IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 27, 2019, accepted September 8, 2019, date of publication September 17, 2019,
date of current version September 30, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2942003

Trust-Based Shard Distribution Scheme for Fault-Tolerant Shard Blockchain Networks

JUSIK YUN, YUNYEONG GOH, AND JONG-MOON CHUNG, (Senior Member, IEEE)
School of Electrical and Electronic Engineering, Yonsei University, Seoul 03722, South Korea

Corresponding author: Jong-Moon Chung (jmc@yonsei.ac.kr)

This work was supported by the Ministry of Science and ICT (MSIT), South Korea, through the Information Technology Research Center (ITRC) Support Program supervised by the Institute for Information and Communications Technology Planning and Evaluation (IITP) under Grant IITP-2019-2018-0-01799.

ABSTRACT Blockchains guarantee data integrity through consensus of distributed ledgers based on multiple validation nodes called miners. For this reason, any blockchain system can be critically disabled by a malicious attack from a majority of the nodes (e.g., 51% attack). These attacks are more likely to succeed as the number of nodes required for consensus is smaller. Recently, as blockchains are becoming too large (making them difficult to store, send, receive, and manage), sharding is being considered as a technology to help improve the transaction throughput and scalability of blockchains. Sharding distributes block validators to disjoint sets to process transactions in parallel. Therefore, the number of validators of each shard group is smaller, which makes shard-based blockchains more vulnerable to 51% attacks than blockchains that do not use sharding. To solve this problem, this paper proposes a trust-based shard distribution (TBSD) scheme that assigns potential malicious nodes in the network to different shards, preventing malicious nodes from gaining a dominating influence on the consensus of a single shard. TBSD uses a trust-based shard distribution scheme to prevent malicious miners from gathering in on one shard by integration of a trust management system and genetic algorithm (GA). First, the trust of all nodes is computed based on the previous consensus result. Then, a GA is used to compute the shard distribution set to prevent collusion of malicious miners. The performance evaluation shows that the proposed TBSD scheme results in a shard distribution with a higher level of fairness than existing schemes, which provides an improved level of protection against malicious attacks.

INDEX TERMS Blockchain, sharding, trust, malicious attack, genetic algorithm, fault tolerant.

I. INTRODUCTION

Blockchain is a key technology that enables cryptocurrency (e.g., Bitcoin), smart contract services (e.g., Ethereum), and various data protection services. The data protection level of blockchains is based on the difficulty to tamper block data due to the hash chain structure. Currently, there are several technical challenges that blockchain systems are facing. According to Ethereum co-creator Vitalik Buterin, blockchain systems at most can only have two of the following three features: decentralization, scalability, or security. Especially, scalability is an indispensable technical challenge for real-time services in connection with various industrial integration of blockchain systems, because the size of a blockchain grows quickly, making it challenging to save, disseminate, check, and mine new blocks to chain. In conventional blockchains like Bitcoin, a block is added to the blockchain about every 10 minutes. Among the candidate blocks proposed, the block that includes a complete transaction list and provides the best solution to the cryptography hash computation (which is a process called Proof of Work (PoW)), is the block that is selected to be added (i.e., chained) to the blockchain. The miner that made the block receives a compensation for its block contribution. This is why numerous miners compete in making the next block to add to the blockchain. Miners periodically generate blocks that are sent throughout the network to have other validation nodes verify and compare their block with other blocks by checking the block headers and list of transactions in the block. If more than half of the validator nodes determine that a block has a valid transaction list and has the minimum hash value in the block header, then that block is connected to the blockchain and the validators maintain a common distributed ledger. As a result, every time a block is generated, all nodes examine each candidate

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

block, so the blockchain faces scalability problems as the number of nodes increases due to an increasing amount of message overhead. To address blockchain scalability issues, the application of sharding technology has been proposed [1]. Blockchain sharding originated from existing database sharding, in which a database is divided into multiple shards where independent transactions are processed and validated in parallel. In other words, transactions are selectively processed by a pre-selected group (shard), which allows transactions to be processed simultaneously on different shards. Blockchain sharding will result in an increase in the transactions per second (TPS) performance, in reference to the number of shards applied to the blockchain. However, since the number of nodes participating in the validation process of a sharded block decreases as the number of shards increases, it becomes easier for malicious nodes to take over the consensus initiative of a single shard, called 'single-shard takeover' [2]. As a result, even though the ratio of malicious users in the network may only be a minority, a single shard can experience an increased vulnerability to 51% attacks, making shard-based blockchains more insecure. In existing sharding based blockchain cryptocurrencies (e.g., Ethereum 2.0, Zilliqa, and ELASTICO), a random shard distribution is employed to maintain the fairness of the shard distribution [1]–[3]. However, this method does not impose any penalty on malicious behavior, and also the scheme assigns shards based on simple randomness, which is not sufficiently reliable to be used in block consensus procedures defending against various malicious attacks. In addition, there are very few defense mechanisms against malicious behavior that block validators can perform during the consensus process. This is why in this paper, a trust-based shard distribution (TBSD) scheme that uses a genetic algorithm (GA) approach to minimize the probability that the consensus group of any single shard is formed by a majority of malicious nodes is presented. The proposed TBSD scheme accomplishes this by distributing consensus nodes fairly according to their trust value, where the simulation results show that the proposed TBSD algorithm is more robust than existing techniques. In the TBSD scheme, nodal trust values are computed based on their previous consensus vote and the consensus result. Through a fair trust distribution, malicious nodes are assigned to different shards, so that the bias in trust level of all sharding group nodes is minimized. The corresponding shard distribution problem is NP-hard (which is proven in Lemma 1 of this paper), so there is no exact algorithm that is guaranteed to find the optimal solution within polynomial time. However, our problem can be solved with a metaheuristic approach that can provide a sufficiently good solution with incomplete computation capacity. GA[1] is one of the representative metaheuristic evolutionary algorithms that have been used in optimization problems of network distribution [4]. The optimal shard distribution is computed based on a modified GA that can quickly find a (near) optimal solution by exploring a variety of solution spaces. For each generation, the population of candidate solutions in the GA become closer to the optimal fitness function through the crossover and mutation process. In the TBSD scheme, the GA assigns an array of shard numbers to each node, which becomes its chromosome. Chromosomes are reproduced in each generation, and a selective population of candidate chromosomes survive, which finally form an optimal chromosome set representing the fair shard distribution set to be used by the sharded blockchain. The proposed TBSD scheme has the following unique features.

1) The TBSD trust model is designed based on previous consensus results. The trust model evaluates nodes by penalizing malicious consensus behavior and trust tampering behavior through comparison with other nodes. This provides a quantitative measurement of which node has been trustable in agreements.
2) To prevent malicious shard formation, a GA is used to achieve maximum fairness of shard trust. The GA finds the best shard distribution set to form the trust level of each shard equally. The GA scheme distributes nodes with similar consensus opinions in the previous round to different shards, preventing coordinated collusive malicious behavior, and thereby enhancing the fairness of the shard distribution.

II. PRELIMINARIES

In this section, the related techniques used in the proposed scheme are described.

