the attackers are always exposed to the system.

B. Security of Intra-shard Consensus

In Sec. V-A, we prove that a shard having more than half of the malicious nodes is almost impossible. Based on this, we prove the safety and liveness of our intra-shard consensus.

1) Safety of Intra-shard Consensus: Raft consensus cannot guarantee the safety of TB due to it is not a Byzantine fault tolerance consensus. safety of the RB and the related TBs is achieved by the CSBFT. Thus, we only need to prove the adding *response* round will not violate the safety property of the original RapidChain's Consensus as follows: As all the honest has know whether or not to accept these blocks, the honest validators can determine whether or not to response the leader before *response* round. Thus, the malicious block proposed by malicious leader cannot gain the response from honest validators to aggregate a collective signature signed by $f + 1$ validators.

2) Liveness of Intra-shard Consensus: We use the definition of liveness as the finality for one block which is the same as other works [10]. Recalling that the expectation round for an honest leader is around two in the worst case. First, we prove the liveness of Raft consensus on transaction block. The honest leader will send the valid TxList and since half of the shard members are honest, the TxDecSet can be given. Thus all the honest shard members will accept a valid TB. If the current leader is a malicious one who remains silence or sends malicious information, the predefined timeout and the Byzantine fault tolerance consensus will launch view-change and a new leader will eventually be selected. Thus a valid TB will be finally proposed by an honest leader. Secondly, the liveness of Byzantine fault tolerance is guaranteed by the RapidChain's consensus. The adding *response* round will not violated the liveness original consensus as follows: At the end of the *accept* round, all the honest validators has known whether to accept the RB. If the leader is malicious, the view-change happen otherwise otherwise the *response* will be finished by all the honest validators.

3) Attack on Reputation: In threat model, we define two attacks on reputation: self-promoting and slandering attack. In self-promoting attack, the malicious leader may try to modify the reputation in RB so that to increase the reputation of malicious validators. However, the reputation score is determined by the TxList and TxDecSet. The inconsistency between them will cause view-change since the hash H of RB will be different between honest validators and malicious leaders. In slandering attack, the malicious will try to decrease the reputation of the honest validators. As mentioned above, the modification of RB is impossible. Thus the only choice is to remove the TxDec from honest validators in TxDecSet. However, the *warning* with its TxDec ensure the honest validators add the reputation on the victim as Sec. IV-B shows.

C. Performance Analysis

Without loss of generality, we analyze the complexity of consensus per tx, supposing one tx is processed in RepChain of shard size m.

The time complexity for Raft consensus on transaction chain is $\mathcal{O}(m^2/b)$ amortized on the block size, b. At first, the client sends the tx to all the input and output shards. Specifically, the client sends tx to one validator of each shard, and the validator helps broadcast tx within the shard, which requires $\mathcal{O}(m)$ communication cost. Next, the shard leader generates a TxList the size of which is $\mathcal{O}(b)$, and broadcast both TxList and txs to all shard members. Such a process brings another communication overhead of $\mathcal{O}(m)$. Then, the leader collects all TxDec which needs $\mathcal{O}(m/b)$ complexity and broadcasts the TxDecSet which needs $\mathcal{O}(m^2/b)$ time to all validators to reach an intra-shard consensus. In total, it costs $\mathcal{O}(m^2/b)$ time.

The time complexity of CSBFT consensus on reputation chain is $\mathcal{O}(m^2/b)$. It is reported that the RapidChain's consensus is $\mathcal{O}(m^2/b)$ and the addition *response* round brings communication overhead of $\mathcal{O}(m)$.

From the analysis above, the time complexity of the intra-shard consensus of RepChain is $\mathcal{O}(m^2/b)$. It is the same as that of the RapidChain's intra-shard BFT consensus. Thus, the additional cost are mainly from the extra round consensus of RB. However, the size of RB is only 200KB which is much smaller than TB (4MB) given $m = 225$. Moreover, RepChain generates one reputation block after several transaction blocks. In general, our reputation block brings insignificant overhead.

In addition, if tx is a cross-shard transaction, the time will be increased due to the cross-shard communication. The complexity of the increased time is the same as the OmniLedger [9] which is $\mathcal{O}(n)$ [10]. Therefore, the time complexity of the whole consensus is $\mathcal{O}(m^2/b + n)$.

