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

Chenyu Huang*†, Zeyu Wang*†, Huangxun Chen*, Qiwei Hu‡, Qian Zhang*, Wei Wang‡, Xia Guan§
*Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong
‡School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan, China
§Oasis Futrue (Shenzhen) Holdings, Shenzhen, China
E-mail: chuangak@connect.ust.hk, zwangas, hchenay@cse.ust.hk, hu2013qiwei@hust.edu.cn
qianzh@cse.ust.hk, weiwangw@hust.edu.cn, gx@oasisfuture.com
†Co-primary Authors

***Abstract*—In today's blockchain system, designing a secure and high throughput on par with centralized payment system is a difficulty task. Sharding is one of the most worth expecting technologies to improve the system throughput when maintain high security level. However, the previous works have two main limitations: Firstly, the throughput of their random-based sharding system is not high enough due to not leveraging the heterogeneity among validators. Secondly, the incentive mechanism can be a huge overhead on their system without an appropriate scheme. We propose RepChain, a reputation-based secure, high incentive and fast blockchain system via sharding. RepChain utilizes reputation to explicitly characterize the heterogeneity among the validators and lay the foundation for the incentive mechanism. We novelly propose the double-chain architecture that including transaction chain and reputation chain. For transaction chain, a Raft-based Byzantine fault tolerant synchronous consensus with high resiliency and high throughput has been presented. For reputation chain, the collective signing has been utilized to achieve consensus on the reputation score and support the 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. Further more, we implement and evaluate RepChain on 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

In recent years, blockchain technology has shown great potential to revolutionize the global financial ecosystem [1], [2]. Technically, blockchain is a decentralized and public digital ledger which records data in a large number of distributed nodes. In ideal cases of such intermediary-free system, consensus, *i.e.* agreement on data is expected to be achieved by the honest majority efficiently (*i.e.*, high throughput) and resistant to retroactive modification by malicious users (*i.e.*, high security). Existing blockchain systems are still far from meeting above expectations. For example, Bitcoin and Ethereum can only handle 7 and 20 transactions per second respectively [3], while their centralized counterparts, PayPal and Visa can deal with hundreds to thousands of transactions per second [4].

The fundamental limitation is attributed to global consensus requirement, *i.e.*, every data should be validated by all validators. Thus, the validation workload increases with the growing number of the validators, but the system capacity remains unchanged. Based on above analysis, sharding is an intuitive solution for throughput improvement, where the validators are separated into several groups so that transactions can be processed in parallel. In the state-of-the-art literature, several sharding-based protocols including RSCoin [5], Elastico [6], OmniLedger [7] and RapidChain [8] have been proposed to address the trade-off between throughput and security.

However, most existing sharding systems ignored the following important aspects of a practical blockchain system. Firstly, they did not consider the heterogeneity among validators. Except distinction on honest/malicious attributes, the validators are regarded as the same. However, validators in a practical blockchain system present much difference in terms of computing capability, communication bandwidth and historic behaviors. Thus, when applying random sharding [5]–[7] or simple balanced random sharding [8] in practical system, those less-competent validators become bottlenecks and hamper system throughput. Secondly, their protocols lack proper incentive mechanisms for validator activation and retention. They simply allocate rewards to the shard leaders [6] or even don't mention reward allocation [7], [8]. A trivial incentive mechanism in the PBFT-based protocol is to reward the validators for their liveness. However, this method requires an appropriate behavior monitor scheme and extra rounds of PBFT to forge agreement on this liveness condition, which brings huge overhead. RScoin [5] allocates rewards to the active users with the help of a trustable node, the central bank, which is unrealistic in most blockchain system.

In this paper, we propose reputation concept in the context of sharding-based system to jointly address the above two issues in practical systems. Reputation established on the 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 [9], [10]; nodes in Peer-to-Peer systems establish their reputations on active participation in file sharing to obtain other nodes' cooperation [11], [12]. Inspired by above observations, we design RepChain, a reputation-based high throughput, secure and high incentive blockchain system, where the reputation scores of validators are measured based on their behaviours.

The reputation scores are good indicators of validators' capability and reliability, and also facilitate the design of incentive mechanism, which is beneficial to sharding systems in many aspects. In terms of throughput, the shard leader selection can utilize reputation scores to elect a high capability leader with higher possibility to boost system throughput. In terms of incentive, reputation-based reward scheme can greatly incent the validators to do their best, i.e., an honest and competent validator deserves more rewards for its contribution to the system. In addition, reputation scores can enhance the system security. Reputation-based sharding and leader

selection scheme can balance the total reputation score across different shards, so that each shard has similar proportions of active, inactive, honest and malicious validators. Thus, it is more difficult for the malicious users to take control of one shard. Generally, reputation-based sharding system is more secure than random-based ones. It only degrades to a random-based one with an advanced attacker in the worst case.