A. TRUST MANAGEMENT SYSTEM

Trust management is a scheme that can enhance the security level by determining the reliability of an entity (based on its behavior pattern) in a network [5], or help improve networking functions, such as secure routing [6] and clustering [7]. The basic trust evaluation is based on peer-to-peer evaluations among nodes, where a honest node should behave in a way that is consistent with the majority of honest nodes in the network. This enables malicious behavior in the network to be detected, based on the belief that a majority of the network nodes are honest. All nodes compute the trust value of other nodes based on peer-to-peer evaluation. Each node forms a local trust opinion (LTO) of all neighboring nodes, where the LTO values are aggregated to form the subjective reputation (SR), behavior reputation (BR), and credit reputation (CR). The trust agent (TA) of the network combines this information to compute the global reputation (GR) of each node [8]. The type of information used to form the LTO, SR, BR, and CR can be modified to fit its purpose. In trust systems there is a motivation to discourage malicious behavior, which is based on the continuous monitoring of behavior patterns. Trust systems help in protecting the network because even if an entity passes the primary security procedures (e.g., authentication and authorization), covert malicious behavior can be

[1] A GA was applied because it is fast and robust, although it may result in a suboptimal solution. Other evolutionary algorithms or machine learning schemes may be used instead.

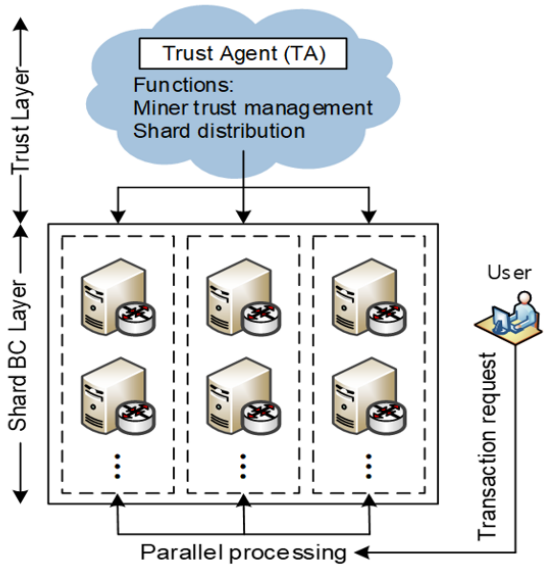

FIGURE 1. Overview of the TBSD scheme structure.

detected. In blockchains, the process of creating and linking blocks is based on the consensus of the network validation nodes (e.g., miners). When a block is proposed, if a majority of the nodes in the network reject the block, that block cannot be connected to the chain. The validator nodes must evaluate the proposed block and follow a consensus rule that follows the dominant opinion by referring to the evaluation of other nodes. The TA[2] of the trust management system supporting the network will support the proposed TBSD process for the sharded blockchain network. Fig. 1 shows the conceptual view of the proposed TBSD scheme. In the Sharded blockchain (BC) Layer, each shard group processes independent transactions received from its users. The Trust Layer is the support layer of the Sharded BC layer, which monitors the consensus behavior of the miners to manage and calculate the trust value of the miner nodes. Once the trust value (e.g., GR) of the miners are computed, the TA uses the GA to derive the optimal shard clustering sets that will maintain a fair shard distribution.

B. CONSENSUS

In blockchains, the consensus procedure determines how to select a new transaction ledger block to be added to the blockchain, and the commit step confirms to its users which block was added. In distributed systems like blockchains, the consensus scheme is desired to have both 'safety' and 'liveness' properties. 'Safety' refers to the property that the consensus result of the network should be consistent for any node, where 'liveness' refers to the property that any node participating in a consensus should participate in reaching a consensus. Extending from the Fischer, Lynch, and Patterson (FLP) impossibility concept [9], it is challenging for a distributed consensus algorithm to fully satisfy both safety and liveness at the same time. For example, the Bitcoin blockchain uses the Nakamoto consensus scheme (based on minimum hash value generation for the new transaction ledger block), where the block to be added is selected based on PoW through majority voting and is confirmed through the longest chain first rule [10]. When conflicts in the consensus occur, a 'fork' is temporarily formed until the conflict is resolved. As can be seen, the Nakamoto consensus algorithm used in Bitcoin is designed to pursue liveness by partially sacrificing the safety property. On the other hand, Byzantine Fault Tolerance (BFT) based consensus algorithms (e.g., the Practical BFT (PBFT)) ensure safety while partially sacrificing liveness, even if the consensus fails [11]. PBFT uses a 5-step protocol to guarantee consistent consensus results. However, PBFT's 5-step message exchange for agreements can cause scalability problems when the number of nodes increases [12]. Another consensus scheme that was made to be more efficient than PoW is Proof of Stake (PoS). PoS is based on a leader selection protocol that increases the probability of being selected as a leader in proportion to the amount of a miner's stake holdings. In PoW, miners consume a significant amount of energy in competing to compute the minimum hash value of the new transactional ledger block, in hope to be the block to be added to the blockchain to receive an incentive reward. Opposing to this, PoS was designed to require much less computational energy than PoW. A conventional PoS algorithm is Follow-the-Satoshi (FTS) algorithm [13]. Currently, many blockchain technologies use a hybrid consensus algorithm combining PoS and BFT based protocols [14]–[16].

C. SHARDING

Blockchain sharding techniques can be classified into transaction sharding and state sharding. Transaction sharding has the advantage that transactions can be processed in parallel by a disjoint shard, resulting in an increased throughput performance, where the increment is almost linear to the amount of sharding applied. State sharding stores the disjoint part or state of the transaction by ledger pruning, therefore it can save the required storage size [17]. Both types of sharding commonly require nodes to be assigned to different shards for every particular epoch. The existing sharding protocol for shard allocation of each node uses a PoW based random scheme. In ELASTICO, each node competes in PoW hashing to get its identity, then all nodes are randomly assigned to different shards [1]. In addition, Rapidchain uses a similar shard configuration scheme, where participant nodes can be randomly assigned by using the randomness of hash values in the peer discovery process [18]. But because the distributed shard grouping process uses the PoW approach for identity verification, these schemes do have hash computation burdens. On the other hand, the SSChain blockchain supports both transaction sharding and state sharding based on its non-reshuffling structure, and uses the existing PoW algorithm for shard consensus [17]. In addition, SSChain manages two

[2]The proposed TBSD scheme can be modified to have multiple TAs support the sharded blockchain.

chain structures (i.e., root chain and side chain), which can easily compromise the overall security of the root chain.

The procedures of the block consensus in sharding protocols can be classified into the following steps [1], [19]. First, each node finds their peer to be clustered for a shard based on a random PoW based algorithm. After all nodes are assigned to a specific shard, each shard runs an intra-shard consensus to process transactions independently. Transactions are assigned to each shard according to their input address. After the intra-shard consensus is completed, the final consensus to create the block to be added to the blockchain is made by the final committee. The connected block contains the SHA 256 transaction hash processed in all shards. Vulnerability exists in the fact that if a single shard takeover occurs in the intra-shard stage, a malicious validator can cause the consensus to fail or result in incorrect transaction processing.

D. USER/MINER AUTHENTICATION

In blockchain systems such as Bitcoin and Ethereum, all users can create their own account address using a public key (PK) and private key (i.e., the secured key (SK)) based on asymmetric key technology. Transaction data encrypted by a SK can be decrypted by the pairing PK. Therefore, if a user's SK is not stolen, the identity of the transaction can be clearly proved. Users can create their PK and SK pair using a PK generator. Problems can arise when a miner creates many accounts, and authenticates its multiple identifications using the PK based approach. Since, the votes of miners are used to verify blocks, a miner with multiple accounts can maliciously influence the voting process, which is what happens in 51% attacks on blockchains [20]. As blockchains become too long and challenging to send, receive, and store, applying sharding technology to blockchains is almost unavoidable. However, considering the possibility of mining pools colluding, an additional layer of authentication on sharded blockchains is needed.

