RTChain: A Reputation System with Transaction and Consensus Incentives for E-commerce Blockchain

YOU SUN, RUI XUE, RUI ZHANG, and QIANQIAN SU, State Key Laboratory of Information Security (SKLOIS), Institute of Information Engineering, CAS School of Cyber Security, University of Chinese Academy of Sciences, China
SHENG GAO, School of Information, Central University of Finance and Economics, China

Blockchain technology, whose most successful application is Bitcoin, enables non-repudiation and non-tamperable online transactions without the participation of a trusted central party. As a global ledger, the blockchain achieves the consistency of replica stored on each node through a consensus mechanism. A well-designed consensus mechanism, on one hand, needs to be efficient to meet the high frequency of online transactions. For example, the existing electronic payment systems can handle over 50,000 transactions per second (TPS), while Bitcoin can only handle an average of about 3TPS. On the other hand, it needs to have good security and high fault tolerance; that is, in the case when some nodes are captured by adversaries, the network can still operate normally. In this article, we establish a reputation system, called RTChain, to be integrated into the e-commerce blockchain to achieve a distributed consensus and transaction incentives. The proposed scheme has the following advantages. First, an incentive mechanism is used to influence the consensus behavior of nodes and the transaction behavior of users, which in turn influence the reputation scores of both nodes and users. That is, when a node correctly processes a transaction, it will receive the corresponding reputation value as a reward, and the reputation value will be reduced as punishment not only when the node is dishonest and violates the consensus agreement but also the transaction is not completed as required. Just like electronic transactions in the real world, the higher the reputation of the user, the more likely it is to be selected as the transaction partner. A user with a low reputation will be gradually eliminated in our system because it is difficult to complete the transaction. Second, RTChain uses a verifiable random function to generate the leader in each round, which guarantees fairness for all participants and, unlike PoW, does not consume a large amount of computing resources. Then our consensus mechanism selects the nodes with high reputation scores to reduce the number of nodes participating in the consensus, thus improving the consensus efficiency, so that RTChain's throughput can reach 4,000TPS. Third, we built a reputation chain to implement the distributed storage and management of reputation. Finally, our consensus mechanism is secure against existing attacks, such as flash attacks, selfish mining attacks, eclipse attacks, and double spending attacks, and allows nodes that participate in the consensus to fail, as long as the reputation of the failure node does not exceed one-third of the total reputation. We build a prototype of RTChain, and the experimental results show that RTChain is promising and deployable for e-commerce blockchains.

The authors acknowledge the support from the National Key R&D Program of China under Grant No. 2017YFB1400700; the National Natural Science Foundation of China under Grant No. 61772514 and 62072487; and Beijing Municipal Science & Technology Commission (Project Number: Z191100007119006).
Authors' addresses: Y. Sun, R. Xue, R. Zhang (corresponding author), and Q. Su, State Key Laboratory of Information Security (SKLOIS), Institute of Information Engineering, CAS, School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China, 100093; emails: {sunyou, xuerui, zhangrui, suqianqian}@iie.ac.cn. S. Gao, School of Information, Central University of Finance and Economics, Beijing 100081, China; email: sgao@cufe.edu.cn.

1533-5399/2020/12-ART15 $15.00
https://doi.org/10.1145/3430502

CCS Concepts: • **Information systems → Reputation systems**; • **Security and privacy → Security protocols**; • **Applied computing → Secure online transactions**;

Additional Key Words and Phrases: Blockchain, consensus mechanism, reputation system, e-commerce, PBFT

ACM Reference format:
You Sun, Rui Xue, Rui Zhang, Qianqian Su, and Sheng Gao. 2020. RTChain: A Reputation System with Transaction and Consensus Incentives for E-commerce Blockchain. *ACM Trans. Internet Technol.* 21, 1, Article 15 (December 2020), 24 pages.
https://doi.org/10.1145/3430502

1 INTRODUCTION

With the stable operation and growing use of Bitcoin [29] for many years, Bitcoin's underlying technology, blockchain, has gradually attracted widespread attention in industry and academia. In recent years, the nature and characteristics of blockchain technology have made it applicable not only to the field of digital cryptocurrencies such as Bitcoin but also as an innovative technical framework in fields that require the establishment of a distributed, point-to-point trust relationship such as e-commerce.

The consensus mechanism is the core technology of the blockchain, which guarantees the consistency of replica stored on each node without the participant of any trusted party. A well-designed consensus mechanism, on one hand, needs to be efficient to meet the high frequency of online transactions. For example, the existing electronic payment systems can handle over 50,000 transactions per second (TPS), while Bitcoin can only handle 3TPS. On the other hand, it needs to have good security and high fault tolerance; that is, even when some nodes are captured by adversaries, the network can still operate normally.

The most popular and widely used consensus mechanism is the Proof-of-Work (PoW), which is used in the mainstream blockchain platforms such as Bitcoin and Ethereum. PoW relies on miners to continuously mine (working on a cryptography puzzle) to maintain the normal operation of the system and achieve consistency of blockchain, and full nodes distributed around the world in many independent jurisdictions validate transactions and blocks. However, the process of mining consumes substantial computing resources and leads to a concentration of global mining power, which enables 51% of attacks. Moreover, it needs to wait for six confirmations (60 minutes on average) to commit a transaction throughout the network. Consensus algorithms that are not based on computing power, such as Proof-of-Stake (PoS) and Delegated Proof-of-Stake (DPoS), have not been proved theoretically. The strong consistency algorithm such as the Practical Byzantine Fault Tolerant algorithm (PBFT) has the disadvantages of high algorithm complexity and low degree of decentralization. Therefore, the design and implementation of a secure and efficient consensus mechanism is one of the major requirements in the application of blockchain technology to conduct secure electronic transactions.

In addition, the openness and dynamics (nodes join and leave the network) of the peer-to-peer (P2P) network greatly increase its security risks. In such a network, there is no centralized authority to authenticate nodes (users) and manage the network. In the e-commerce blockchain, the nodes (users) need to judge and select the transaction partners by themselves. Untrustworthy and improper cooperation between nodes will increase the security risk of the system. Thus, it is necessary to establish a complete reputation system, which can provide the basis for the nodes to select the transaction partners and constrain the node behavior. However, existing reputation systems used in blockchains focus only on consensus mechanisms and do not reflect the behavior of users in the transaction. That is, there is no guarantee that both parties of the transaction will

comply with the pre-defined rules. If dishonest or malicious users deliberately violate the transaction rules, it will compromise the security of the blockchain network. Combining the reputation system with the consensus mechanism and transaction behavior allows both parties of the transaction to freely choose the service provider based on their reputation and can better constrain the user's behavior in the consensus and transaction processes, thus enabling the entire blockchain system to be secure and reliable.