To achieve the desired system, two aspects of challenges as follows should be addressed. Firstly, we should design an efficient consensus protocol for transactions, which explicitly consider heterogeneity among the validators, to avoid that the validators with lower processing capability and more security breaches become the bottleneck. Secondly, we should design an efficient consensus protocol for reputation scores. On one hand, all the validators should calculate and reach a consensus about the reputation scores to incent the honest and competent validators and also secure the system. On the other hand, the reputation scoring should occur at a moderate frequency without bringing too much overhead to the system.

Thus, RepChain novelly proposes the double-chain architecture to address above challenges. Specifically, RepChain maintains transaction chain and reputation chain separately. Firstly, for transaction chain, RepChain revises and amends the traditional Raft algorithm to a Byzantine fault tolerant synchronous consensus protocol for sharding-based blockchain, which achieves transactions agreement efficiently and securely. Compared with other works, our scheme eases the workload of the least capable validator to improve the system performance by incenting competent shard leader to contribute more. An honest and competent validator has more chance to be selected as a shard leader, and the reputation-based scheme will ensure that being honest is its best strategy. Secondly, RepChain proposes a new reputation chain to monitor validators' behaviour and achieve consensus on reputation scores efficiently. Besides the hashes of multiple confirmed transaction blocks and the reputation scores of all validators, a reputation block also includes a signature generated via collective signing [13]. One signature in a reputation block provides authenticity proof for multiple transaction blocks between last reputation block and the current one. Such compression enables a reputation chain with moderate generation speed to support a high throughput transaction chain. It is worth mentioning that existing sharding-based systems can also benefit from our core idea that leveraging the reputation chain with moderate generation speed to support the high throughput transaction chain. With a little modification on their consensus and sharding scheme, the incentive and security property can be both enhanced.

To evaluate RepChain, we analyze the security, the performance and the incentive mechanism of RepChain. Different from previous works [7], [8] that only estimated the failure probability of one epoch, we propose a new recursive formula to give the exact solution in time complexity of $\mathcal{O}(km^2)$. Furthermore, we implement RepChain and test it under Amazon Web Service with 900 instances of which 450 are from US West and 450 are from US East. It is different from Elastico that put all the instances in US West or Omniledger and RapidChain that put all the nodes in one local area where the latency is controlled well. The 900 instances simulate 1800 nodes and the shard size is 225 to maintain a high security level. The result shows our system achieves the throughput of 6852 transactions per second (tps) and the user-perceived latency of 58.2 seconds in average. We also test RepChain under three different threat models and different validator capabilities. The results present RepChain can enhance the throughput and security level of sharding-based blockchain system by leveraging reputation scores.

In summary, this paper makes the following contributions:

1) RepChain is the first to utilize reputation scores in sharding-based system, which explicitly characterize the heterogeneity among the validators and lay the foundation for the incentive mechanism.
2) RepChain is the first sharding-based system with double-chain architecture which enable a reputation chain with moderate generation speed to support a high throughput transaction chain.
3) We propose a Raft-based Byzantine fault tolerant synchronous consensus for transaction chain to achieve high throughput and high resiliency.
4) We propose the reputation-based sharding and leader selection scheme to boost the system throughput and enhance its security.
5) We propose a recursive formula to analyze the epoch security precisely in time complexity of $\mathcal{O}(km^2)$.
6) We implement the proposed system and conduct extensive evaluations on Amazon Web Service to validate the effectiveness of our design.

II. Background and Related Work

In this section, we introduce the blockchain scalability problem, the related sharding-based blockchain systems and the reputation-based system.

A. Blockchain Scalability

It is well known that scalability is an unsolved issue in the existing blockchain systems. In the state-of-the-art literature, Bitcoin-NG [14] boosts system throughput by allowing the shard leaders in an epoch to append more blocks than those of Bitcoin. However, the computation overhead of each node also increases substantially with the boosted throughput. ByzCoin [13] 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. Lightning network [15] and Bolt [16] both utilize the off-chain payment to deal with scalability. Lightning network [15] allows the participants to transfer micro-payments through a payment channel without appending it onto the main chain. Bolt [16] implements a privacy-preserving payment channel with zero-knowledge proofs. As an enhancement to the main chain, the off-chain solutions could increase the system throughput significantly. However, the off-chains generally do not have the same security guarantees as the main chain, which results in vulnerability to various attacks [17], [18].