Although the transaction block is valid iff it is linked by a valid reputation block, the performance of the TB is not influenced by RB too much. Within the shard, we assume the synchronous network. Given such setting, each step of both reputation chain and transaction chain consensus lasts for $\mathcal{O}(\Delta)$ time, i.e., the generation speed of both blocks are fixed and are only determined by Δ. We reserve the predefined time for each step. Thus, the reputation chain will not affect the performance of the transaction chain too much in normal operation. The only exception is the view-change due to malicious leaders. In this case, the transaction blocks that haven't been linked by a valid reputation block will be deleted. However, the expectation of such case only depends on the probability of "the shared leader is malicious", which is the same as the other single chain sharding based blockchain systems and has nothing to do with double-chain architecture.

It is worth mentioning that the shard leader contributes more bandwidth and computing resources to generate the TxList, TxDecSet, and TB than other validators. Different from the previous work, our scheme explicitly considers capability difference among the validators. Specifically, the reputation scheme eases the workload of the least capable validators to improve the system performance by encouraging the honest and competent validator to contribute more. These validators generally have higher computing resources and bandwidth to process the transactions, thus they will accumulate reputa-

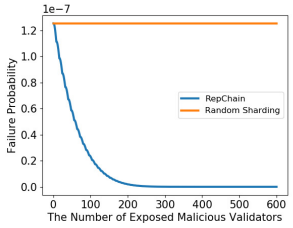


Fig. 6. The failure probability of RepChain with different number of exposed malicious validators.

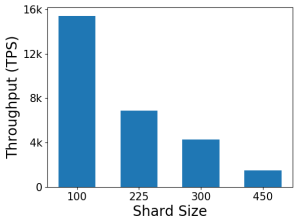


Fig. 7. The average throughput of the transactions for different shard sizes.

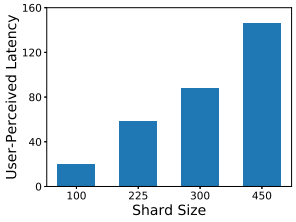


Fig. 8. The average user-perceived latency of a transaction for different shard size

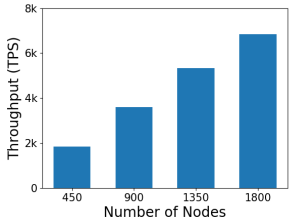


Fig. 9. Throughput scalability

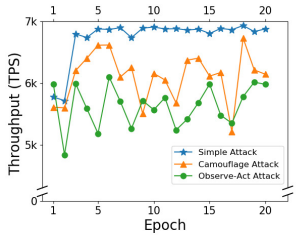


Fig. 10. The average throughput for three attack models

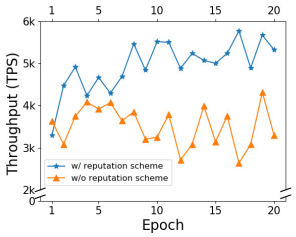


Fig. 11. The average TPS under different validators' abilities w/ and w/o reputation scheme.

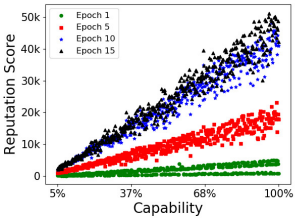


Fig. 12. The reputation score o f validators with different capabilities from 5% to 100%

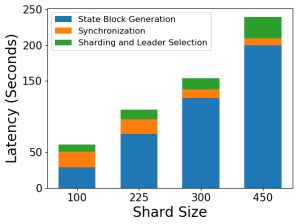


Fig. 13. The epoch transition latency

tion scores more quickly than others. In our scheme, these validators have more chance of being shard leaders. With more contributions from the more capable shard leaders, the throughput of the system can be significantly improved.