1.1 Contributions

In this article, we design a complete reputation system with transaction and consensus incentives, named RTChain, for e-commerce blockchain, which mainly divides the user's reputation into consensus reputation and transaction reputation. The main contributions are described as follows.

- First, in RTChain, the behavior in the processes of consensus and transaction will affect the user's reputation. After each block is determined, the reputation of the node will be recalculated. The behavior of a node in the system will affect its reputation value; nodes with high reputation will be more likely to be selected as transaction partners and there will be more rewards in the consensus.
- Second, we propose a new reputation-based consensus mechanism with the following three characteristics: (1) We use a verifiable random function to select the leader, rather than solving a puzzle like the Bitcoin system, so that our scheme does not consume substantial computing power. (2) We choose a subset of the nodes in the system with high reputation scores to implement the consensus mechanism. Reducing the number of consensus nodes decreases the system load and network traffic, thus improving the efficiency of consensus. (3) Reputation scores will not only affect the weight of nodes voting in the consensus but also make the nodes with high reputation value more likely to receive the rewards through the weighted sampling algorithm. The dishonest users will be punished with lowering reputation scores. This incentive mechanism will enable users to honestly complete transactions and consensus agreements and make the network more robust.
- Third, we use a reputation chain to store and update the reputation of the nodes, without the need for trusted third parties to manage reputation.
- Fourth, we provide a detailed security analysis of RTChain. The analysis shows that RTChain has the properties of persistence, liveness, and compatibility, and can resist most network attacks such as flash attacks, selfish mining attacks, eclipse attacks, and double spending attacks.
- Finally, we build a prototype of the RTChain and evaluate its performance in terms of the time cost of transaction confirmation and consensus and the throughput of the system. The experiment results show that the throughput of RTChain can achieve 4,000TPS; thus, RTChain is efficient and deployable compared with existing consensus algorithms.

1.2 Article Organization

The rest of the article is organized as follows. In Section 2, we introduce the existing consensus mechanisms, especially reputation-based consensus mechanisms, and their advantages and disadvantages. Section 3 introduces the detailed process of our consensus mechanism. In Section 4, we introduce the reputation system, including reputation in consensus, reputation in transaction, and reputation storage and management. Section 6 gives the security analysis of our scheme. We implemented the system and evaluated its efficiency in Section 7. Finally, we conclude the work in Section 8.

2 RELATED WORK

The consensus mechanism is a method for determining the accounting rights of distributed ledgers (i.e., blocks) according to a pre-negotiated rule among distributed nodes so that different nodes can reach consensus on transaction data and ensure the consistency and authenticity of the distributed ledger data.

2.1 Strong Consistency Consensus and Eventual Consistency Consensus

According to various criteria, consensus algorithms can be divided into two classes: strong consistent consensus algorithms and eventually consistent (probabilistic) consensus algorithms.

Strong consistency consensus algorithms are mostly used in private blockchains and consortium blockchains where the number of nodes is small and there is a stronger requirement for consistency and correctness. Typical strong consistency consensus algorithms include the Byzantine Fault Tolerance (BFT) mechanism [25], the PBFT mechanism [27], the Paxo mechanism without considering Byzantine faults [24], and the Raft mechanism [31], among others.

With the development and application of blockchain technology, researchers have put forward some new strong consistency consensus algorithms. HoneyBadgerBFT [28] is the first practical asynchronous BFT protocol and therefore does not require any time hypothesis. Even when the network is unreliable (provided the network remains connected), HoneyBadgerBFT can track the available bandwidth in the network so that the message is finally delivered to each node. HotStuff [36] combines the view change process with the normal process to reduce the complexity of view change. In addition, the threshold signature is introduced to reduce the complexity of message verification. SBFT [16] is a state-of-the-art Byzantine fault tolerant permissioned blockchain system that addresses the challenges of scalability, decentralization, and world-scale geo-replication. Compared with the highly optimized system that implements the PBFT protocol, SBFT provides almost twice the throughput. The next 700 BFT protocols [4] present a new abstraction for designing and reconfiguring generalized replicated state machines. Abstract can be used to considerably simplify the incremental development of efficient Byzantine fault-tolerant state machine replication protocols that are notorious for being difficult to develop.

Eventually consistency consensus algorithms are mostly used in public blockchains with a large number of nodes, and it is difficult to achieve 100% consistency and correctness for all nodes. Typical eventual consistency consensus algorithms include PoW [11, 18], PoS [10, 21, 35], and DPoS [1], among others.

Among the eventual consistency consensus algorithms in recent years, Snow White [6] and Ouroboros [20] are two competitive algorithms. Snow White solves the problem of the dynamic distribution of stakeholders and uses a corruption delay mechanism to protect the security of the system. Snow White is the first provably secure, robust, and reconfigurable PoS consensus protocol. Even if the adversary controls a small portion of the stake, it can still maintain the security of the system. Ouroboros is also a blockchain consensus protocol based on PoS and capable of providing strict security guarantees (the security attributes of two robust transaction ledgers: persistence and liveness). Ouroboros combined new reward mechanisms to motivate the PoS protocol and proved that honest behavior under this mechanism is close to the Nash equilibrium and therefore can withstand attacks such as selfish mining.

Hybrid consensus mechanisms combine different consensus mechanisms. AlgoRAND [9] combines POS and BFT; the algorithm has a new property, participant replaceability, which makes it more secure in an adversarial environment. The main advantages of AlgoRAND are (1) the probability of forking is negligible, (2) the computational complexity of the algorithm is small, and (3) the time for generating a block is close to the time of propagation of the block in the network. ELASTICO [26] and Omniledger [23] are good combinations of PoW and PBFT. ELASTICO's core

idea is to divide the nodes in the network into several small committees; each committee will deal with disjoint transaction sets. Omniledger uses a bias-resistant public-randomness protocol to select large, statistically representative shards to process transactions and introduces an efficient cross-shard commit protocol that atomically handles transactions affecting multiple shards, so as to ensure the security and correctness. Chainspace [2] and RapidChain [38] combine the PoW and BFT protocol. On the basis of sharding, Chainspace constructs the smart contract application platform, which realizes the communication sharding and calculation sharding of transactions and smart contracts. Rapidchain is the first public blockchain protocol based on sharding. It can resist Byzantine faults of up to a third of participants and achieves complete sharding of the communication, computation, and storage overhead of transactions without assuming any trusted setup.