E. GENETIC ALGORITHM

When each trust value is calculated by the trust management system, the GA aims to find the best shard distribution that makes each shard trust level equivalent. However, GAs are metaheuristic schemes that can help find a near-optimal solution. Strictly optimal solutions are difficult to find because computing shard distribution sets (to have similar trust levels) is a NP-hard problem, which is proved later in Lemma 1. This is why a feasible suboptimal scheme like GA is suitable for blockchains that operate block adding procedures based on a strict periodical schedule.

III. SYSTEM MODEL

The proposed TBSD trust model is based on the block consensus results from each miner, which is used by the modified GA algorithm that distributes the shard fairly using the calculated miner trust value. The notation of the parameters used in the TBSD scheme are described in Table. 1.

TABLE 1. Notations in TBSD scheme.

Notations	Description
s_i	Stake of the ith node
S_i	Subjective consensus opinion of the ith node
p_i	Leader selection probability of node i
$L_{i,j}$	Local consensus result of node j from node i
ℓ_j	Row vector of local consensus result of node i
$Sim_{i,j}$	Cosine similarity of l_i and l_j
W_i	Aggregated similarity of node i
t_i	Final trust value of node i
G_i	Set of the ith shard group
T_{G_i}	Aggregated trust of the ith shard group
K	Number of total shards
P_C	Crossover probability
P_M	Mutation probability
G	Shard distribution set

A. TRUST MODEL ON BLOCKCHAIN MINER

Node (miner) trust is computed based on the following steps: (1) Leader selection, (2) Block commit phase, (3) Reporting subjective consensus opinion (SCO), (4) Local consensus result (LCR) formation, and (5) Final node trust evaluation.

1) LEADER SELECTION

In every consensus round, a block is proposed by the elected leader based on PoS. If the FTS PoS algorithm is used, the leader selection probability of miner node i among n participants is

$$p_i = \frac{s_i}{\sum_{j=1}^{n} s_j} \tag{1}$$

where s_i is the stake of node i. Without loss of generality, it is assumed that the stake and computational capabilities of all mining nodes are equivalent, and therefore, the probability of leader selection is $\frac{1}{n}$. After the leader is selected, the leader broadcasts its own block composed of the transaction list to the network.

2) BLOCK COMMIT STEP

The block proposer broadcasts the block to the network, and other nodes participate in the commit process to check the validity of the proposed block. The validation decision of the nodes participating can either be 'valid block' or 'invalid block.' Invalid transaction list, faulty hash value, or invalid key are proper reasons that lead to an 'invalid block' decision. However, malicious nodes and colluding attack nodes would claim 'invalid block' without having a proper reason. After each node makes a decision on the validness of the new block, all nodes broadcast their block commit results to the other nodes. If a proposed block has been verified by a majority of other nodes, the block is accepted and linked to the blockchain.

3) REPORTING SCO

SCO is a trust table generated by the validator node for each round of consensus. After the block commit phase, nodes

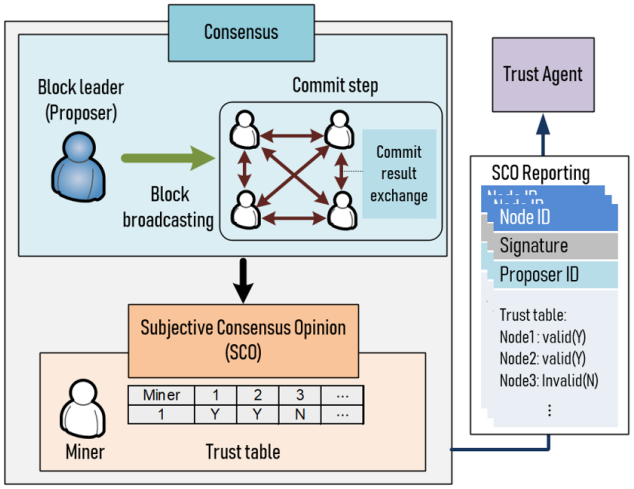

FIGURE 2. Overview of SCO reporting.

verify the commit result of the proposed block. At this time, a node must refer to the block evaluation of other nodes when chaining the proposed block to its own blockchain. In addition, a node can collect tamper-proof validation results of other nodes based on PK authentication. Then all nodes verify the proposed block of the proposer and broadcast the validation result to the entire network. Assuming there are n validator nodes in the network, each node saves the other n validation results into a $(1 \times n)$ vector form. If the block is judged to be valid, it is marked as 'Y' (i.e., Yes, accepted block), otherwise, it is marked as 'N' (i.e., No, rejected block). Each SCO acts as a trust table that nodes evaluate validation behavior of the other nodes subjectively. The SCO vector recorded by node i is marked as S_i and is reported to the TA. A conceptual view of SCO reporting is described in Fig. 2.

4) LCR FORMATION

LCR is computed by collecting the SCO from each node. First, the ratio of 'Y' and 'N' recorded in each SCO is converted into a number. If $n_i(Y)$ and $n_i(N)$ represents the number of Ys and Ns in the SCO vector of the ith node (S_i), the ratio of Ys and Ns among the total nodes can be represented as $\frac{n_i(Y)}{n_i(Y)+n_i(N)}, \frac{n_i(N)}{n_i(Y)+n_i(N)}$, respectively. Then all values of 'Y' and 'N' in the existing SCO vector are converted to corresponding ratio values. For example, if there are three nodes, where nodes 1 and 2 approve a block while the malicious behaving node 3 rejects the block, in this case, nodes 1 and 2's SCO value will be $(\frac{2}{3}, \frac{2}{3}, \frac{1}{3})$. Then the LCR values (i.e., $L_{i,j}$) are computed using the collected converted SCO values of all nodes, forming a $N \times N$ matrix, where N is the number of total nodes, and $L_{i,j}$ represents the entire network's opinion of the jth node based on the perspective of the ith node. Then, $L_{i,j}$ can be expressed as in (2).

$$L_{i,j} = \begin{cases} \dfrac{n_i(Y)}{n_i(Y)+n_i(N)}, & \textit{if committed result of } j \textit{ is '}Y' \\ \dfrac{n_i(N)}{n_i(Y)+n_i(N)}, & \textit{if committed result of } j \textit{ is '}N' \end{cases} \tag{2}$$

The LCR is a quantitative value indicating the opinion of a node based on the evaluation of other nodes in the network.

5) FINAL NODE TRUST EVALUATION

The final node trust evaluation is described in Fig. 3. To obtain the final node trust value, the average trust value and trust weights are computed. The final nodal trust of node i (t_i) is expressed as the product of the average trust and trust weight. The average trust of node j (u_j) is represented as

$$\sum_{i=1}^{n} \frac{L_{i,j}}{n} = u_j \tag{3}$$

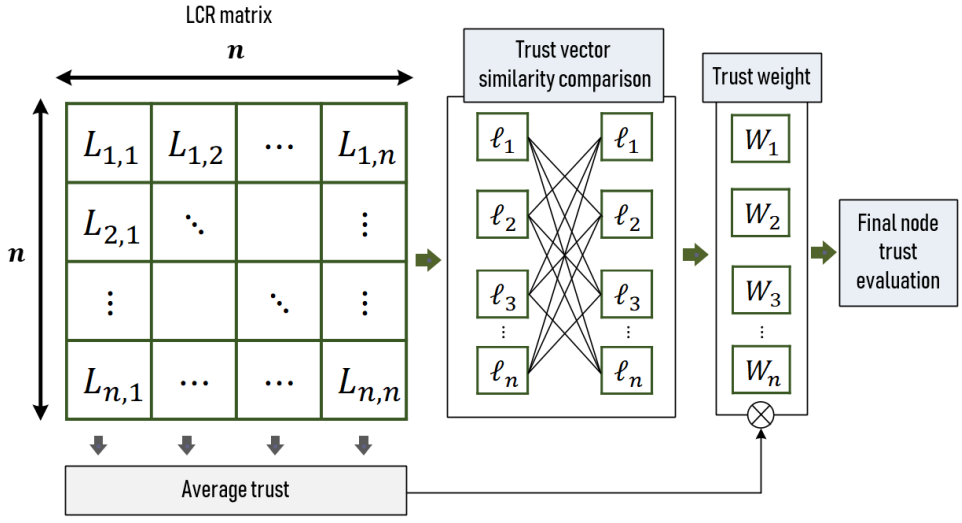

FIGURE 3. Node trust evaluation in TA.

where u_j is the average of the column vectors of the LCR matrix. The value u_j is an indicator of how the commit result of the jth node matches the opinion of the entire network.