D. Incentive Mechanism

RepChain measures the behaviors of the validators in the consensus and records them as reputation in the blocks so that it can provide incentive than other sharding-based blockchains. The incentives including reputation scores and money - transaction fees to make sure being honest is more beneficial to the validators than being dishonest. In transaction verification, half of the transaction fee is given to the leader and the rest is allocated to the other validators based on their reputation scores earned in the current epoch. Thus, even a newly joined node could earn reputation scores and get a stable income under our scheme, if it is honest and works hard. A malicious node may try to cheat the system by being dishonest occasionally. However, on the one hand, such a node would obtain fewer reputation scores than the honest majority, thus it has barely any chance of being a leader and threatening the system. On the other hand, it also earns much fewer rewards than others during this process, i.e., cheating occasionally is not a good strategy for its own benefit. Thus, it is believed that a rational node would not adopt being dishonest. Furthermore, a proper sliding window w is utilized to make sure the selected leader is the honest node who has contributed continuously. In the meantime, w should not be too large to prevent the monopoly. The considerable profit for being the leader would encourage the validators to stay and contribute.

VI. Evaluation

In this section, we first describe the setup of our system and then illustrate the detailed evaluation as follows:

1) The performance of RepChain under different settings.
2) The scalability of RepChain.
3) The performance under different threat models
4) The throughput enhancement via utilizing the heterogeneity among validators.
5) The epoch transition time.

A. Evaluation Setup

We implement RepChain in Go [36]. For collective signing, we use the code in Go cryptography libraries [37]. For the parameters in RepChain, we set the sliding window $e = 10$. The size of the transaction block is 4MB. No more than a 1/3 fraction of nodes is a malicious node. The scaling factor of $S(j)$ in reputation score calculation is set to be 0.1 for a correct decision, 0 for *Unknown*, -0.5 for an incorrect decision that the transaction should be passed but the validator decides to give *Reject*, and -1 for an incorrect decision that the transaction should be rejected but the validator decides to give *Accept*. To simulate the transactions, each validator will generate transactions by themselves and send them to each other.

We run all the experiments on the Amazon Web Service. More specifically, 900 c4.large EC2 instances from two different regions to simulate 1800 nodes. Elastico set all their instances in the US West, OmniLedger and RapidChain set all their nodes in one location. Different from them, we set 450 instances in Oregon from the US West and 450 instances in North Virginia from the US East to simulate a realistic, globally distributed deployment. Besides, we use an extra instance to send the transactions to these 1800 clients. The 1800 clients will use the transaction hashes to represent the transaction in the consensus. Each instance is equipped with 2 Amazon vCPUs and 3.75GB of memory. The bandwidth is set to be 20Mbps for each node. Every node will have the same capabilities in most tests except the throughput enhancement test. The Δ is set to be 1.2s according to the TCP ping testing

on AWS measured by XFT [38]. The interval between two RB is 5 TBs.

We implement three attacks as follows: for all three attacks, we randomly select 1/3 validators as attackers when the system lauches and will not change during running. For *Simple Attack*, the attacker will stay silence or send fake tranasction vote to others all the time during protocol; for *Camouflage Attack*, the attacker only stay silence when it is elected the leader otherwise it operates as an honest validator; for *Observe-Act Attack*, we test our system with a very strong attacker that can observe the score distribution of all the validators. In other words, we cannot distinguish the attackers and honest validators based on their reputation scores which can be viewed as random sharding. Thus, we use the random sharding instead of the reputation-based sharding and use the random leader election instead of the reputation-based leader election. Since the sharding and leader election only operates during epoch transition, the replacement will not change any operation within epoch. And the attackers will stay silence or send fake voting to other validators all the time.