B. Sharding-based Blockchain

A sharding-based blockchain could have higher throughput with more validators in the network. By distributing the transactions to different shards, the total throughput is equal to the product of the in-shard throughput and the number of shards. According to the literature, several high throughput and secure systems have been proposed. RSCoin [5] implemented a centrally banked system via sharding. A simple Two Phase Commit (2PC) has been utilized between the user and a set of mintettes from one shard. Elastico [6] proposed the first sharding-based consensus protocol for public blockchain. They

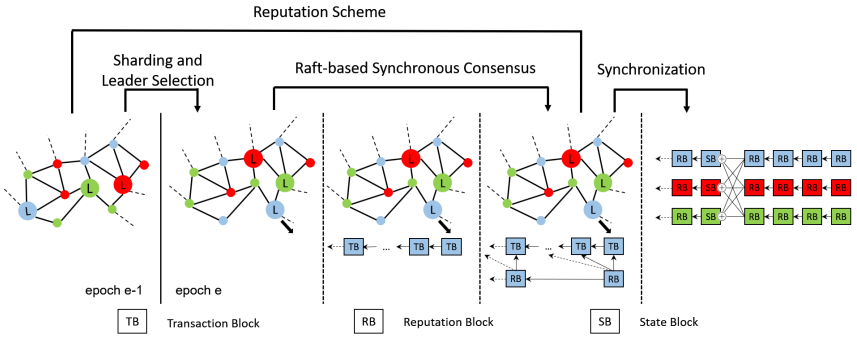

Fig. 1. RepChain Overview

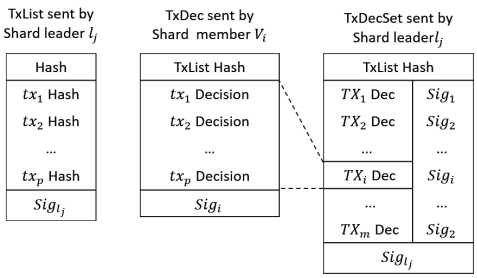

Fig. 2. Data structure of TxList, TxDec and TxDecSet

achieved a near-linear computational scalability which can tolerates 1/4 fraction of corruptions among all the nodes. In their system, PoW was adopted to establish the identities of the validators, and BFT protocol was used within the shard. OmniLedger [7] shards and selects a leader via a verifiable random function (VRF). A variant of ByzCoin [13] has been used to improve the throughput. RapidChain [8] improves the throughput and achieves a total resiliency to a 1/3 of corruptions from all nodes via a synchronous protocol.

In terms of incentive mechanism, RSCoin implemented an incentive mechanism with the help of a central bank, which is unrealistic in most blockchain systems. The remaining three works [6]–[8] do not consider the incentive mechanism. Also for all the members to monitor the behaviours of each other and to reacha a consensus on these behaviours will incur extra overheads. Compared with them, our reputation scheme rewards validators based on their behaviours. Considering the throughput, these works do not consider the capability differences between validators and thus the performance is the average of all the validators. Our system adopts the double-chain architecture which generates the reputation block at a moderate speed without too much overhead to record the reputation of all the validators. Also the reputation leader selection will elect a high capability leader with higher possibility. The raft-based consensus ultlizes the high capability to improve the throughput. From the security aspect, almost all these systems are based on random sharding and leader selection of which the security level is lower than the reputation-based one. RepChain only degrades to the random-based one in the worst case where an advanced attacker firstly observes the reputation distribution of all validators, and then acts with the same probability as the distribution. Moreover RapidChain proposed to separate the active and inactive validators, and balance their proportions among the groups. Their classification method is quite coarse which simply regards half of the validators as active nodes and the remaining half as inactive nodes. Moreover, the classification is determined by the reference committee generated at the very beginning, which could result in security problems without a proper update scheme for the reference committee.

C. Reputaion and Blockchain

In traditional P2P network, it is common to leverage reputation as an incentive mechanism [11], [12]. In the state-of-the-art literature in the blockchain area, CertChain [19] proposed a Dependability-rank based consensus as well as an incentive mechanism that takes the economic benefits and misbehavior into consideration. However, their design is tailored for the certifying authorities (CA) but not for the scalability issue. RepuCoin [20], a non peer-reviewed work, tried to use reputation in the blockchain. They measured the reputation of the users based on their behaviours, and proposed a reputation-based weighting scheme consensus to overcome the computation cost in PoW. However, there are some critical defects in their system: (1) RepuCoin used a committee-based consensus that only rewards the keyblock miner and the consensus group, which leads to an unstable income for the participants. (2) Their reputation scores are based on the total amount of valid work over time. The cumulative reputation scores together with the committee-based consensus could result in a severe monopoly problem and double spending attacks if the high reputation users collude with each other. Moreover, it is difficult for the new validators to join the consensus group and get the reward. (3) Their system could be defeated if an attacker joins the system in the very beginning.

III. Model and Overview

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

A. Notation

We assume there are n validators $V^n = \{v_1, v_2, ..., v_n\}$ and k shards denoted as $C^k = \{C_1, C_2, ..., C_k\}$. Thus, there are $m = n/k$ validators in each shard, including one leader and $m-1$ members. The reputation score of each validator is r_i. tx denotes a transaction. Each epoch e has a fixed time and each step in the consensus is finished within the time Δ.

B. Network Model

Our network model is similar to those in the existing works [6]–[8]. Specifically, it is assumed that the connections between honest nodes are well established, and the transmission between them is within Δ. The communication channel is synchronous within one shard, as we adopted the synchronous consensus. Traditionally, the drawback of such a consensus is poor responsiveness [8]. However, we adopt the method from RapidChain [8], where all the members would agree on a new Δ every week to achieve a long-term responsiveness. The other parts of RepChain are based on partially-synchronous channels with optimistic exponentially increasing time-outs.