The trust weight is used to impose a penalty on a node that presents a conflicting trust report. Each LCR row vector of node i is denoted as ℓ_i (trust vector in Fig. 3). If any two nodes have submitted the same SCO, each LCR row vector would be completely identical. However, if there are malicious nodes modifying the commit result of the other nodes, the inconsistency in the LCR matrix increases. To measure the level of inconsistency, cosine similarity is used. The cosine similarity of two LCR vectors of node i and j ($Sim_{i,j}$) is expressed as

$$Sim_{i,j} = cos(\ell_i, \ell_j) \tag{4}$$

where $cos(\ell_i, \ell_j)$ is bounded to [0, 1] if all components of the vector are positive. The trust weight of node i is expressed based on the aggregated similarity $W_i = \sum_{j=1}^{n} \frac{Sim_{i,j}}{n}$. The trust weight ($W_i$) is an average of all similarities of the trust vectors with other nodes, which measures how much ℓ_i is similar to the average of all other LCR row vectors. The ith node trust (t_i) is computed by multiplying the average trust and the trust weight, which is represented as

$$t_i = W_i u_i \tag{5}$$

where, if node i has an outlier opinion ℓ_i (trust vector of i), W_i will be close to zero. On the other hand, if ℓ_i is similar with the overall trust vectors of other nodes, W_i will be close to one.

The final node trust value is also bounded to [0, 1]. If t_i is closer to 1, both the behavior of the trust reporting and the commit results of node i become the majority consensus behavior in the corresponding consensus round. If there is an unanimous consensus without any malicious node detected, the trust value of all nodes will be 1. However, if there is a malicious node reporting an outlier trust vector, the trust weight of the malicious node will be drastically reduced by the majority opinion of the honest nodes. The final node trust value is used to make shard distributions fair, which is described in the following sections.

B. ADVERSARY MODEL

A malicious node in consensus is defined by its commit phase behavior. An honest node makes 'valid' decisions for properly prepared blocks and makes 'invalid' decisions for faulty blocks. In addition, a honest node always generates a honest valid block when it is selected as a leader. However, malicious nodes do not follow the honest consensus rules. In this paper, the objective of malicious nodes is to induce failure of consensus to proper blocks, and to commit a faulty block if they can form a malicious majority in a single shard. If some behavior does not negatively affect the consensus of the blockchain, it cannot be considered as a successful malicious attack. The specific classification of a malicious behavior is modeled referring to existing network attack scenarios as follows [8].

1) NAIVE MALICIOUS ATTACK (NMA)

A malicious node provides improper services, not complying with the given network protocol. If a NMA node is selected as the leader, it will always make a faulty block and draw commitments to the faulty block. Otherwise, these nodes will deny the block from a honest leader node. However, it does not modulate the SCOs. The purpose of NMA is to hamper honest consensus behavior. The attacker continues the malicious behavior consistently.

2) COLLUSIVE RUMOR ATTACK (CRA)

In addition to providing improper commit results same as in NMA, the malicious nodes report opposite SCOs generated by honest nodes (i.e., good/bad mouthing attack). CRA nodes will accept a false block from a malicious node and reject a honest block from a honest node. If a CRA node is selected as the leader, it will always make malicious decisions to commit to a faulty block. The purpose of this attack is to disrupt accurate consensus and trust evaluations.

3) CONFLICTING BEHAVIOR ATTACK (CBA)

In this attack, malicious nodes behave inconsistently. If a CBA node is selected to be the leader, it will propose proper and faulty blocks with a 50% probability. In addition, it will also falsely commit and report SCO vectors with reversed faulty data for half of the honest nodes. The purpose of CBA is to cause an inconsistent consensus behavior and faulty SCOs, which critically interferes with the accuracy of the trust evaluation system.

NMA and CRA nodes will consistently generate malicious blocks when they are elected as a block leader. Invalid blocks created by malicious nodes in a single shard are rejected if the majority of that shard is honest. If malicious nodes occupy a majority of the shard, a corrupted block can be committed and connected to the chain. On the other hand, CBA nodes increase discrepancy between the SCOs of nodes through inconsistent consensus behavior and induce disturbance in the trust computations.

C. SHARD DISTRIBUTION

In the following, a 'corrupted shard' refers to a shard whose block consensus is wrongfully decided by the influence of its malicious nodes, and a 'uncorrupted shard' refers to a shard whose block consensus was correctly decided by the influence of its honest nodes. Shard distribution is performed based on node trust, but initially it will randomly distribute nodes to shards at the beginning, due to the absence of any trust information. After the initial random sharding is finished, the TA collects all SCO information from each shard and composes the corresponding LCRs. Then, node trust values are evaluated through the composed LCRs, and finally each node is assigned to each shard using the GA trust-based shard distribution scheme.

1) PROBLEM STATEMENT

The proposed TBSD scheme's objective in assigning nodes to shards is to find a fair shard distribution set to prevent colluding based on the trust value of the nodes. When trust is calculated for each shard in the trust model, the node with the majority opinion on that node gets a high trust value, and vice versa. Assuming that the trust model actually reflects the reliability of the node, the aggregated trust of each shard should be kept as close as possible. If discrepancy of the aggregated trust in a shard is large, that shard can be heavily biased, and the threat of malicious colluding may increase.

In addition, malicious nodes may form a majority of a single shard in the initial random shard grouping. In this case, malicious nodes may receive a high trust value due to being the majority. These malicious node will have high similarity between their trust vectors, therefore they will eventually receive the same trust values. To prevent further collusion, in the proposed scheme, nodes that receive the same trust evaluation are separated and assigned to different shards in the following epoch (round).

Therefore, to properly isolate malicious nodes from clustering, the following two conditions should be required: 1) Find the shard distribution sets that result in the same aggregated trust value for each shard. 2) Nodes with the same consensus behavior should be separated into different shards in the next epoch. These two properties are applied to the proposed modified GA-based TBSD scheme.

To distribute nodes to each shard fairly, the objective function of the proposed scheme is to find the optimal shard distribution set while minimizing the root mean square (RMS) error (E) of the shard trust values,

$$\min E = \sum_{i=1}^{K}\sum_{j=1}^{K}\sqrt{\frac{(T_{G_i}-T_{G_j})^2}{K}}, \quad 1 \le i,\ j \le K \tag{6}$$

$$\text{Subject to } T_{G_i} = \sum_{j\in G_i} t_j \tag{7}$$

where K denotes the given number of shards in the blockchain system, G_i and T_{G_i} represent the ith shard set and the aggregated trust of the corresponding shard, respectively. In (7), t_j denotes trust value of node j in set G_i, where G_i stores the index of the node to be assigned in the ith shard group.

Finding the optimal shard set $(G_1, G_2, \cdots, G_K)$ given the trust set $T = (t_1, t_2, \cdots, t_N)$ to minimize (6) is a NP-hard problem, which is validated through *Lemma 1*.