B. Performance

In this evaluation, we test the performance under different shard sizes of RepChain. The metrics of performance comprises two parts: throughput and user-perceived latency. The definition of these two metrics are the same as other works [8]–[10]. The throughput is the number of transactions the system processes in one second. The user-perceived latency is the time that a user sends a tx to the network until the time that tx can be confirmed by any (honest) node in the system [10]. In our case, it is the time when a transaction is proposed until the RB is built so that any user can validate the transaction. More specifically, the system is tested under the *simple attack* with three different shard sizes: 100, 225, 300 and 450. The average throughput and user-perceived latency have been measured and shown in Fig. 7 and Fig. 8. The throughput for the four shard sizes is 15421 tps, 6853 tps, 4288 tps and 1485 tps respectively. The user-perceived latency is 20.4s, 58.2s, 88.5s, 146s With the increase in sharding size, the throughput will decrease and the latency will increase. The poor performance under the shard size of 450 is due to the following reasons. First, RepChain shifts some workload from the members to the leader. With the larger shard size, the bandwidth of the leader for package transmission will increase a lot. Second, the CoSi will cost more when the shard size increase. On one hand, with smaller shard size, the latency will decrease, and the throughput will increase (i.e., better performance). On the other hand, with larger shard size, the system can better ensure honest majority in the committee. Thus, we choose shard size of 225 to balance the trade-offs. The failure probability is 1.25553e-7 in the worst case, which is at the same level as existing sharding-based blockchains such as OmniLedger, RapidChain. In such setting, the throughput of RepChain than other previous works. The latency is almost equal to OmniLedger. This indicates our design really can support a high-throughput transaction chain.

C. Scalability

In this section, we test the scalability of RepChain. The evaluation measure RepChain's throughput based on the shard size of 225 with different numbers of nodes: 450, 900, 1350 and 1800. For each setting, half of the nodes are set in the US West and the remaining half are set in the US East. The result is presented in Fig. 9. The throughput is 1834 tps, 3610 tps, 5333 tps and 6853 tps. With more nodes, the throughput increases linearly. The result indicates the system can be scaled up with more computation power joining the system.

D. Security

In this section, we test three different attack models: *Simple Attack*, *Camouflage Attack* and *Observe-Act Attack* with 1800 nodes and a shard size of 225. The throughput is measured and shown in Fig. 10. For the *Simple Attack*, RepChain runs poorly for first two epochs, but all the attackers will have a lower reputation after the third epoch so that they can barely degrade the performance of RepChain. For the *Camouflage Attack*, the average throughput is 6104 tps. Once the leader is a malicious node, the view-change protocol will be launched and its reputation score will be cleared. Thus, the malicious node needs a long time (around 10 epochs) to gain enough reputation score and be selected as a leader again. Based on this, the system can still perform well. For the *Observe-Act Attack*, the average throughput is 5628 tps which is worse than the previous two attack models. As we mentioned in Sec. V-A, such an attacker will degrade the RepChain to the same as the random sharding system. In other words, the possibility of a malicious leader is always around 1/3 every epoch. From the results and analysis in Sec. V-A, we can conclude that the RepChain can enhance the performance and security level when facing *Simple Attack* and *Camouflage Attack* and is the same level as the random sharding scheme under *Observe-Act Attack*.

E. Throughput Enhancement

In this section, we illustrate that the reputation scheme can provide the benefit in throughput enhancement mentioned in Sec. V-C. We set different validators with different capabilities and test whether the reputation scores are relevant to their capabilities. Specifically, we define the capability of the node in Sec. VI-B as 100% and limit the speed of handling transactions of the validators so that a low capability validator will give more *Unknown* responses than a high capability validator. In this setting, we set $k = 8$ and $n = 1800$. Moreover, all the validators are honest for controlling the variables and their capabilities follow the uniform distribution from 5% to 100%. Fig. 11 shows the throughput enhancement benefits from our reputation scheme. At the beginning, the throughput is less than 4000 tps due to the low capability leader. However after about 8 epochs, the throughput is at average of 5243 tps due to a more capable leader being selected based on its cumulative reputation score. We also run 20 epochs of RepChain without the reputation scheme, *i.e.*, the validators is randomly sharded into 8 groups and the leader is randomly elected. The system without reputation scheme is equivalent to a single-chain plain-BFT-based sharding scheme,

TABLE I
COMPARISON BETWEEN REPCHAIN AND THE EXISTING WORKS.