C. Threat Model

In our protocol, we assume a Byzantine adversary who corrupts less than $g = n/3$ nodes. The corrupted nodes can collude with each other act out 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 roundly adaptive attackers in [6]–[8]. In reality, 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 corrupted nodes to 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. Therefore, this paper considers three attacker strategies as follows.

Simple Attack: The attacker does bad things continuously.

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

Observe-Act Attack: The attacker observes the reputation score distribution of the normal users. Then the attacker controls the malicious nodes to act and have the same reputation score distribution as the normal one. It indicates that the malicious nodes would have higher chances to be grouped into the same shard according to the sharding algorithm.

D. System Overview

Our system is built on double-chain architecture – a transaction chain and a reputation chain. It mainly has four components as Fig. 1 shows: sharding and leader selection, Raft-based synchronous consensus, reputation scheme, and synchronization .

Sharding and Leader Selection: All the nodes are sharded into different groups at the beginning of each epoch. Then each shard selects its shard leader.

Raft-based Synchronous Consensus: The clients send their txs to the shards which are responsible for the input UTXOs (unspent transaction output). Then the shard runs an intra-shard Raft-based synchronous consensus to generate both the transaction blockchain at high speed and the reputation blockchain at moderate speed. Moreover, the system adopts an atomic cross-shard protocol for cross-shard transactions.

Reputation Scheme: The reputation scheme enhances the previous two components. Given the behaviour of all validators by the consensus, all validators can calculate and reach a consensus on the reputation scores. The cumulative reputation scores support better sharding and leader selection. As a result, all three components guarantee a secure, high-incentive and fast blockchain.

Synchronization: At the end of the epoch, each shard generates a state block to conclude the transaction blockchain and the reputation blockchain. The nodes synchronize and update the stored reputation scores based on the state blocks from all the shards, and then the system begins the next 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 and inactive and honest and 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 of being selected as the shard leader.

More specifically, when a new epoch e begins, all the validators will be sharded into different groups. For each group, a shard leader will be selected based on their cumulative reputation score R^w over previous w epochs as Alg. 1 shows. Specifically, the validators use previous state block hashes from all the shards to generate the random generator RNG (Line 2 in Alg. 1, SBH_i^{e-1} denotes the hash of shard i's state block at epoch $e-1$). R^{sort} denotes the validators' reputation scores sorted in descending order(Line 3). Next, each validator is randomly assigned to a shard of 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. First, the validators with reputation scores higher than the median have chances of being selected as the leader. Each score of these validators will divide a number that is randomly generated based on the same seed $Seed^e$, then the validator with the minimum result is selected as the leader. (Line 11-17)

Algorithm 1 Sharding and Leader Selection Algorithm

Input:
A random $Seed^e = \{SBH_1^{e-1}, SBH_2^{e-1}, ..., SBH_k^{e-1}\}$;
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 sequence $\{t_1, t_2, ..., t_j\}$ which makes the subset $\{C_{t_1}, C_{t_2}, ..., C_{t_j}\}$ of C satisfies $|C_{t_1}| = |C_{t_2}| = ... = |C_{t_j}| = \min(|C_1|, |C_2|, ..., |C_k|)$.
6: Generate a random integer x from RNG.
7: Assign validator v_g to C_{t_u} that $u = x \mod j$.
8: **end for**
9: **for** each shard $C_i \in C$ **do**
10: rm = median of the subset of R^w that belongs to C_i
11: **for** each validator $v_j \in C_i$ **do**
12: **if** $rm \leq r_j^w$ **then**
13: Generate a random float $0 \leq y \leq 1$ from RNG.
14: $p_{i,j} = y / r_j^w$
15: **else**
16: $p_{i,j} = +\infty$
17: **end if**
18: **end for**
19: $l_i = v_j$ where $p_{i,j} = \min(p_{i,1}, p_{i,2}, ...p_{i,m})$
20: **end for**

B. Consensus

Inspired by Raft [21], XFT [22] and the protocol proposed by Ren *et al.* [23] and RapidChain [8], we propose a Raft-based synchronous consensus to achieve high throughput and

1/2 resilience within a shard. Our consensus constructs two blockchains in each shard: the transaction blockchain and the reputation blockchain. The transaction blockchain includes all the transaction txs that belong to the shard, while the reputation blockchain includes the reputation scores that all the validators earned in this epoch.

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. All the clients will send their transactions to the shards responsible for the input UTXOs and output UTXOs listed in these transactions. The shard i will validate the tx if its ID mod k is equal to i. The input UTXOs may come from different shards. Thus, our system should handle both intra-shard consensus and 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.