Lemma 1: The TBSD problem is NP-hard.

Proof: In the TBSD problem (denoted as **D**), it is shown that there is a shard distribution set $G = G_1 \cup G_2 \cup \cdots \cup G_k$ of G into K disjoint non-empty shard sets satisfying the objective function $\min_{i,j\in K} \sum_{i=1}^{K}\sum_{j=1}^{K}\sqrt{\frac{(T_{G_i}-T_{G_j})^2}{K}}$, while the trust set T is given. Then, the decision problem of TBSD is changed to find the shard distribution set G satisfying $E = 0$ because it is evident that the condition $T_{G_i} = T_{G_j}$ for $\forall\,(i, j)$ is needed to minimize the objective function E to zero. In this case, the decision problem of TBSD is reduced to a K-partition problem finding the set partition with an equal trust sum of each shard, which is a NP-complete problem [21]. Therefore, the decision problem version of TBSD is NP-complete. On the other hand, the TBSD problem **D** is an optimization problem that finds an optimal set G that minimizes E. For a given arbitrary solution set G^*, it cannot be guaranteed that there exists a polynomial time algorithm that can verify that the given G^* results in a minimum E. Therefore, since the problem $\mathbf{D} \notin$ NP, it is NP hard not in NP-complete (NPC). ■

Next, when the shard distribution is processed randomly, the advent probability of a malicious shard is analyzed in the following *Lemma 2*.

Lemma 2: The generalized equation for probability of malicious shard clustering is given as $P_{mal} = 1 - \prod_{i=1}^{K}\sum_{l=\lceil N_i b\rceil}^{N_i}\binom{N_i}{l}p^{N_i-l}(1-p)^l$.

Proof: Let N be the total number of nodes satisfying $N = \sum_{i=1}^{K} N_i$, where N_i represents the number of nodes in the ith shard. Let b be the consensus bound, which represents the minimum quorum required for agreement in the consensus algorithm (which is $\frac{1}{2}$ for majority voting and $\frac{2}{3}$ for PBFT), and p is the malicious node ratio of the network.

Then, the shard needs at least $(N_i b)$ nodes to start a safe consensus round and remain as an uncorrupted shard. Random variable X_j^i is 1 if the jth node of the ith shard is a honest node, and 0 elsewise. Let $X^i = \sum_{j=1}^{N_i} X_j^i$, where X^i is the total number of honest nodes in the ith shard. Then, X^i follows a binomial distribution with a malicious node ratio of p. The probability p_{honest}^i denotes the probability that shard i is an uncorrupted shard based on the following equation.

$$\begin{aligned} p_{honest}^i &= Pr[X^i \ge N_i b] = 1 - \sum_{l=0}^{\lceil N_i b\rceil} Pr[X^i = l] \\ &= \sum_{l=\lceil N_i b\rceil}^{N_i}\binom{N_i}{l}p^{N_i-l}(1-p)^l \end{aligned} \tag{8}$$

In addition, the probability of malicious shard clustering (p_{mal}) is a complementary event of all K shards being uncorrupted shards. Therefore, the generalized probability equation for malicious shard clustering is derived as follows.

$$p_{mal} = 1 - \prod_{i=1}^{K}\sum_{l=\lceil N_i b\rceil}^{N_i}\binom{N_i}{l}p^{N_i-l}(1-p)^l \tag{9}$$

■

2) GA BASED SHARD DISTRIBUTION PROTOCOL

The proposed TBSD scheme assigns nodes to shards based on their trust level. In order to prevent the composition of a shard with a majority of malicious nodes, the proposed system uses a GA to distribute nodes fairly based on their trust level. The GA fitness function is based on the optimization function (E), which is obtained by the RMS of the average trust of each shard and average trust of the entire node set. Until the fitness function is minimized, the GA process repeatedly attempts to

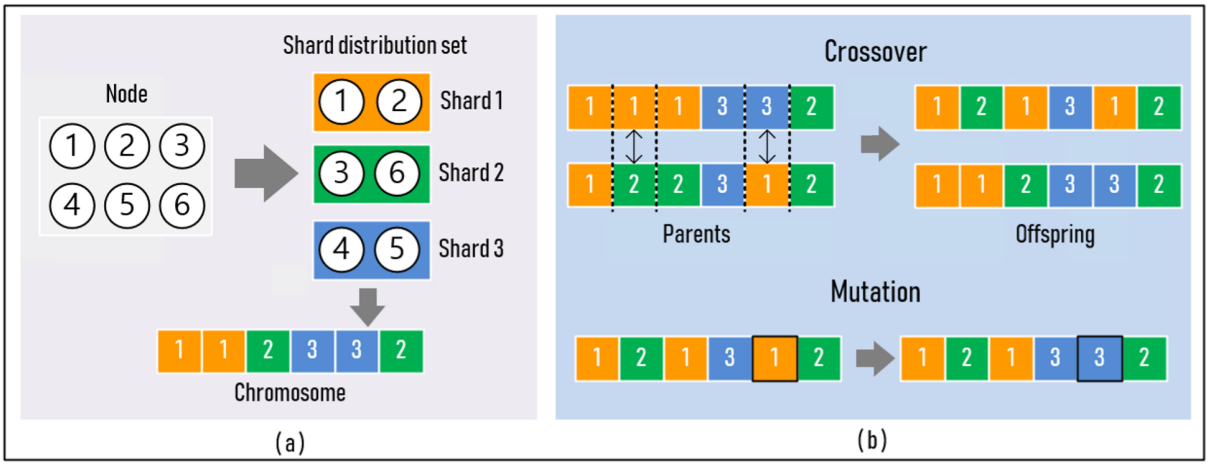

FIGURE 4. Shard distribution example: a) chromosome generation; b) crossover and mutation process.

form optimal shard distribution sets. Fig. 4a shows how the shard distribution set is converted to chromosomes, in which 6 nodes are distributed to 3 shards, where nodes 1 and 2 are assigned to shard 1, nodes 3 and 6 are assigned shard 2, and nodes 4 and 5 are assigned to shard 3. The shard number to which a node is assigned is indicated in the index of the chromosome, in which the first generation of the shard distribution set is presented. The GA forms multiple populations of the initial chromosome set (which represents the first generation) and repeats the crossover and mutation process iterations until the fitness function is minimized.

a: CROSSOVER PROCESS

Among the many populations initially created, parents are selected, and the process of generating offsprings through genetic exchange of the parents is executed in the crossover process. Through the crossover process, the features of the parents are exchanged and inherited to each offspring. The algorithm filters the dominant chromosomes suitable for the fitness function through the offspring generation process. In the TBSD scheme, uniform crossover was applied in the one-by-one chromosome switching process. In general, the GA process combines a number of parent pairs randomly without evolutionary direction. The detailed crossover process is shown in Fig. 4b. First, the initial parents' population are generated by a random shard distribution, and the two parents are selected by the crossover probability P_c. After the parents have been determined, the fitness function that represents the RMS error of the trust is computed. If the parent fitness function is too high, the gene with a higher trust value should be replaced with a gene of a lower trust value of the other parent to balance the trust level of the shard. In this example, the parents have shard distributions of [1, 1, 1, 3, 3, 2] and [1, 2, 2, 3, 1, 2]. If the trust of shard 1 in the upper parent is higher than the average, it can be exchanged with the genes of the second node of the parent below. Therefore, the crossover process changes genes to find the optimal shard distribution set that satisfies the fitness function. The offspring pairs generated by the crossover become parents in the next iteration. Through this process, chromosomes that do not fit the fitness function are made extinct, and only a few chromosomes near the optimal solution will survive.