	#Nodes	Resiliency	Complexity (per tx)	Throughput	Latency	Shard Size	Sharding Scheme	Incentive Mechanism	Time to Fail
Elastico	1600	t<n/4	$\Omega(m^2/b+n)$	40 tx/s	800s	100	Random	No	1 hour
OmniLedger	1800	t<n/4	$\Omega(m^2/b+n)$	3500 tx/s	63s	600	Random	No	230 years
RapidChain	1800	t<n/3	$\mathcal{O}(m^2/b+m\log n)$	4220 tx/s	8.5s	200	Simple - active/inactive	No	1950 years
RepChain	1800	t<n/3	$\mathcal{O}(m^2/b+n)$	5628 tx/s	58.2s	225	Based on reputation	Yes	Depends on attacker's strategy. The worst case is the same as [10].

since transaction blocks are generated via BFT consensus and reputation blocks will not be generated. The average throughput of the system without reputation scheme is 3512 tps which is similar to the first epoch of RepChain but much lower than the following epochs of RepChain with the reputation scheme. The average user-preserved latency is 58.2s and 19.6s with and without reputation scheme respectively. The reason of less latency without reputation scheme is that the block will be confirmed once it achieves the consensus in the plain BFT consensus while the transaction block can only be finally confirmed when the reputation block is generated in RepChain. For the failure probability, RepChain without reputation scheme can be viewed as the random sharding scheme which has the same probability as it with reputation scheme that under the *Observe-Act attack*. With reputation scheme, RepChain will be more secure in *simple attack* and *camouflage attack*.

Fig. 12 shows the relationship between their capabilities and the reputation score in the 1st, 5th, 10th and 15th epoch. We can see the validators barely distinguish between each other at the first epoch. However with more epochs, their abilities can be more easily distinguished according to their reputation scores. Also, the relationship between reputation scores and their capabilities is linear, which indicates the reputation score can correctly reflect the ability of the validators. From the results above, RepChain can distinguish the validators with different capabilities and select a better leader via the sharding and leader selection scheme. Thus, our design can enhance the throughput of the sharding-based blockchain system and incentivize the highly capable nodes.

F. Epoch Transition Latency

In this evaluation, we test the average epoch transition latency of RepChain under different shard sizes of 100, 225, 300 and 450. The latency includes the time for generating state block via collective signing, PoW, synchronization, sharding and leader selection. The results are shown in Fig. 13 which is 61.2s, 111.8s, 155.3s and 242.0s. Among the three stages, the state block generation cost the most time. With a larger shard size, the transition latency will increase due to the overhead on collective signing.

G. Compare with other works

In this section, we compare RepChain with other sharding blockchain works. Monoxide [11] are built on PoW which is different with other BFT based works. The work of [31] is rely on SGX which needs dedicated hardware. Thus we list the main metrics of rest works [8]–[10] as Table. I shows.

We argue that compared with above works, our work integrate reputation with sharding-based blockchain, and explicitly leverage the benefits brought by reputation on incentive scheme, throughput and security level to fully realize the potential of sharding technique.

In terms of throughput, existing works ignore capability differences between validators so that less-competent ones may hinder the overall performance. Our system leverages reputation to characterize such heterogeneity and elect the competent nodes as shard leaders to improve the throughput of Raft consensus. It is worth mentioning that the time complexity of Rapidchain is the smallest among all the works. However, the cost is to generate at least three more extra transactions for one cross-shard transaction. Considering the majority of the transactions are cross-shard transaction, such solution means only one fourth of their throughput is used for the transactions that sent by clients.

In terms of incentive mechanism, some works [9], [10], [31] do not consider the incentive mechanism. Other works [8], [11] only give rewards to block miners, which causes large income variance for miners with modest computational power [39]. RSCoin [7] resorts to a central bank, which is undesired in most blockchains. Our system allocates rewards based on validators' reputations, which recognizes every devoted contribution and incentivizes validators to do their best.

In terms of security, almost all these systems adopt random sharding and leader selection except RapidChain [10]. RapidChain proposes a coarse committee reconfiguration built on Cuckoo rule, which simply regards the validators joined previous epoch as active nodes and the new joined validators as inactive nodes. Thus, their system will degrade to the random sharding in Camouflage attack in which the adversary is active only a short time in every epoch. Our scheme further utilizes reputations to establish a more balanced (*i.e.*, each shard has similar proportions of active/inactive, honest/malicious validators) sharding scheme, so that it is difficult for attacker to take control of one shard. The Sec. V shows RepChain only degrades to the random sharding in the worst case - *Observe-Act Attack* which means it is more secure in *Camouflage attack* than RapidChain.