1) Intra-Shard Consensus: The intra-shard consensus mainly contains five synchronous steps. The related data structures are shown in Fig. 2. TxList refers to the transaction list including the transaction hashes in ascending order and the signature of the leader. TxDec refers to transaction decisions given by the member, which contains the decisions in the same order as the TxList and the signature. TxDecSet refers to the transaction decision set including all the TxDecs from shard members. Before the consensus protocol, the leaders will gather the transactions and broadcast them to all the validators. Thus, the validators can do a consensus on the TxList to reduce the bandwidth cost. We also remark that all the messages will be signed by the sender with their public key during the consensus so that the sender and the integrity of the message can be verified.

Firstly, the leader signs and sends a transaction list (TxList), which contains a list of transactions' IDs to all shard members. Then each validator checks all the transactions in the TxList and makes a decision, *Yes*, *No* or *Unknown*. *Unknown* is given to avoid punishment on reputation score when it cannot handle so many transactions due to the hardware limitations. Secondly, the validators sign and send the decisions (TxDec) in which the txs are ordered the same as the TxList back to the leader. After the leader collects all the TxDec within the shard, it will generate a transaction block (TB) including the transactions with more than half of *Yes*. Thirdly, the leader sends the TB as well as the transaction decision set (TxDecSet) including all the validators' TxDecs to all shard members. Fourthly, each validator checks the TxDecSet and the TB from the leader. If the leader does any of the bad things described in Sec. V-B, the honest validators will send *Warning* with its signature to each other. When half of the validators send out *Warning*, the honest validators can begin rolling: kicking out the current leader, dropping the current incorrect transaction block, clearing the cumulative reputation score of the leader, and reselecting a leader. The consensus above indicate the transaction block has been confirmed within the shard.

The fifth step is for reputation chain. After the generation of several transaction blocks without warning, the validators can calculate the reputation scores to generate the Reputation Block (RB) based on the TxDecSet and the TB, and achieve a consensus via Collective Signing. Specifically, the leader and the validators run two sequential rounds including announcement and commitment, challenge, and response to sign on the messages in RB. The RB contains the reputation scores of all the validators and the TB hash.

The intra-shard consensus separates the confirmation of the transaction validation within and out of the shard to reduce the bandwidth cost. If no *Warning* is received, the consensus on the TB has been achieved by all the validators within the shard. Besides, a validator can send TxList and TxDecSet to prove whether a tx is valid or not to the client. However, it will cost a lot of bandwidth to transmit the TxDecSet that contains the signatures from all the validators. Moreover, the cross-shard transaction makes the condition worse, because all input shards should send such proof to the client. To reduce the bandwidth cost, we adopt the collective signing [13] in which the amount of signatures will only be $1/m$ of those in the TxDecSet. Thus, only the output shard needs to transmit one RB with a collective signature to the client as a proof.

2) Cross-Shard Transactions: Omniledger [7] introduces the Byzantine Shard Atomic Commit (Atomix) protocol that supports secure cross-shard transactions. We propose an enhanced protocol to overcome two drawbacks of their protocol: (1) Atomix requires the client to be active to help the input shards send the proof-of-acceptance to the output shards. Such a requirement can be satisfied in reality because the client may be offline. (2) For each tx, the client should send one proof-of acceptance to the output shard which costs lots of bandwidth. Also, it brings more communication overhead on the client. In RapdiChain [8], they propose to batch the transactions and separate the multiple input UTXOs of cross-shard transactions. For each cross-shard tx_a, the output shard will generate multiple transactions that only has one input UTXO and one output UTXO of the current shard. The input UTXO corresponds to one of the input shards of tx_a. However, their method creates more transactions so that the transaction blocks become larger. To overcome these disadvantages, we combine their method and propose a new cross-shard protocol.

In RepChain, the input shard leader sends the TxList and TxDecSet to the relevant input and output shards' leaders. Then the leader distributes the TxList and TxDecSet to all the validators within the shard. It is worth mentioning that the TxDec from validators contains multiple signatures and each will be used to sign one subset of the TxList. The txs in one subset are sent to the same output shard to reduce the bandwidth cost. For a specific cross-shard tx, the shard will accept if its UTXOs are not used or locked otherwise the tx will be rejected. Then both the input and output shards will lock it and its UTXOs. When the output shard receives the proof of accept - TxDecSets containing the input UTXOs from all the relevant input shards, it adds the tx onto their TB. When the input shard receives the proof of accept from other relevant input shards, it will release the tx and its UTXO. If one of the input shards rejects the tx, then the relevant input and output shards will abort the tx.

3) Optimization with Parallelizing: To enhance the throughput of RepChain, we adopt parallelizing similar to Omniledger, i.e., the transactions would be handled in different blocks in parallel if they do not conflict with each other. More specifically, each TxList, TxDecSet, and TB include an iteration number to indicate the round. The leader can send a new TxList once it receives all TxDecs of previous rounds from the validators. Recalling that our consensus protocol is a synchronous one, it guarantees that before receiving the TxList of the current round, the honest majority have finished the job

of sending the TxDec of the previous round to the leader. Thus the validators can determine whether the new TxList is valid or not.