b: MUTATION PROCESS

The mutation process is a way to pursue the diversity of solutions from an evolutionary point of view. Candidate solutions go through the crossover process to the optimal solution, but at the end of the phase, the variability of the solution decreases. Often, due to the lack of variety of solutions, the optimal solutions that satisfy the fitness function can easily fall into a local optimal point. The mutation process makes certain variants of the solution candidates more diverse by applying a mutation probability (P_m) to the chromosomes. An example of the mutation process is shown in Fig. 4b. In Fig. 4a, the shard group assigned to node 5 has changed from 1 to 3. In this case, until the mutation occurs, the number of nodes belonging to shard 1 is 3, which occupies half of the total nodes. The mutation process assigns nodes that were assigned to shard 1 to other shards randomly.

c: PARTITIONING NODES WITH SAME TRUST

In the TBSD scheme, modification in the GA were applied to prevent malicious colluding. As mentioned in the problem statement, trust computations may be unreliable in the early stages. This is because in the initial stage, nodes are assigned to shards randomly due to not having any calculated trust values of the nodes, and also it takes a few epochs to accumulate reliable trust values of the nodes. However, if a colluded attack is conducted in the initial stage, the colluded nodes will receive the same trust values, and therefore will be separated in the following stage. This method will quickly separate nodes that attempt to collude, preventing further colluding opportunities.

To execute this method, the trust values are used as follows. In order for malicious nodes to collude, the honest nodes should be isolated by modulated SCO opinions generated by the colluding malicious nodes. In addition, in order for a malicious node to get a malicious consensus result in the shard, the commit result of the block and the SCO report need to be matched, and therefore, the colluding nodes will receive the same trust evaluation. To prevent further collusion, the proposed scheme distributes nodes with the same trust value to different shards. This feature can be applied in the mutation process as follows. In Fig. 4b, the mutation process selects node 5 as a mutation node based on P_m. At this time, since the shard allocation of node 5 is 1, it can be changed to shard 2 or 3. If there is a node with the same trust value as node 5 in shard 2, it will be reallocated to shard 3 through a one-by-one exchange. In other words, nodes with the same trust value will be separated to different shards in the mutation process. The pseudocode for the TBSD scheme is described in **Algorithm 1**. After the trust computations are completed for all nodes, the GA shard distribution process begins. In the GA procedures, the crossover, mutation, and partition process are iteratively operated to find the optimal shard distribution set.

Algorithm 1 Optimal Shard Distribution Based on the TBSD Modified GA

Input T: trust set of nodes, where $T = (t_1, t_2, t_3 \cdots, t_N)$, K: number of shards, T_{G_i}: aggregated trust of the ith shard, N: number of total nodes, P_c: crossover probability, P_m: mutation probability, $C = (C_1, C_2, \cdots, C_N)$: shard distribution chromosomes, C_i: assigned shard number of the ith node, $1 \leq C_i \leq K$, P: set of parents, O: set of offsprings, M: number of populations, θ: shard distribution threshold, γ: crossover operator, ω: mutation operator

Initialize generate initial parent group
Arbitrarily generate M different chromosomes and store in parent set P
Iterative crossover and mutation process
while $\sum_{i=1}^{K}\sum_{j=1}^{K}\sqrt{\frac{(T_{G_i}-T_{G_j})^2}{K}} > \theta$ **do**
  Crossover process
  Generate pair for different M chromosomes in P
  Select random index $\lceil P_c N \rceil$ number of times to interchange selected gene of each pair
  Output $P \underset{\gamma}{\rightarrow} P^*$
  Mutation process
  for $\forall i$ in P_i^*, $1 \leq i \leq N$ **do**
    Select random index $\lceil P_m N \rceil$ number of times.
    Change the gene (shard number) of the selected index
  end for
  Partitioning nodes with same trust
  for the selected index i **do**
    if there is j with $t_i = t_j$ **then**
      C_i mutated to C_x where $C_x \neq C_j$ where $1 \leq C_x, C_j \leq K$
    else
      C_i mutated to arbitrary C_j
    end if
    Output $P^* \underset{\omega}{\rightarrow} O$
  end for
end while

IV. SECURITY ANALYSIS

A 51% attack on a single shard is called a 'single shard takeover,' which is a major security challenge in sharded blockchains [22]. Since transactions are independently processed for each shard, the occurrence of a corrupted shard is fatal to sharded blockchains. Therefore, a 51% attack on a blockchain can be devastating, since the effects can be extended to a variety of malicious behaviors (e.g., double spending attack, eclipse, and denial of service) [20]. Therefore, preventing a 51% attack on single shard is important in sharded blockchain security, because the distributed ledger of a blockchain relies on the majority consensus of the validator (miner) nodes. To effectively attack a single shard, malicious nodes need to have the same commit result to overturn the majority vote of the consensus process. Based on this point, the proposed trust model is designed to make the trust evaluation of malicious nodes to be similar if their commit behaviors are similar by applying a penalty weight W value. However, the trust value of a node is computed based on a local consensus of each shard, where malicious nodes in the initial shard may be a majority. In this case, malicious nodes can obtain high trust values if they collude and exclude honest trust reporting by intentionally adding discrepancy to their SCO tables. To solve this problem, the modified GA process includes additional partitioning of nodes with the same trust value to separate potential collusive nodes. If malicious nodes in a particular shard conspire a false consensus, they will attempt to manipulate the majority commit result to successfully launch a 51% attack. Their coordinated malicious behavior will result in them being isolated in the following round. In addition, even if malicious nodes have different trust values due to their inconsistent behavior patterns, because the honest nodes will be evenly separated to different shards, this will also help to effectively hinder malicious nodes from succeeding in their collusive attack. Due to this feature of the TBSD scheme, even if a single shard takeover is attempted in a particular round, it cannot be sustained in the following round. In this sense, the proposed TBSD scheme is resilient against both consistent and inconsistent behavior of malicious nodes.

V. PERFORMANCE ANALYSIS

In this section, the performance metric to analyze the proposed TBSD scheme is described. In Table 2, there are four possible outcomes (i.e., TN, TP, FN, FP) based on the consensus result. Based on this, we defined two performance parameters in the following.