VII. DISCUSSION

In this section, we discuss the future works.

A. Slowly Adaptive Attacker

Slowly adaptive attacker which the attacker determine the victim between epochs but cannot change this set within the epoch is a common threat model in other sharding blockchain

system [8]–[10]. However, such attacker could easily defeat the reputation based blockchain system such as RepChain as well as other state-of-the-art works [5], [16], [17]. Such attacker seems too strong since it consider all the victims have the same security level. In this paper, we think a more realistic model that different validators may have the different security level. Thus, we consider a more realistic threat model in Sec III. In the future work, we would like to solve this problem by adopting some privacy-preserving technologies such as zero-knowledge proof.

B. Dynamic Parameters

In the worst case of the current RepChain, an advanced *observe-act attack* degrades the system security to the same as the random-based solutions. However, we can improve it via dynamically adjusting some system parameters. Specifically, RepChain could adjust the reputation score formula, the sliding window and the sharding scheme dynamically based on some randomness. Therefore, the distribution of reputation scores would change in every epoch with some randomness, which effectively defends the observe-act attacks.

C. Reputation Scheme and Incentive Analysis

RepChain focus on filling the gap on utilizing reputation in a committee-based sharding blockchain to gain the benefits on throughput, security and incentive. In current stage, we prefer to adopt a simple reputation score formula to realize our core idea. Recent reputation-based blockchains [5], [17] have proposed different reputation schemes such as attenuation in reputation decaying. Moreover, it is non-trivial to derive the optimal parameters to provide the best incentives to the validators in the RepChain due to: (i) there could be three kinds of validators in the system: honest and rational validators, honest but not rational validators and malicious validators; (ii) it is challenging to model the sharding scenario since the validators will change to different shards in every epoch. Thus, we leave the formal analysis of incentive mechanism to our future work.

VIII. Conclusion

This paper proposes RepChain, a reputation-based secure, fast, and high incentive blockchain system via sharding. RepChain leverages reputation scores which describe the heterogeneity among validators and provide the incentive mechanism. RepChain is the first sharding blockchain with double-chain architecture. For transaction chain, a Raft-based synchronous consensus which achieves high throughput has been presented. For reputation chain, CSBFT that combines collective signing and RapidChain's consensus has been utilized to achieve a consensus on the reputation score. The reputation chain supports the high throughput transaction chain with a moderate generation speed. To boost the system throughput and enhance its security, a reputation based sharding and leader selection scheme has been proposed. To analyze the security of RepChain, we propose a recursive formula to calculate the epoch security within only $\mathcal{O}(km^2)$ time. The evaluation shows RepChain have a good performance under different threat models. It can also enhance both the throughout and security level and provide an incentive mechanism to the existing sharding-based blockchain system.

Acknowledgment

This work was supported in part by the RGC under Contract CERG 16204418, R8015, and the Guangdong Natural Science Foundation No. 2017A030312008.

References

[1] J. Rivera and R. van der Meulen, "Gartner says 4.9 billion connected "things" will be in use in 2015," Gatner. Available: : http://www.gartner.com/newsroom/id/2905717, 2014.

[2] Y. Lu and L. Da Xu, "Internet of things (iot) cybersecurity research: a review of current research topics," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2103–2115, 2018.

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

[4] Z. Yang, K. Yang, L. Lei, K. Zheng, and V. C. Leung, "Blockchain-based decentralized trust management in vehicular networks," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1495–1505, 2018.

[5] J. Huang, L. Kong, G. Chen, L. Cheng, K. Wu, and X. Liu, "B-iot: Blockchain driven internet of things with credit-based consensus mechanism," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2019, pp. 1348–1357.

[6] "Vitalik — ethereum en route to a million transactions per second," Brave New Coin. Available: https://bravenewcoin.com/insights/vitalik-ethereum-en-route-to-a-million-transactions-per-second, 2018.

[7] G. Danezis and S. Meiklejohn, "Centrally banked cryptocurrencies," *arXiv preprint arXiv:1505.06895*, 2015.

[8] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 17–30.