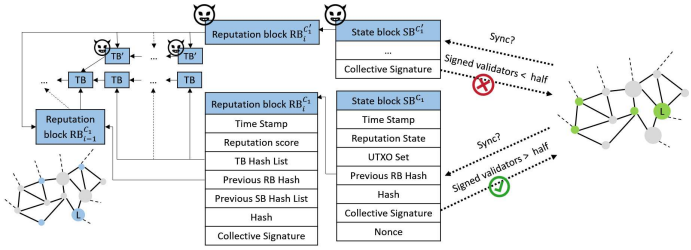

Fig. 3. This figure shows the structure of the reputation block (RB) and state block (SB) as well as the working procedure of colective signing in the synchronization process.

C. Reputation Scheme

Recalling that we use double-chain architecture, previous section explained the generation of the transaction chain in detail. In this section, we illustrate the design of the reputation scheme which includes the calculation of the reputation score and the generation of the reputation blockchain. Our proposed reputation scheme 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. Inspired by PeerTrust [11], 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. $T(j)$ is the value of transaction j to prevent the case where some validators are honest on small transactions but dishonest on a large transaction. The scaling factors $S(j)$ are used to reward or penalize different behaviours differentially. eBay [24] 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 behaviours to make the punishment for dishonest behaviours larger than the reward for honest ones. Furthermore, an illegitimate tx being passed is more dangerous than a legitimate tx being aborted. Thus, if a validator gives a *"Yes"* while most of the others give a *"No"*, the validator will be punished more than the situation where it gives a *"No"* while the others gives a *"Yes"*. 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 utilized collective signing to generate the RB, the structure of which is shown in Fig. 3. RB contains the reputation score, the confirmed TB list, the previous RB and the collective signature. It also contains the previous state blocks from all shards if it is the first block of one epoch. The collective signature contain the information on the number and identity of validators who agree on the block. Other shards can check the collective signature, and accept it if more than half of the validators sign on the block. Thus, the user-perceived latency of a transaction is almost the same as the time interval between two RBs. Within one epoch, the shard will generate a few of RBs and form a reputation blockchain. The cumulative reputation scores over the sliding window on w epochs can easily be calculated via traversing through the reputation blockchain and used in sharding and leader selection.

Moreover, the malicious validators can fork the transaction blockchain and the reputation blockchain. However, as shown in Fig. 3, 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 utilize collective signing to generate a state block (SB). Then the validators use PoW to generate the nonce in SB, which brings the randomness for sharding and leader selection in the next epoch.

The structure of SB is shown in Fig. 3. It contains the overall reputation scores of the validators over the past w epochs and the UTXO set of the clients by this epoch. The UTXO set is the set containing all the UTXO that are generated within this epoch. We also combine the UTXO 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 UTXO. 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. Previous sections detail all of the parts. 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. The transaction chain achieves the consensus within the shard via our intra-shard consensus protocol at a fast speed to provide a high throughput. The reputation blockchain is created at a moderate speed to record the behaviour based on TxDecSet and TB. Since the reputation block contains the collective signature, it can provide the proof of accept to the client. Moreover, the reputation blockchain supports a balanced sharding and leader selection, incents the validators for working hard and enhances the throughput. As a result, such a design can provide a secure, fast and high incentive sharding based blockchain system

V. System Analysis

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

A. Epoch Secuirty

For epoch security, previous work [8] 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 $F(x, 1) = C_m^x \mathbb{1}_{x \leq d}$. 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 ($a = pk + q, 0 \leq q \leq k - 1$) malicious nodes are exposed, they will be allocated into k shard equally accroding 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:

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

where $l = k - y$, $u_l = \mathbb{1}_{l \in [0, q-1]}$, $v = max\{0, q - l - 1\}$ and $F(x, 1, 0) = C_m^x \mathbb{1}_{x \leq d}$. 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 4 shows the failure probability with different a. 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: For a transaction block, we first prove a malicious leader cannot construct a malicious transaction block with a legal TxSecSet. A malicious leader cannot modify TxDecSet because it contains the signature of shard members. However, the leader may remove one TxDec of a certain validator from TxDecSet on purpose. Furthermore, the malicious leader may add an illegitimate tx without half of the *"Yes"* to TB, drop a legitimate tx on purpose, or remains silent for a long time and so on. The shard members will broadcast *"Warning"* signals when they find the above malicious behaviours. Once half of the shard members send out a *"Warning"*, the rolling scheme will be executed to prevent the malicious leader. The cumulative reputation scores of the malicious leader would be cleared, thus it will take a long time and hard work before the malicious node would be selected as shard leader again. Next, we prove a malicious member cannot construct a malicious transaction block when there is an honest leader. Once the honest leader construct a transaction block, it will be accepted by the honest majority. No malicious member can construct a safe proposal for another transaction block because it cannot construct or modify the TxDecSet with enough "Yes" as a proof.