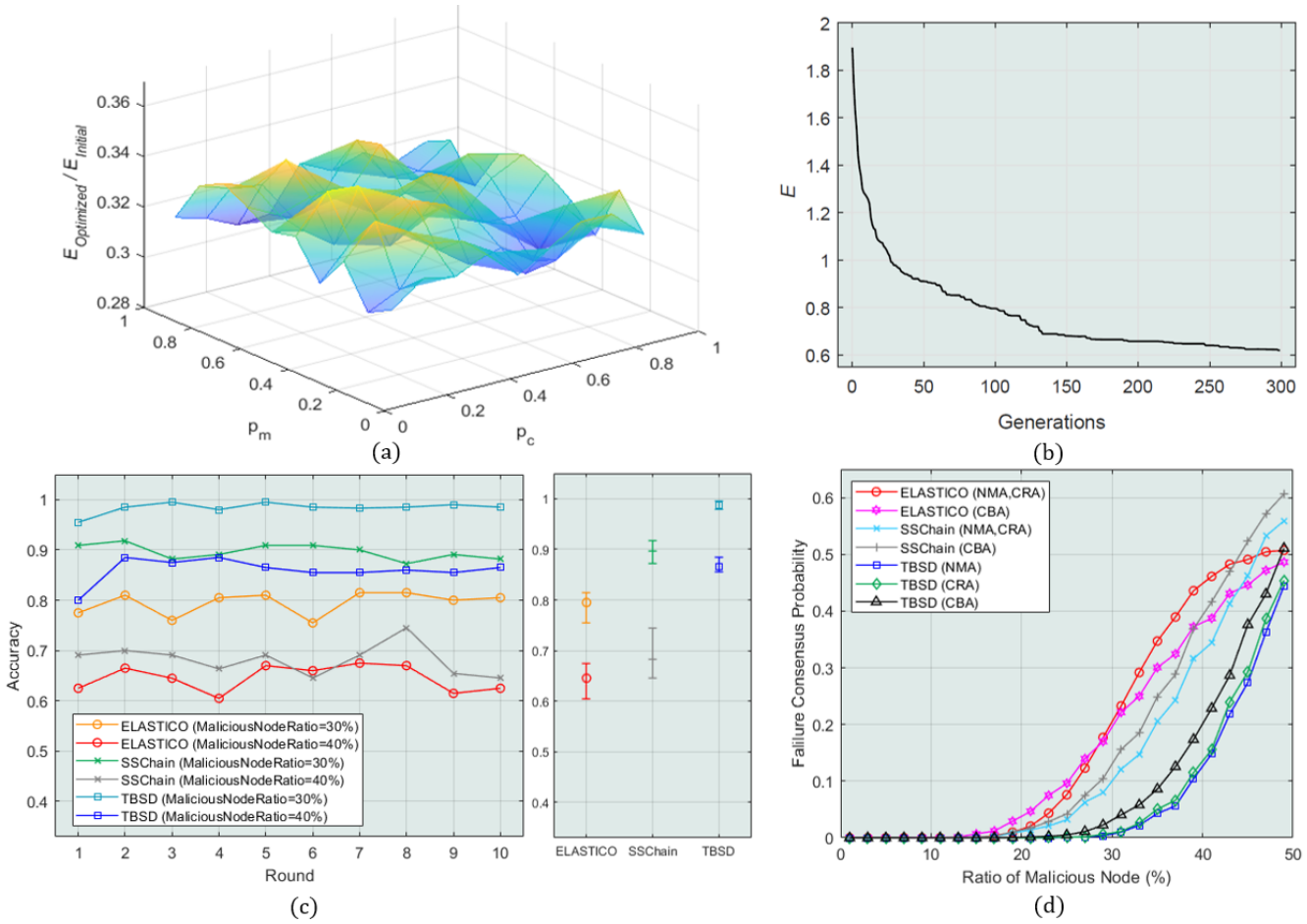

FIGURE 5. Simulation results, a) parameter selection; b) fairness evaluation; c) convergence test; d) false consensus probability.

TABLE 2. Performance metric on consensus.

		Consensus Result	
		Valid	**Invalid**
Ground truth of the miner	**Valid**	True Negative (TN)	False Positive (FP)
	Invalid	False Negative (FN)	True Positive (TP)

A. PERFORMANCE PARAMETERS

Accuracy (ACC): Accuracy is a measure of how well shards are distributed so that a corrupted shard does not occur. ACC is represented by the probability of an uncorrupted shard. An uncorrupted shard results in the determination that the block from a honest node as valid (TN) and judges the block from a malicious node as invalid (TP). Therefore, the accuracy is computed as $ACC = \frac{TN+TP}{TN+TP+FN+FP}$.

False Consensus Probability (FCP): FCP represents the probability of a corrupted shard occurring. A corrupted shard results in the determination that the block proposed by a honest node as invalid (FP), and evaluates the block proposed by a malicious node as valid (FN). As a result, the FCP is computed as $FCP = \frac{FN+FP}{TN+TP+FN+FP}$.

B. SIMULATION RESULTS

The shard distribution scenario for the blockchain was implemented using Python based on a total of 400 block validation nodes, which were allocated to 10 shards. The generations of TBSD were set to 300 per each shard distribution. In Fig. 5a, the change of the E value was observed based on different values of P_c and P_m for the range of 0.1∼1, with an internal size of 0.1. All experiment results were taken from the average values of 100 independent simulations. The vertical axis refers to the ratio of E after optimizing the initial E. As shown in Fig. 5a, the ratio of E values varies slightly depending on different P_c and P_m values, and converges within the range of 0.3∼0.33. As P_c and P_m increase, several local minimum points were observed, but the global minimum was obtained at $P_c = 0.8$ and $P_m = 0.4$. Thus, these points were selected by the GA to use in the following procedures. The shard distribution cycle takes place every 10 commit phases, where each cycle is denoted as a round (epoch) in the simulation figure.

Fig. 5b shows the change in E, where through the generations the GA finds the optimal chromosome shard distribution that results in a minimum trust average difference among the shards, which reduces the value of E. As shown in Fig. 5b,

the slope of the declining curve stagnates after 300 generations, which is why the 300th generation was set as a stop point.

Fig. 5c shows the change in accuracy over 10 rounds of the consensus process under NMA. In TBSD, the trust-based shard distribution starts after the first-round ends. For performance comparison, the shard-based blockchain ELASTICO of [1] and SSChain [17] were compared with the proposed TBSD scheme. Because TBSD finds the optimal shard distribution configuration using a GA, the consensus accuracy of each shard shows a more stable convergence characteristic compared to ELASTICO and SSChain. In the case of SSChain, the overall accuracy is higher than that of ELASTICO, but the accuracy deviation range is also the highest. This is because SSChain uses a PoW consensus based on a majority rule, so the average accuracy is higher than that of ELASTICO using PBFT with relatively tight consensus bounds. SSChain, on the other hand, has no reshuffling method and is very vulnerable when the nodes in charge of the root chain are compromised by having over 51% of malicious nodes in a shard. Therefore, SSChain shows the largest performance deviation in accuracy.

Fig. 5d shows the FCP under NMA, CRA, and CBA attack scenarios comparing both ELASTICO, SSChain, and TBSD. The NMA and CRA nodes disturb the consensus process, while the CBA nodes conduct inconsistent consensus behavior to hinder accurate trust evaluations. Because ELASTICO does not apply any penalty for malicious behavior, it cannot effectively prevent malicious node grouping to a shard. Since NMA and CRA differ in SCO tampering, the two attack scenarios are equivalent in ELASTICO, where no trust management is used. Under CBA, due to random consensus behavior, there is a difference in FCP compared to NMA and CRA. SSChain shows a moderate performance at less than 40%, but as the malicious node ratio increases, SSChain becomes more vulnerable because it manages not only the side chain of each shard, but also the root chain.

In the case of the proposed TBSD scheme, because nodes are assigned to shards corresponding to their trust evaluations to form a fair distribution, the probability of having a majority of malicious nodes in a particular shard is significantly reduced, compared when the ELASTICO algorithm is used. If malicious nodes had been assigned to the same shard in the previous round, the malicious nodes will all receive bad trust evaluations for their collusive behavior, in which the modified GA process will divide malicious nodes to different shards in the next round, which helps to prevent further collusive actions.

However, CBA nodes will not always behave the same because they intentionally apply inconsistent consensus behavior patterns. However, even in this case, because the modified GA attempts to evenly distribute honest nodes to different shards, the defense against CBA nodes is also very effective. As a result, the TBSD scheme effectively makes it very difficult for malicious nodes to collude in any shard. This effect results in the performance gain presented in Fig. 5c and 5d.

In addition, in terms of accuracy performance under both NMA and CRA attacks, TBSD achieves an average performance gain of 24% and 18% respectively over ELASTICO and SSChain at the 30% malicious node rate, and also achieves an average performance gain of 34% and 31% respectively over ELASTICO and SSChain at the 40% malicious node rate. In addition, under the CBA scenario, the accuracy performance gain reaches a maximum of 29% at the 40% malicious node ratio.