In summary, the strong consistency consensus algorithm is more secure, but it has a higher complexity and is a multi-center mechanism. The eventual consensus algorithm has a higher degree of decentralization and a lower complexity of the algorithm, but it is less secure. How to design security and efficient consensus algorithms for cross-secure domain data sharing applications requires further research.

2.2 Reputation-Based Consensus

In recent years, some scholars have introduced reputation systems into blockchains to improve reliability and efficiency. Carboni provides a decentralized and distributed feedback management system in his paper [8] . It can be built on top of the Bitcoin protocol and it is directly implementable on top of the Bitcoin network. Using feedback management, which is a decentralized architecture without a single point of control, makes the system decentralized, secure, and global. A reputation-based consensus protocol for the peer-to-peer network called Proof of Reputation (PoR) [14] is proposed by Gai et al. where reputation serves as the incentive for good behavior, and the service provider who has the highest trust value can publish a new block. Participants improve their trust score by providing a certain service and broadcasting the transaction honestly. Yu et al. propose a consensus mechanism based on reputation called RepuCoin [37]. A miner's power is decided by its reputation instead of its computing power in a short time range. RepuCoin separates leader election from transaction serialization, and miners solve the Bitcoin-like puzzle to generate a new keyblock. The leader is chosen from a consensus group, which is a group of users who have a high reputation value and commit the transactions into microblocks. The consensus group members need to run a consensus algorithm (such as Byzantine fault-tolerant) with reputation-based weighted voting to determine the final blocks.

However, none of the existing reputation-based consensus mechanisms or reputation systems used in blockchain consider the user's behavior in the process of transaction. For example, if a user does not comply with or willfully breaks the trading rules, if the service provider does not ship the goods after the purchaser pays, or if the purchaser cancels the transaction without any reason, there will be no impact or punishment. Malicious users will exploit this flaw and place the e-commerce blockchain at risk and render it unstable.

In this article, we propose a reputation system with transaction and consensus incentives for the e-commerce blockchain. In our system, every action of the user in the processes of consensus and transaction will affect his or her reputation and the system reaches high efficiency compared with existing consensus mechanisms.

3 THE CONSENSUS MECHANISM

In this section, we will present the basic flow of the reputation-based consensus mechanism. It mainly concerns leader election and block publication. In addition, we also introduce the security assumptions, basic concepts, and a reward system in this section.

3.1 Security Assumptions

In the scheme discussed in this article, the system is asynchronous and distributed, which means that messages in the network may be lost, duplicated, delayed, or out of order in some cases. In addition, in the system proposed in this article, each node is independent, and the failure of each node is an independent event, ensuring that a node failure will not cause other nodes to fail.

We assume that there is a malicious adversary A in the system, and that adversary A can manipulate multiple failure nodes. In order to ensure the safety and liveness of the system, we assume that when f nodes become faulty, at least $3f + 1$ nodes must be honest; that is, they conform to the protocol rules. In this way, all clients will eventually receive a response to their request, and the delay will not increase indefinitely.

Finally, we assume that the adversary A cannot delay honest nodes indefinitely, and the computing power of the adversary A is limited; he or she cannot crack the encryption algorithm, forge a digital signature, or extract the message content from the hash data.

3.2 Basic Concepts

In our proposed scheme, there are several basic concepts that differ from other consensus mechanisms. In this subsection, we will introduce these concepts in detail.

3.2.1 Users and Nodes. In our scheme, anyone who enters the e-commerce blockchain system at the transaction level is called a *user*. At the same time, at the network level, we call it a *node*. A public key pk_j identifies the user (node) j who owns the corresponding secret key sk_j.

On one hand, in the transaction phase, each transaction has a *service provider* and a *purchaser*. We represent the service provider and the purchaser in a transaction as a group of *transaction partners*. After the service provider provides the corresponding service to the purchaser, the transaction record is broadcasted to the network. The transaction is not completed until it is recorded in a block that is appended to the blockchain.

On the other hand, in the consensus phase, the role of a node can be a *leader*, a *leader candidate*, and a *consensus group member*. The scheme selects leader candidates based on a specific cryptographic method, and the leader is elected from a set of leader candidates. The consensus group is a subset of the most reputable users.

3.2.2 Transaction. In a transaction scenario, the purchaser first needs to send a service request to the service provider:

$$Req = (pk, pk', I, E_{pk}(I'), \sigma), \tag{3.1}$$

where pk is the service provider's public key, pk' is the purchaser's public key, I is the service request information that is not sensitive, I' is the service request information that is sensitive, $E_{pk}(I')$ is the ciphertext of the sensitive information I' encrypted using the service provider's public key pk, and σ is the digital signature of $(pk, pk', I, E_{pk}(I'))$ signed with the purchaser's secret key sk'. It should be noted that pk not only represents the value of the public key but also uses pk to mark a user in our system. Therefore, although the service provider knows its own public key, both sides of the transaction are included in the Req message, so as to retain the complete transaction information in the block.

After the service provider completes the corresponding service, it needs to give the purchaser a response. The form of the response message is as follows:

$$Res = (Req, E_{pk'}(S), \sigma'), \tag{3.2}$$

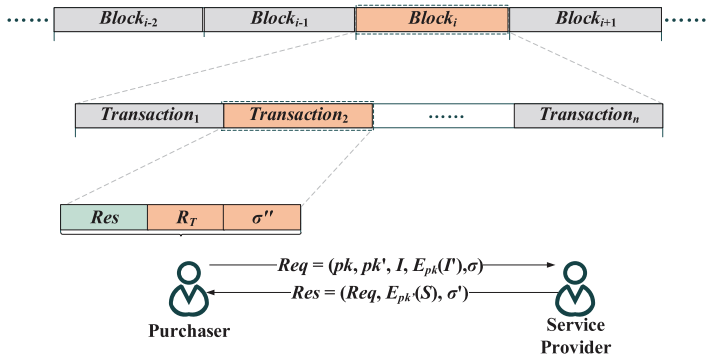

Fig. 1. The transaction structure.

where Req is the original request message, S is information about the service provided, $E_{pk'}(S)$ is the ciphertext of the service information S encrypted using the purchaser's public key pk', and σ' is the signature of $(Req, E_{pk'}(S))$ signed with the provider's secret key.