For the reputation block, the proof is almost the same as the transaction block. The only difference is instead of utilizing TxDecSet as a proof, the collective signature is used.

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 [8]. As we mentioned before, the expectation round for an honest leader is around two in the worst case. 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 block. If the current leader is a malicious one who wants to stop the consensus or other malicious behaviour, the rolling process will be launched and a new leader will eventually be selected. Thus a valid block will be proposed by an honest leader and accepted by the honest majority with a proof - TxDecSet. Although the proof above is for transaction block, it is also suitable for reputation block.

C. Security of Cross-shard Transaction

In cross-shard transactions, all shards are honest since the consensus has been achieved within the shard. Thus we can come to the following conclusion. Firstly, a cross-shard transaction tx will eventually be recorded in the block or be aborted. The input shards give a proof of accept or reject for the tx. One reject could abort the tx, while all accepts from the input shards could validate the tx. Otherwise, the input and output shards lock the tx and its UTXOs. Secondly, the transaction would not be recorded twice because the shard responsible for the UTXO will check whether it has been used or locked. Thirdly, if a transaction is aborted, the client could reclaim the UTXOs released by the input shards. Fourthly,

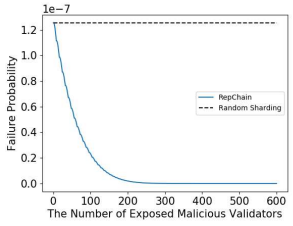

Fig. 4. The failure probability of RepChain with different number of exposed malicious validators

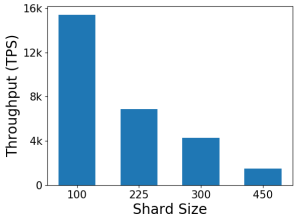

Fig. 5. The average throughput of the transactions for different shard sizes.

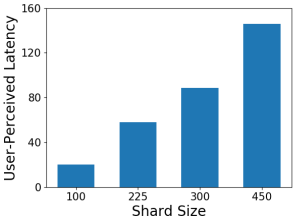

Fig. 6. The average user-perceived latency of a transaction for different shard size

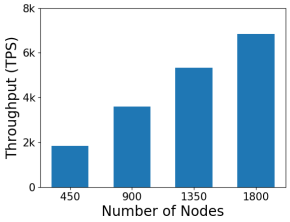

Fig. 7. Throughput scalability

the shard leader may be malicious or crash. In such a case, the rolling process will be launched and the new leader will re-check all the tx processed via the previous leader.

D. 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. 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 of size 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. In addition, if tx is a cross-shard transaction, the TxDecSet of one shard should also be transmitted to all related input and output shards. Given the assumption that the number of inputs and outputs is constant, the total communication between cross-shards for tx costs $\mathcal{O}(m^2/b)$. Therefore, the consensus complexity of one transaction tx is equal to $\mathcal{O}(m^2/b)$ for both intra-shard and cross-shard.

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 reputation 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.

E. Incentive Mechanism

RepChain provides great 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 language [25]. For collective signing, we use the code in Go cryptography libraries [26]. For the parameters in RepChain, we set the sliding window e to be 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 *No*, and -1 for an incorrect decision that the transaction should be rejected but the validator decides to give *Yes*. 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. 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.

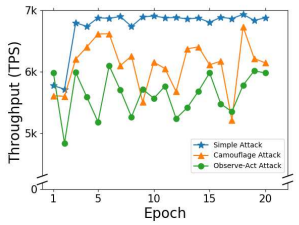

Fig. 8. The average throughput for three attack models

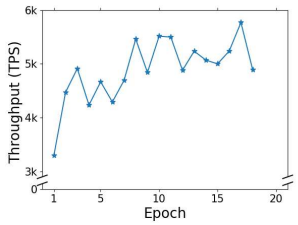

Fig. 9. The average throughput of the transactions with different validators abilities.

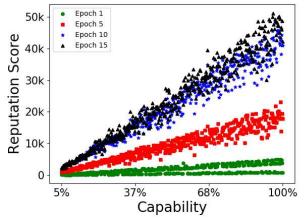

Fig. 10. The reputation score of validators with different capabilities from 5% to 100%