[9] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "Omniledger: A secure, scale-out, decentralized ledger via sharding," Cryptology ePrint Archive, Report 2017/406, Tech. Rep., 2017.

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

[11] J. Wang and H. Wang, "Monoxide: Scale out blockchains with asynchronous consensus zones," in *16th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 19)*, 2019, pp. 95–112.

[12] F. Selnes, "An examination of the effect of product performance on brand reputation, satisfaction and loyalty," *European Journal of marketing*, vol. 27, no. 9, pp. 19–35, 1993.

[13] P. Resnick and R. Zeckhauser, "Trust among strangers in internet transactions: Empirical analysis of ebay's reputation system," in *The Economics of the Internet and E-commerce*. Emerald Group Publishing Limited, 2002, pp. 127–157.

[14] L. Xiong and L. Liu, "Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities," *IEEE transactions on Knowledge and Data Engineering*, vol. 16, no. 7, pp. 843–857, 2004.

[15] R. Zhou and K. Hwang, "Powertrust: A robust and scalable reputation system for trusted peer-to-peer computing," *IEEE Transactions on Parallel & Distributed Systems*, no. 4, pp. 460–473, 2007.

[16] J. Yu, D. Kozhaya, J. Decouchant, and P. Verissimo, "Repucoin: Your reputation is your power," *IEEE Transactions on Computers*, 2019.

[17] L. Bahri and S. Girdzijauskas, "Trust mends blockchains: Living up to expectations," in *IEEE 39th International Conference on Distributed Computing Systems-ICDCS 2019*, 2019.

[18] E. K. Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford, "Enhancing bitcoin security and performance with strong consistency via collective signing," in *25th USENIX Security Symposium (USENIX Security 16)*, 2016, pp. 279–296.

[19] K. Iwanicki, "A distributed systems perspective on industrial iot," in *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2018, pp. 1164–1170.

[20] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, "Bitcoin-ng: A scalable blockchain protocol." in *NSDI*, 2016, pp. 45–59.

[21] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Annual International Cryptology Conference*. Springer, 2017, pp. 357–388.

[22] B. David, P. Gaži, A. Kiayias, and A. Russell, "Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2018, pp. 66–98.

[23] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th Symposium on Operating Systems Principles*. ACM, 2017, pp. 51–68.

[24] "Iota." Available: https://www.iota.org/.

[25] Y. Sompolinsky, Y. Lewenberg, and A. Zohar, "Spectre: A fast and scalable cryptocurrency protocol." *IACR Cryptology ePrint Archive*, vol. 2016, p. 1159, 2016.

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

[27] J. Poon and T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments," 2016.

[28] M. Green and I. Miers, "Bolt: Anonymous payment channels for decentralized currencies," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 473–489.

[29] "Lightning network ddos sends 20 percent of nodes down," Trustnodes. Available: https://www.trustnodes.com/2018/03/21/ lightning-network-ddos-sends-20-nodes, 2018.

[30] T. Ruffing, A. Kate, and D. Schröder, "Liar, liar, coins on fire!: Penalizing equivocation by loss of bitcoins," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015, pp. 219–230.

[31] H. Dang, T. T. A. Dinh, D. Loghin, E.-C. Chang, Q. Lin, and B. C. Ooi, "Towards scaling blockchain systems via sharding," in *Proceedings of the 2019 International Conference on Management of Data*. ACM, 2019, pp. 123–140.

[32] J. Chen, S. Yao, Q. Yuan, K. He, S. Ji, and R. Du, "Certchain: Public and efficient certificate audit based on blockchain for tls connections."

[33] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *2014 {USENIX} Annual Technical Conference ({USENIX}{ATC} 14)*, 2014, pp. 305–319.

[34] I. Abraham, S. Devadas, D. Dolev, K. Nayak, and L. Ren, "Efficient synchronous byzantine consensus," *arXiv preprint arXiv:1704.02397*, 2017.

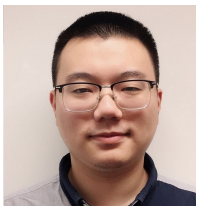
[35] "Feedback scores, stars, and your reputation," eBay. Available: https://pages.ebay.ca/help/feedback/scores-reputation.html, 2018.