After the purchaser receives the response message, it checks the service information and gives the service provider a transaction score R_T for this transaction and broadcasts it to the network as a $Transaction$:

$$Transaction = (Res, R_T, \sigma''), \tag{3.3}$$

where Res is the transaction information created by the service provider with its signature, containing the corresponding request message; R_T is the transaction score; and σ'' is the digital signature of (Res, R_T) sign with the purchaser's secret key. This process is shown in Figure 1.

A $TransactionList$ consists of all transactions in the network over a period of time, which is shown as below:

$$TransactionList_i = \{Transaction_1, Transaction_2, ...\}. \tag{3.4}$$

3.3 Consensus Group Election

At the beginning of each round of consensus, members of the consensus group need to be elected. Let G_r denote the set of consensus group members in round r. They are selected from the users with the top $|G|$ highest reputation scores as computed in round $r-1$, where $|G|$ represents the number of members in the consensus group G. The specific value of $|G|$ should be defined during system initialization according to system requirements and can be appropriately adjusted according to the number of users and reputation distribution in the system. Specifically, in this article, the initial value is selected based on the reputation of the consensus group members and is 50% of the reputation of all nodes. The value of $|G|$ that meets this requirement changes with the operation of the system, so it is possible to adjust the $|G|$ after the system runs for a period of time to meet this requirement. User i in the consensus group in round r can be expressed as

$$pk_i \in G_r. \tag{3.5}$$

The consensus group is re-elected at the beginning of each round and can only perform the corresponding operations in this round. Becoming a member of the consensus group and completing its proper operations improve the reputation score of a node, and in addition, the node is awarded a transaction fee. This provides an incentive for members of the consensus group to conform to the rules of the consensus mechanism.

ALGORITHM 1: Leader Election

Input: the user's identity pk_i, the number of round r, the previous block $Block_{r-1}$, and a constant p
Output: a candidate block $Block_{r,i}$

if $H(SIG_{sk_i}(r, Q_{r-1})) \geqslant p$ **then**
  break;
else
  $L_r = L_r \cup pk_i$;
  calculate $Q_{r,i} = H(SIG_{sk_i}(r, Q_{r-1}))$;
  Package transactions in this time period into $TransactionList_{r,i}$;
  Integrate the round number r, parameter $Q_{r,i}$, hash value of $TransactionList_{r,i}$, hash value of $Block_{r-1}$, pk_i, and the timestamp into the $BlockHeader_{r,i}$;
  Let $Block_{r,i} = (BlockHeader_{r,i}, TransactionList_{r,i})$;
  Send $(Block_{r,i}, \sigma_{r,i}, SIG_{sk_i}(H(Block_{r,i})))$ and $\sigma_{r,i}$ to the consensus group G_r respectively.
end

3.4 Leader Election

In a new round of the system, each user needs to calculate if the hash value of the signature signing with his or her own secret key sk_i satisfies the following formula:

$$H(SIG_{sk_i}(r, Q_{r-1})) < p, \tag{3.6}$$

where sk_i is user i's secret key, and r represents the number of rounds of the system. p is a system parameter that determines the size of the set of leader candidates; its value should be set during system initialization and can be adjusted according to the system operation. The larger the value of p, it means that there are more leader candidates, the security of the system is higher, but at the same time its efficiency will be reduced. Q_r is the seed of the rth round and Q_r is calculated as

$$Q_r = \begin{cases} H(SIG_{l_{r-1}}(r, Q_{r-1})) & \text{if } B_{r-1} \text{ is a legal block} \\ H(r, Q_{r-1}) & \text{if } B_{r-1} \text{ is an empty block.} \end{cases} \tag{3.7}$$

Q_0 is a set of random numbers generated during system initialization. l_{r-1} is the leader of round $r-1$. If the system successfully reached consensus in the $r-1$ round, the block $Block_{r-1}$ is a legal block, and the parameter Q_r will be stored in the block r's block header by the leader of round r. The empty block does not contain any transaction, and there will be no leader in this round.

If the user's signature result satisfies Equation (3.6), he or she becomes a leader candidate for this round. In each round, multiple users meet the formula to become leader candidates. Every leader candidate needs to complete the work of the leader in the consensus, but no one can know at this time who the true leader is, including the leader candidates themselves. In this way, malicious attackers can be prevented from attacking the leader of this round.

Each leader candidate packages the transactions in the network in a specific order (e.g., in chronological order of transactions) and integrates them into the block. The leader candidate i in round r should send the block the following message to the consensus group G_r:

$$< Block_{r,i}, \sigma_{r,i}, SIG_{sk_i}(H(Block_{r,i})) >, \tag{3.8}$$

where $\sigma_{r,i} = SIG_{sk_i}(r, Q_{r-1})$. In addition, the leader candidate also needs to send the signature $\sigma_{r,i}$ to the consensus group separately so that in most instances the consensus group members will receive the signature of the leader candidate before receiving the whole message.

The leader election algorithm is shown in Algorithm 1, where L_r is a list of leader candidates in round r.

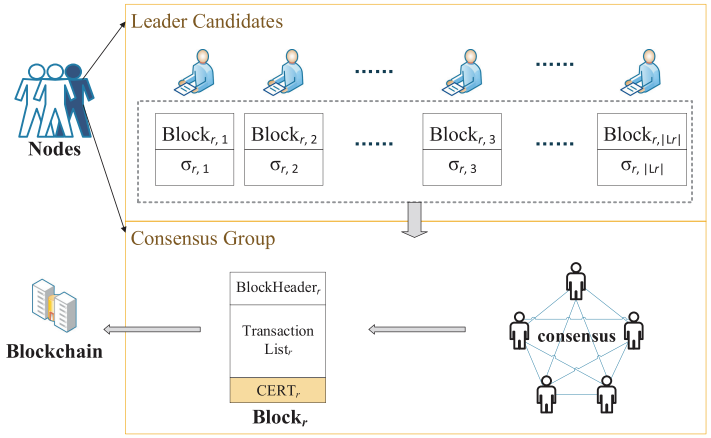

Fig. 2. Block generation of RTChain.

3.5 Block Publication

In this stage, the consensus group determines the final block $Block_r$ of round r.

In the previous stage, the leader candidates send a block message and certificate signature $\sigma_{r,i}$ to the consensus group. Due to the length of the two messages, the consensus group members will probably get the certificate signature σ first. In this way, members of the consensus group can first verify that they are the current leader candidates of this round by hashing the received signature and find the one with the smallest hash value. The leader candidate with the smallest hash value of the signature is the leader in this round and expressed it as l_r, which is the leader of round r.