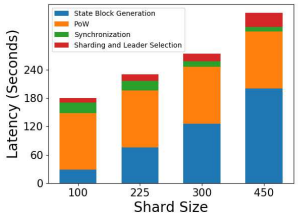

Fig. 11. The epoch transition latency

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 [6]–[8]. 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 [8]. 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. 5 and Fig. 6. 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 and 146.0s. With the increase in sharding size, the throughput will decrease and the latency will increase. The poor performance under the shard size of 600 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 transaction needs to be sent to one other after it is generated which also costs lots of time. Anyway, when satisfying the security property, the shard size of 225 has a larger throughput 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. 7. 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. 8. The first two attacks can be implemented easily. However, the performance under *Observe-Act Attack* depends on the ability of the attacker. Thus 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. For the *Simple Attack*, our system 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 throughput of RepChain is around 6000 tps. Once the leader is a malicious node, the rolling scheme 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 throughput is around 5500 tps which is worse than the previous two attack models. As we mentioned in Sec. V-A, such an attacker will cause 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-D. 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 *Unknon* 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. 9 shows the throughput enhancement benefits from our reputation scheme. It can easily be concluded that at the beginning, the throughput is less than 4000 tps due to the low capability leader. However after about 8 epochs, the throughput is stable around 5500 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. That is to say, the validators is randomly sharded into 8 groups and the leader is randomly elected. The average throughput is 3273 tps which is similar to the first epoch but much lower than the following epochs of RepChain with the reputation scheme. Fig. 10 shows the relationship between their capabilities and the reputation score in the 1st, 5th, 10th and 15th epoch. We can see the validators barely

TABLE I
COMPARISON BETWEEN REPCHAIN AND THE EXISTING WORKS.

	#Nodes	Resiliency	Complexity	Throughput	Latency	Shard Size	Balanced Sharding	Incentive Mechanism	Time to Fail
Elastico	1600	t<n/4	$\Omega(m^2/b+n)$	40 tx/s	800s	100	No	No	1 hour
OmniLedger	1800	t<n/4	$\Omega(m^2/b+n)$	3500 tx/s	63s	600	No	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	$\mathcal{O}(m^2/b+m)$	6852 tx/s	58.2s	225	Based on reputation	Yes	Depends on attacker's strategy. The worst case is the same as [8].

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 incent 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. 11 which is 180.2s, 230.8s, 274.3s and 361.0s. Among the four stages, the PoW and 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 works [6]–[8]. Table. I shows that the resiliency of RepChain is the same as RapidChain and larger than others. Our contributions are mainly on secure, incentive mechanism and throughput enhancement. Other works are based on random sharding or on simple balanced sharding [8]. RepChain shards the validators based on their reputation. As a result, the security of RepChain depends on the attacker's strategy. In the worst case, the time to fail is the same as the RapidChain under the same shard size and number of nodes. Also other works do not consider the incentive mechanism while RepChain incents the honest and hard-working validators. The complexity of RepChain is $\mathcal{O}(m^2/b+m)$ which seems the smallest. However it is worth mentioning that other works are built on the gossip protocol which brings complexity into the network communication. Our system will have the same complexity if we built our system on the gossip protocol. However, RepChain effectively utilizes the capability of different validators and enhance the throughput via a higher capability leader being elected.

VII. DISCUSSION

In this section, we discuss the future works of RepChain.

A. Information Dispersal Algorithms

In RepChain, the Δ is set to be 7.5 seconds because all the validators send the messages to each other via TCP. Such a setting results in a relatively high latency that reduces the system throughput. However, it can be improved by leveraging more efficient information dispersal algorithms, such as IDA-Gossi, an unreliable broadcast protocol with an erasure code scheme proposed by RapidChain.

B. Permissionless Property

Permissionless is another important property for a public blockchain, which allows anyone to participate in the system at anytime. Such a property can be easily integrated into RepChain as in RapidChain. More specifically, to join our system, a new node should run a PoW on the previous state block hashes and send its solution to one shard. The members of that shard will review it and add it into the state block of this epoch. Then the new node can join the system at the beginning of the new epoch after the synchronization. In our system, the new node has a relatively low reputation score compared with other normal nodes who have worked for several epochs. Thus, the join-leave attack could not threaten our system.

C. 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. More 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

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 and high resiliency has been presented. For reputation chain, the collective signing has been utilized to achieve consensus on the reputation score and support 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 good performance under different threat models. And it can enhance both the throughout and security level and provide incentive mechanism to the existing sharding-based blockchain system.

REFERENCES

[1] "How blockchain is changing finance," Harvard Business Review. Available: https://hbr.org/2017/03/how-blockchain-is-changing-finance, 2017.

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

[3] "The raiden network could allow instant transactions in ethereum," ETHNews. Available: https://www.ethnews.com/the-raiden-network-could-allow-instant-transactions-in-ethereum, 2017.

[4] "Scalability," Bitcoin Wiki. Available: https://en.bitcoin.it/wiki/Scalability, 2017.

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

[6] 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.

[7] 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.

[8] M. Zamani, M. Movahedi, and M. Raykova, "Rapidchain: A fast blockchain protocol via full sharding."

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

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

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

[16] 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.

[17] "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.

[18] 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.

[19] 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."

[20] J. Yu, D. Kozhaya, J. Decouchant, and P. Esteves-Verissimo, "Repucoin: Your reputation is your power."

[21] J. Chiefari, Y. Chong, F. Ercole, J. Krstina, J. Jeffery, T. P. Le, R. T. Mayadunne, G. F. Meijs, C. L. Moad, G. Moad *et al.*, "Living free-radical polymerization by reversible addition- fragmentation chain transfer: the raft process," *Macromolecules*, vol. 31, no. 16, pp. 5559–5562, 1998.

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

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

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

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

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