[36] "The go programming language." Available: https://golang.org/.

[37] "golang-x-crypto/cosi." Available: https://godoc.org/github.com/bford/golang-x-crypto/ed25519/cosi.

[38] S. Liu, P. Viotti, C. Cachin, V. Quéma, and M. Vukolic, "Xft: Practical fault tolerance beyond crashes." in *OSDI*, 2016, pp. 485–500.

[39] L. Luu, Y. Velner, J. Teutsch, and P. Saxena, "Smartpool: Practical decentralized pooled mining," in *26th {USENIX} Security Symposium ({USENIX} Security 17)*, 2017, pp. 1409–1426.

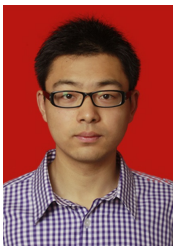


Chenyu Huang received the Ph.D. degree from Hong Kong University of Science and Technology, Hong Kong in 2020 and the BS degree in computer science from Wuhan University in 2015. He is currently a Postdoc in the Hong Kong University of Science and Technology. His research interests include mobile computing, IoT security and blockchain.

Zeyu Wang received the PhD degree from the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology. And he received the BS degree in computer science from Shanghai Jiao Tong University, in 2013. He is currently an algorithm engineer at TuSimple, Beijing. His research interests include mobile sensing, IoT, visible light communication and autonomous driving.

Huangxun Chen received the Ph.D. degree from Hong Kong University of Science and Technology, Hong Kong in 2020 and the B.S. degree from Shanghai Jiao Tong University, Shanghai, China in 2015, both in computer science. She is currently a researcher in Theory Lab of 2012 Labs, Huawei.

Qiwei Hu received a B.S. degree in applied mathematics from the Huazhong University of Science and Technology (HUST), Wuhan, P. R. China in 2017. He is currently working toward the Ph.D. degree in the School of Electronic Information and Communications, HUST. His research interests include fault tolerance technologies, consensus protocols and IoT blockchain.



Wei Wang (S'10-M'16-SM'20) is currently a full professor with the School of Electronic Information and Communications, Huazhong University of Science and Technology. He received the Ph.D. degree from the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology. His research interests include PHY/MAC design and mobile computing in wireless systems. He has published 2 books and 70 refereed papers in international leading journals and primer conferences. He is the inventor of 3 US and 20 Chinese patents. He won the best paper award in IEEE ICC 2019. He was selected as Young Elite Scientist Sponsorship Program, China Association for Science and Technology, and Hundred Talents Program, Hubei Province, China. He served on TPC of INFOCOM, GBLOBECOM, ICC, etc. He served as Editors for IEEE Access, IJCS, China Communications, and Guest Editors for Wireless Communications and Mobile Computing and the IEEE COMSOC MMTC COMMUNICATIONS.

Qian Zhang(M'00-SM'04-F'12) joined Hong Kong University of Science and Technology in Sept. 2005 where she is now Tencent Professor of Engineering and Chair Professor of the Department of Computer Science and Engineering. She is also serving as the co-director of Huawei-HKUST innovation lab and the director of digital life research center of HKUST. Before that, she was in Microsoft Research Asia, Beijing, from July 1999, where she was the research manager of the Wireless and Networking Group. Dr. Zhang has published more than 400 refereed papers in international leading journals and key conferences. Her current research interests include Internet of Things (IoT), smart health, mobile computing and sensing, wireless networking, as well as cyber security. She is a Fellow of IEEE. Dr. Zhang is serving as Editor-in-Chief of IEEE Trans. on Mobile Computing (TMC). She is a member of Steering Committee of IEEE Infocom. Dr. Zhang received the B.S., M.S., and Ph.D. degrees from Wuhan University, China, in 1994, 1996, and 1999, respectively, all in computer science.

Xia Guan received the M.S. degree in Financial Engineering from the Nation University of Singapore, Singapore in 2018 and B.S. degree in Economics, Jinhe Center for Economic Research, Xi'an Jiaotong University, Xi'an, China in 2016. He is currently the Chairman and Managing Director of the Oasis Future (Shenzhen) Holdings, Shenzhen, China.