Each user $pk_i \in G_r$ checks the complete message from the leader l_r, which contains $Block_{r,i}$ and its signature $Sig_{sk_i}(H(Block_{r,i}))$. The consensus group members first check if the signature is legal and, if so, check the block $Block_{r,i}$ further. The consensus group member sends $H(Block_{r,i})$ with its signature to other consensus group members only if the block and the signature of it are both legal. Otherwise, send $\emptyset$ to other consensus group members.

For a period of time, if the consensus group member receives $2f' + 1$ messages that are the same as their own calculated hash value and are not $\emptyset$ (where f' is the number of failure nodes in the consensus group), and the signatures of these messages are correct, he or she sends a commit message to other consensus group members. A commit message contains the leader of this round l_r, $Block_{r,i}$, and its hash $H(Block_{r,i})$:

$$< COMMIT, l_r, Block_{r,i}, H(Block_{r,i}) > . \tag{3.9}$$

If there are more than $2f' + 1$ commit messages, a consensus is reached on the newly generated block $Block_{r,i}$ of the leader l_r in round r. This block will be issued to the blockchain with a certificate $CERT_r$ issued by the consensus group G_r as block $Block_r$. Among them, in the consensus process of the consensus group, each message (vote) they send will be signed with their private key, and $CERT_r$ is the collection of signed commit votes. This process is shown as Figure 2.

For other situations that may exist in the system, if the consensus group did not reach a consensus, or the leader posted an illegal block, the system will generate an empty block in this round. That is to say, each block in the blockchain is in one of the two forms: if the leader of the rth round correctly generates the transaction list and block, and the consensus group reaches the consensus,

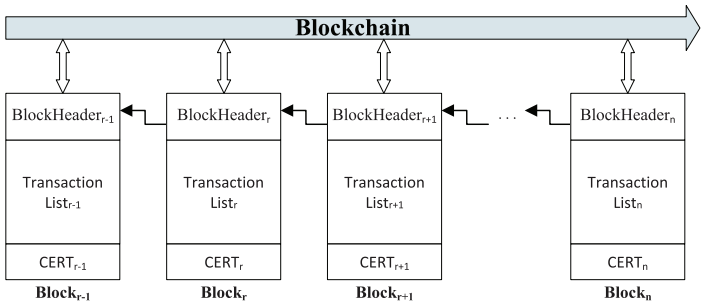

Fig. 3. The blockchain structure.

then

$$\begin{aligned} Block_r =& (r, Q_r, H(Block_{r-1}), H(TransactionList_r), \\ & pk_{l_r}, timestamp, TransactionList_r, CERT_r); \end{aligned} \tag{3.10}$$

if not, then

$$Block_r = (r, Q_r, H(Block_{r-1}), timestamp, \phi, CERT_r). \tag{3.11}$$

Separating transaction serialization and block publishing can prevent users from generating multiple keys to influence system operation. When a user has multiple keys, he or she will have a greater probability of becoming a leader, but after the leader packages the transaction, it needs the consensus group to check and reach a consensus before it can be published on the blockchain. The selection of the consensus group is based on the user's reputation score, which requires the user's long-term good performance in the system, and we will introduce it in detail later. In this way, the system can avoid Sybil attacks.

Finally, the blockchain structure generated by our proposed scheme is shown in the Figure 3.

3.6 Consensus Group Vote

In Section 4, we will introduce that the user's reputation is determined by the consensus reputation and transaction reputation. The selection of members of the consensus group is based on the consensus reputation. The weight of voting for each consensus group member is different. That is to say, in the above consensus process, the members of the consensus group do not have to wait until more than two-thirds of the votes are received, but the sum of the weights of the votes received by them exceeds two-thirds of R_G, where R_G is the sum of the reputation of the consensus group members as computed in round $r-1$.

In our scheme, the weight of the consensus group members' votes is determined by their final reputation (consensus reputation and transaction reputation). In the last two steps of the consensus, the consensus group members only have to wait for the total reputation of the votes received to be greater than $\frac{2R_G}{3}$.

One advantage of this is that only users who correctly participate in the consensus mechanism are more likely to enter the consensus group. Users cannot enter the consensus group by initiating more transactions, thereby increasing their transaction reputation.

3.7 Rewards System

Similar to most blockchain systems, there are two types of rewards in our system. First, the user who is eventually selected as the leader and correctly completes his or her task will receive a reward. This reward is similar to the mining reward in Bitcoin, and the amount of this award

should be set in advance in the system. Just as the mining reward in Bitcoin will be adjusted according to the length of the blockchain, we hope that RTChain can be applied to more different application scenarios, so RTChain can also be adjusted according to the operating status of the system.

In our system, another form of reward comes from transaction fees. Each transaction that is integrated into the transaction blockchain requires a certain transaction fee. The amount of the transaction fee is pre-set in the system and is felt by many factors such as the transaction amount. In RTChain, it has an accompanying crypto-currency according to different transaction scenarios. For example, when we use RTChain in electronic transactions, the system uses electricity coin as crypto-currency. Users use electricity coins to conduct electricity transactions, and transaction fees are also paid in the form of electricity coins. Each transaction fee in the block $Block_r$ of the tth round will be distributed to some consensus group members of the round r or the leader l_r according to the following rules. We represent the transactions of the $TransactionList_r$ in the rth round as $\{Transaction_1, Transaction_2, ..., Transaction_n\}$. In this subsection, the nodes we consider are the set of consensus group members and the leader in the rth round (i.e., a total of $(|G|_r + 1)$ nodes). A random number $\omega_i \in (0, 1)$ is generated for each of the nodes i. Then calculate

$$S_i = \omega^{\frac{1}{R_i}}, \tag{3.12}$$

where R_i is the consensus reputation of node i, which we will cover in detail in the next section. Select the corresponding n nodes with the highest S to obtain the transaction fee of n transactions in the $TransactionList_r$. In the above weighted sampling algorithm, any member of the leader and consensus group numbers may receive a transaction fee, but the higher the user's reputation, the more likely it is to obtain a transaction fee. This will incentivize users to honestly implement consensus agreements to improve their reputation. In this process, we will introduce a smart contract to calculate user rewards. Smart contracts are written and executed in code, combined with the traceability and non-tampering characteristics of the blockchain, to avoid errors and disputes in artificially calculating rewards. We write the above-mentioned reward distribution method into the smart contract. After each round, the smart contract is automatically executed and the reward is distributed to each user.