VI. CONCLUSION

Sharding technologies are being considered to solve blockchain scalability problems. However, sharding can make blockchains more vulnerable, as the fault tolerance level of a single shard will decrease, making shards more vulnerable to malicious attacks. To solve this problem, the proposed TBSD scheme distributes nodes to shards based on trust evaluations to avoid malicious node grouping. The fairness of the shard distribution is maintained by minimizing the discrepancy between the aggregated trust of each shard using GA processing. The ACC and FCP performance of the proposed TBSD scheme is compared with the ELASTICO and SSChain scheme in reference to NMA, CRA, and CBA attacks for different malicious node ratios, where the results show that the TBSD scheme provides a performance improvement for the adversary models tested.

REFERENCES

[1] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2016, pp. 17–30.

[2] J. Ray. (2019). *Sharding Introduction R&D Compendium*. [Online]. Available: https://github.com/ethereum/wiki/wiki/Sharding-introduction-R&Dcompendium#information

[3] (Aug. 2017). *The ZILLIQA Technical Whitepaper*. [Online]. Available: https://docs.zilliqa.com/whitepaper.pdf

[4] D.-L. Duan, X.-D. Ling, X.-Y. Wu, and B. Zhong, "Reconfiguration of distribution network for loss reduction and reliability improvement based on an enhanced genetic algorithm," *Int. J. Elect. Power Energy Syst.*, vol. 64, pp. 88–95, Jan. 2015.

[5] Z. Movahedi, Z. Hosseini, F. Bayan, and G. Pujolle, "Trust-distortion resistant trust management frameworks on mobile ad hoc networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1287–1309, 2nd Quart., 2016.

[6] J. Yun, S. Seo, and J.-M. Chung, "Centralized trust-based secure routing in wireless networks," *IEEE Wireless Commun. Lett*, vol. 7, no. 6, pp. 1066–1069, Dec. 2018.

[7] S. Seo, J.-W. Kim, J.-D. Kim, and J.-M. Chung, "Reconfiguration time and complexity minimized trust-based clustering scheme for MANETs," *EURASIP J. Wireless Commun. Netw.*, vol. 1, pp. 155–161, Dec. 2017.

[8] X. Chen, J.-H. Cho, and S. Zhu, "GlobalTrust: An attack-resilient reputation system for tactical networks," in *Proc. IEEE SECON*, Singapore, Jun./Jul. 2014, pp. 275–283.

[9] M. J. Fischer, N. A. Lynch, and M. S. Paterson, "Impossibility of distributed consensus with one faulty process," *J. ACM*, vol. 32, no. 2, pp. 374–382, Apr. 1985.

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

[11] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *Proc. 3nd Symp. Oper. Syst. Design Implement.*, Feb. 1999, pp. 173–186.

[12] M. Vukolić, "The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication," in *Proc. Int. Workshop Open Problems Netw. Secur.* Cham, Switzerland: Springer, 2015, pp. 112–125.
[13] C. T. Nguyen, D. T. Hoang, D. N. Nguyen, D. Niyato, H. T. Nguyen, and E. Dutkiewicz, "Proof-of-stake consensus mechanisms for future blockchain networks: Fundamentals, applications and opportunities," *IEEE Access*, vol. 7, pp. 85727–85745, 2019.
[14] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Proc. 37th Annu. Int. Cryptol. Conf. (CRYPTO)*, Santa Barbara, CA, USA, Aug. 2017, pp. 357–388.
[15] V. Buterin and V. Griffith, "Casper the friendly finality gadget," Oct. 2017, *arXiv:1710.09437*. [Online]. Available: https://arxiv.org/abs/1710.09437
[16] J. Kwon. *Tendermint: Consensus Without Mining*. [Online]. Available: https://cdn.relayto.com/media/files/LPgoWO18TCeMIggJVakt_tendermint.pdf
[17] H. Chen and Y. Wang, "SSChain: A full sharding protocol for public blockchain without data migration overhead," *Pervasive Mobile Comput.*, vol. 59, Oct. 2019, Art. no. 101055.
[18] M. Zamani, M. Movahedi, and M. Raykova, "RapidChain: Scaling blockchain via full sharding," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2018, pp. 931–948.
[19] M. H. Manshaei, M. Jadliwala, A. Maiti, and M. Fooladgar, "A game-theoretic analysis of shard-based permissionless blockchains," *IEEE Access*, vol. 6, pp. 78100–78112, 2018.
[20] M. Conti, E. S. Kumar, C. Lal, and S. Ruj, "A survey on security and privacy issues of bitcoin," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 3416–3452, 4th Quart., 2018.
[21] R. E. Korf, "A complete anytime algorithm for number partitioning," *Artif. Intell.*, vol. 106, no. 2, pp. 181–203, Dec. 1998.
[22] Albert. (2019). *Sharding FAQ*. [Online]. Available: https://github.com/ethereum/wiki/wiki/Sharding-FAQ

JUSIK YUN received the B.S. degree in electrical and electronic engineering from Yonsei University, Seoul, South Korea, in 2016, where he is currently pursuing the combined M.S. and Ph.D. degrees in electrical and electronic engineering and is also a Researcher with the Communications and Networking Laboratory (CNL). His current research interests include blockchain, trust management systems, and machine learning.

YUNYEONG GOH received the B.S. degree in electrical and electronic engineering from Yonsei University, Seoul, South Korea, in 2019, where he is currently pursuing the combined M.S. and Ph.D. degree in electrical and electronic engineering and is also a Researcher with the Communications and Networking Laboratory (CNL). His current research interests include blockchain, trust systems, and augmented reality.

JONG-MOON CHUNG received the B.S. and M.S. degrees in electronic engineering from Yonsei University and the Ph.D. degree in electrical engineering from Pennsylvania State University. From 1997 to 1999, he was an Assistant Professor and an Instructor with Pennsylvania State University. From 2000 to 2005, he was with Oklahoma State University (OSU) as a tenured Associate Professor. Since 2005, he has been a Professor with the School of Electrical and Electronic Engineering, Yonsei University, where he is currently the Associate Dean of the College of Engineering. He is also a Pledge Book Award Winning Member of Eta Kappa Nu (HKN). He received the Outstanding Accomplishment Faculty Awards in 2008, 2018, and 2019, and the Outstanding Teaching Awards from Yonsei University, in 2007, 2009, 2014, and 2019, the Korean Government's Defense Acquisition Program Administration (DAPA) Award, in 2012. As a tenured Associate Professor at OSU, in 2005, he received the Regents Distinguished Research Award and the Halliburton Outstanding Young Faculty Award. He received the Distinguished Faculty and the Technology Innovator Award from OSU, in 2003 and 2004, respectively, and the First Place Outstanding Paper Award at the IEEE EIT 2000 Conference held in Chicago, USA, in 2000. In addition, his 12 Coursera courses are among the most popular, which focus on deep learning, big data, cloud computing, 5G and 4G mobile communications, Wi-Fi, Bluetooth, augmented reality (AR), the Internet of Things (IoT), Skype, YouTube, TCP/IP, and hardware and software of smartphones and smartwatches. He will be serving as the General Chair for the IEEE ICCE 2022 and the IEEE ICCE-Asia 2020. He was the General Chair for several conferences, including the IEEE MWSCAS 2011. He is also serving as a Vice President for the IEEE Consumer Electronics Society, an Editor for the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, an Associate Editor for the IEEE TRANSACTIONS ON CONSUMER ELECTRONICS, a Section Editor for the *ETRI Journal* (Wiley), and a Co-Editor-in-Chief for the *KSII Transactions on Internet and Information Systems*.