On the other hand, in the transaction blockchain, although the identity information of the service provider and the purchaser will be protected, its reputation will be disclosed. This means that the increase or decrease of their reputation can also be a form of reward and punishment for users. When the user's reputation is higher, he or she will be more likely to be selected as the transaction partner. Every action a user takes when participating in the system will affect his or her reputation. We will introduce the detailed reputation mechanism in Section 4.

4 REPUTATION SYSTEM

In this section, we define the reputation system used in the consensus mechanism in detail. In the process of users' participation in the consensus mechanism or after completing a transaction, users' reputation will change. The notations we will use are defined in Table 1.

4.1 Reputation in Consensus Mechanism

In this part of the reputation system, the reputation score of a node depends on its performance in the consensus mechanism. In each round, if the node is not selected as a leader or a member of the consensus group, its consensus reputation score will not change. For the node participating in the consensus, the reputation score will increase if it performs its work as a leader or a member

Table 1. Notations

Notation	Explanation
r	the number of rounds;
x_l	the number of times that the node has become a leader;
x_w	the number of times that the node has misbehaved when it becomes a leader;
α	the reputation system parameter, $\alpha \in [0, 1]$;
β	the consensus reputation system parameter, $\beta \in [0, 1]$;
k	the number of the consensus group members and leader performing their work correctly in consensus mechanism;
m	the number of consensus group members who behave incorrectly and vote differently from the majority vote;
n	the number of nodes in the system;
$p_i^{(r)}$	the consensus reputation score of node i in round r;
$T_i^{(r)}$	the transaction reputation score of node i in round r;
t_{ij}	the transaction score for node j scoring by node i;
c_{ij}	the normalizing transaction score for node j scoring by node i;
$R_i^{(r)}$	the reputation score of node i in round r;
R_T	the score of the transaction, which has the same meaning as t_{ij}.

of the consensus group. Conversely, if it behaves improperly, its reputation score will drop. Next, we will define the reputation score in the consensus mechanism in detail.

Let $p_i^{(r)} \in [0, 1]$ be the consensus reputation of node i in round r. The consensus reputation of node i at the end of round $r + 1$ is determined based on the previous consensus reputation $p_i^{(r)}$ and the performance in the current round of the consensus. The consensus reputation of round $r + 1$ is calculated as

$$p_i^{(r+1)} = \begin{cases} p_i^{(r)} & \text{(if node } i \text{ is not selected as the leader or as a member of consensus group)} \\ p_i^{(r)} + \frac{1}{k}(1 - p_i^{(r)}) & \text{(if node } i \text{ is selected as the leader or as a member of consensus and performs its work)} \\ \frac{m-1}{m} p_i^{(r)} & \text{(if consensus group member } i \text{ behaves incorrectly and votes differently from the majority vote)} \\ \beta p_i^{(r)} & \text{(if leader } i \text{ misbehaves and submits a conflicting transactions),} \end{cases} \tag{4.1}$$

where k is the number of the consensus group members and leaders performing their work correctly, m is the number of consensus group members who behave incorrectly and vote differently from the majority vote, and $\beta \in [0, 1]$. In our scheme, $\beta = \frac{x_l - x_w}{x_l}$, where x_l is the number of times that the node has become a leader, and x_w is the number of times that the node has misbehaved and submitted conflicting transactions when it became a leader. In this way, the more misconduct of the leader, the greater the impact on its reputation.

If node i does not participate in any work in this round of consensus, its consensus reputation will not change compared with the previous round. In addition to this, the node may be a leader or a consensus group member in this round. As described in Section 3, the function of a leader is to gather validated transactions into a block. If the fees are significant compared to the block reward, this will encourage the leader to package as many transactions in the current network as possible into the block. In addition, if the block is legal and eventually added to the blockchain, the consensus reputation value of this leader will also be improved by $p_i^{(r+1)} = p_i^{(r)} + \frac{1}{k}(1 - p_i^{(r)})$. However, if the leader submits conflicting transactions that cause the block to be illegal, its consensus reputation will decrease. The extent of its decline in its consensus reputation is determined by the system parameter β. Another role in the consensus mechanism is the members of the consensus group whose duty is to select the leader from the leader candidates and implement a consensus mechanism to determine the final block. If node i in consensus group G^{r+1} of round $r + 1$ sends legitimate messages at every step of the consensus mechanism, then its consensus reputation will improve by $p_i^{(r+1)} = p_i^{(r)} + \frac{1}{k}(1 - p_i^{(r)})$. Under these circumstances, when the node i has a lower consensus reputation, a correct consensus behavior will increase its consensus reputation score more significantly. This encourages lower-reputation users to better implement consensus agreements. Another situation in the $r + 1$ round of members of the consensus group is that its vote is in conflict with most people (in PBFT, its vote conflicts with two-thirds of consensus team members). At this time, the consensus reputation of node i will decline by $\frac{m-1}{m} p_i^{(r)}$.

4.2 Reputation in Transactions

In the previous section, we mentioned that in each published transaction, the purchaser's rating R_T for the service provider is included. Let t_{ij} be the transaction score for node j as scored by node i; the range of t_{ij} is $[-1, 1]$. If t_{ij} is less than 0, it means bad transaction behavior; otherwise, it is good. When no transaction occurs between i and j or $i = j$, then $t_{ij} = 0$. t_{ij} has the same meaning as R_T, which emphasizes both sides of the transaction. In order to express it more clearly, we use t_{ij} in the calculation of transaction reputation. c_{ij} is the normalizing transaction score that can be calculated by

$$c_{ij} = \frac{t_{ij} + 1}{\sum_k (t_{ik} + 1)}. \tag{4.2}$$

This makes the sum of all the standardized transaction scores of node i for all other nodes equal to 1, and each standardized transaction score is in $[0, 1]$. This reduces the influence on the node's final transaction reputation that node i maliciously gives other nodes too high or too low scores. Note that if node i has multiple transactions with node j (node i has multiple transaction scores for node j), the numerator in the above equation should be the sum of them.

We define the transaction score matrix C to be $[c_{ij}]$, which contains transaction scores for all users in the system. After each round, the normalizing transaction scores c_{ij} will be recalculated and the matrix C will also be recalculated. The node's transaction reputation scores in the $r + 1$-th round are determined by its transaction reputation in the rth round and the updated transaction score matrix C. The specific calculation method is as follows:

$$\overrightarrow{T^{(r+1)}} = (C^T)\overrightarrow{T^{(r)}}, \tag{4.3}$$

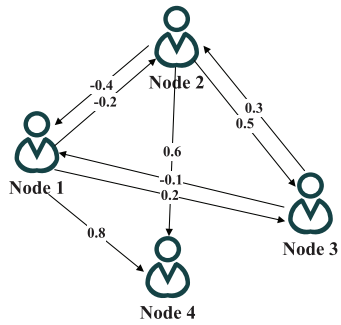

Fig. 4. Transaction scores in round r+1.

where $\overrightarrow{T^r}$ is the transaction reputation scores of each node in the rth round. When the system was first established, we defined the initial transaction reputation score of each node as

$$\overrightarrow{T^{(0)}} = \left[\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n}\right]^T, \tag{4.4}$$

where n is the number of nodes in the system. In some specific transaction environments, the initial reputation of some nodes can also be set higher. Our system allows these parameters to be adjusted for different scenarios.

We will give an example of four nodes to better understand the reputation system. At the end of the rth round, we assume that the transaction reputation of the four nodes is

$$\overrightarrow{T^{(r)}} = [0.4, 0.15, 0.2, 0.25]^T.$$

In the $r + 1$-th round, the transaction scores of these nodes are shown in Figure 4. We can standardize the transaction reputation score, get the reputation matrix C, and calculate the transaction score after the end of this round:

$$\overrightarrow{T^{(r+1)}} = (C^T)\overrightarrow{T^{(r)}} = \begin{bmatrix} 0.21 & 0.17 & 0.25 & 0.38 \\ 0.25 & 0.18 & 0.27 & 0.29 \\ 0.21 & 0.31 & .024 & 0.24 \\ 0.25 & 0.25 & 0.25 & 0.25 \end{bmatrix}^T \begin{bmatrix} 0.4 \\ 0.15 \\ 0.2 \\ 0.25 \end{bmatrix}.$$

Therefore, we get the transaction reputation scores of the four nodes after the end of the new round:

$$\overrightarrow{T^{(r+1)}} = [0.23, 0.22, 0.25, 0.31]^T.$$

4.3 Global Reputation System

In the global reputation system, we will combine the reputation scores in transactions and the consensus to derive the change in reputation of each node in each round.

The reputation of each node in the blockchain system will be recalculated at the end of each round, and the latest reputation value for each node can be known by all nodes in the blockchain. Below we will introduce the calculation method of reputation R.

Vector $\overrightarrow{R^{(r)}}$ represents the reputation value of each node in the rth round; we define

$$\overrightarrow{R^{(r)}} = (1 - \alpha)\overrightarrow{T^{(r)}} + \alpha\overrightarrow{p^{(r)}}, \tag{4.5}$$

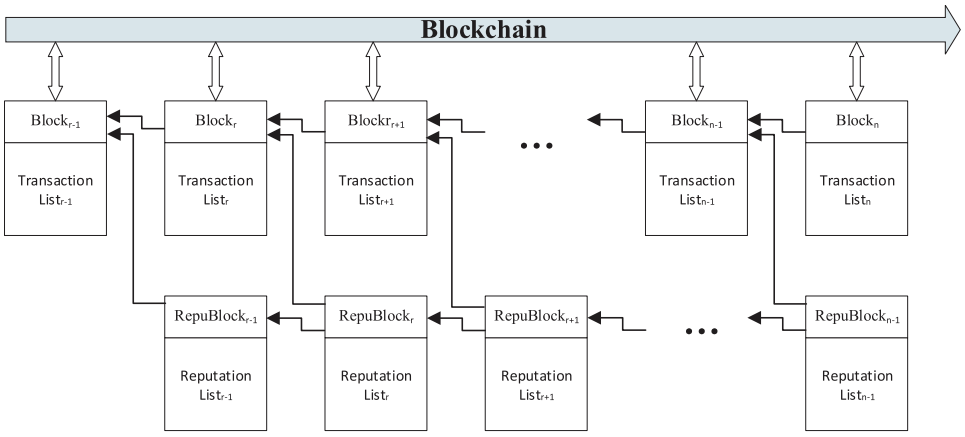

Fig. 5. Transaction blockchain with reputation blocks.

where $\overrightarrow{T^{(r)}}$ is the transaction score of each node in round r, $\overrightarrow{p^{(r)}}$ is the consensus reputation score of each node in round r, and α is a system parameter in $[0, 1]$. The system parameter α will be set at system initialization. In general, the value of α should be 0.5. In this case, the proportion of consensus reputation and transaction reputation in the total reputation is the same. But we hope that our solution can be applied to more different application scenarios, so α can also be adjusted according to demand. It can be seen that the reputation of the node is determined by the transaction reputation score and the consensus reputation score; the size of α determines the proportion of the transaction scores and the consensus reputation scores when measuring the user's reputation value. The greater the value of α, the greater the impact of the consensus reputation score. Therefore, when the system pays more attention to the user's consensus performance, it can increase the value of α; on the contrary, when the system pays more attention to the user's transaction performance, it decreases the value of α.

5 REPUTATION STORAGE AND MANAGEMENT

In this section, we improve the reputation system built in Section 4. We introduced a reputation chain to store reputation scores and update their changes using incremental storage, which we will explain in detail in this section.

5.1 Reputation Storage

Currently, in most transaction systems with reputation assessments, reputation is stored as one of the user's attributes in a central authority and managed and updated by the central authority. In our scheme, reputation does not require any central authority to manage, but rather is stored through a chain of specialized reputation blocks. The structure of a transaction blockchain with reputation blocks is shown in Figure 5.

In this section, we focus on the reputation block of the above scheme. The reputation block consists of two parts: the block header and the main block:

$$RepuBlock_i = (RepuBlockHeader_i.ReputationList_i). \tag{5.1}$$

The main block contains a list of reputations in the current round, which consists of multiple transactions, each of which represents the reputation change of a node. The form of each transaction

in the reputation block is

$$RepuTransaction = (pk_i, T_i, p_i, R_i, \sigma), \tag{5.2}$$

where pk_i is node i's public key recorded by this reputation transaction; T_i, p_i, and R_i are the updated transaction reputation score, consensus reputation score, and total reputation score, respectively; and σ is the signature of the block packer (the leader l_r of round r) on the transaction.

In addition to the information in the general blockchain's block header, the block header of the transaction chain needs to contain the hash value of its corresponding block in the transaction blockchain. The block header of the reputation block of the ith round contains

$$RepuBlockHeader_i = (i, H(ReputationList_i), H(RepuBlock_{i-1}), H(Block_i), timestamp). \tag{5.3}$$

5.2 Reputation Management

The reputation block corresponding to block $Block_r$ of the rth round is $RepuBlock_r$. Block $RepuBlock_r$ stores the reputation of the nodes after the rth round. These reputation changes will be calculated and released by the leader of the $r+1$-th round. (In the consortium blockchains, some privileged nodes can calculate and publish the reputation changes as the system's reputation administrator). In addition, to the reach a consensus on the transaction block of the rth round, the consensus group of the $r+1$-th round needs to reach a consensus on the reputation block of the rth round. That is to say, in the transaction blockchain with reputation blocks, the consensus group of the rth round is no longer based on the reputation of nodes after the completion of round $r-1$, but on round $r-2$, because at the beginning of the rth round, the reputation block of the $r-1$-th round has not yet reached a consensus. In this way, distributed management of reputation can be achieved. After the leader generates the reputation block, all nodes of the network can jointly maintain and manage the reputation chain. In addition, the reputation score is also related to the calculation of reward. Similar to the consensus group election, the reward calculation of round r also uses the reputation of round $r-2$. As described in Section 3.6, a smart contract is introduced in this article to perform the reward calculation. Therefore, the workflow of the rth round reward calculation is as follows:

- the leader l_{r-2} commits reputation calculation of round $r-2$ in the $RepuBlock_{r-2}$;
- the smart contract takes reputation block as an input for reward calculation;
- the contract is executed along with other transactions to extend the public blockchain.

In particular, we will use incremental storage to reduce the size of the reputation blocks: a reputation block only saves the node reputation that changed compared to the previous round. If the reputation of a node has not changed in this round, the data of the node will not be saved in the reputation block of this round. To find the reputation information of a node, we can traverse the reputation chain from the back to the front until the reputation score of the node is found in the reputation block. The number of blocks to traverse does not exceed the height of the current reputation block. Users can also locally establish a reputation list that contains the reputation information of all system users and update the local reputation list through the reputation chain after each round. In this case, the user does not need to traverse the previous reputation block repeatedly, and the total overhead does not exceed the overhead of traversing the reputation chain once. Each reputation block only saves the final reputation of the nodes after the end of the round. No matter how many times the reputation of the node changes in this round, only one data of the node is saved in the reputation block. Assuming that the ratio of the active node to the total number of nodes in the system is a, the compression ratio of the storage mode is at least $1-a$.

Table 2. Attack Resilience

Attacks	Bitcoin	BitcoinNG	ByzCoin	RTChain
Liveness	√	√	×	√
Persistence	×	×	√	√
FlashAttacks	×	×	×	√
SelfishMiningAttacks	×	×	×	√
EclipseAttacks	×	×	√	√
DoubleSpendingAttacks	×	×	√	√
RecordPastTransactionBehavior	×	×	×	√

6 SECURITY ANALYSIS

In this section, we performed a security analysis of the scheme. We analyzed the aspects shown in Table 2 and compared them with existing systems [12, 22, 29].

6.1 Reputation-Based Consensus Mechanism

The reputation-based consensus mechanism is different from the PoW mechanism used in the Bitcoin system. In the consensus mechanism of PoW, the adversary can attack the system by controlling a majority of the hashing power. However, in the system of this article, the leader is randomly selected, and the result depends on the random algorithm and is unpredictable. The verifiers of the blocks in the system (i.e., members of the consensus group) are selected according to their reputation scores. In each round, $|G|$ users with the highest reputation are selected as members of the consensus group to participate in the consensus.

If the adversary is selected as the leader, the maliciously generated block may not be reached by the verifier, and in addition, the reputation score of the adversary will be reduced. If an attacker wants to enter the consensus group, it will have to spend a lot of time to work honestly in the system, and not directly attack the system through its computing power. If the adversary becomes (or controls) a member of the consensus group, he or she still cannot control the operation of the consensus mechanism, because the members of the consensus group adopt the PBFT consensus mechanism. Only when the adversary controls more than one-third of the members of the consensus group ($\frac{|G|}{3}$) at the same time can the security of the consensus mechanism be destroyed.

In addition to this, our system also satisfies the following four properties:

Persistence: If a node declares a certain transaction as confirmed, the remaining nodes will either report this transaction in the same position on the blockchain or will not report any other transaction in conflict to this transaction. The proposed algorithm solves the consistency problem of blockchain systems such as Bitcoin and provides a deterministic transaction guarantee. Transactions in the network are added by the leader candidate to their published pre-blocks ($Block_{r,i}$ in Figure 2), which pass through the consensus group and ultimately select the final block (or empty block) for the current round. That is to say, once a transaction appears in the final block of consensus, then the transaction is determined, and each node in the system will add this block to the blockchain.

Liveness: All honest nodes that participate in the consensus can finally reach a consensus result. In the basic PBFT protocol, it needs to wait for more than one-third of the honest nodes to vote, but the system may have a transmission delay, causing it to lose its liveness. The following definition of liveness is given in [34].

Definition 1. A consensus algorithm P is live if and only if for every honest validator v and finite time t:

(1) there exists some time $t' > t$ where v will have finalized a correct block b, and
(2) there is some time $t'' > t$ where every other honest validator will have also finalized b.

In this article, the consensus group members who participated in the voting are the most active and credible nodes in the system, and in the reputation-based consensus mechanism, a consensus can be reached when the node participating in the consensus receives a total reputation of more than two-thirds of the consensus group's reputation.

Compatibility: The greater the contribution a node makes to the system, the more reward it will receive. In the reward system proposed in Section 3, we divide rewards into reputation score and transaction fees. The transaction fee obtained by a node is determined by a weighted random sampling function, and the higher the reputation score, the higher the probability of the node obtaining the transaction fees. That is to say, in our system, the honest behavior of the node will lead to the improvement of its reputation score, thus obtaining more rewards. Specifically, we define the probability that the consensus group members behave correctly as $\Pr_h(G)$, the probability that the consensus group members are dishonest and detected by the system is $\Pr_d(G)$, and the accuracy of the system to detect malicious operations is $\Pr_v(G)$. So we have

$$\Pr_d(G) = \Pr_v(G) \cdot (1 - \Pr_h(G)). \tag{6.1}$$

Further, we have the following theorem:

Theorem 1. *A consensus group member i is honest with a probability of at least ε if $m(1 - \Pr_d(G)) \cdot (1 - p_i) < k \cdot \Pr_d(G) \cdot p_i$, where $\Pr_d(G) = \Pr_v(G) \cdot (1 - \varepsilon)$.*

Proof. According to the probability of honest behavior of the node, we define the benefit of the member participating in consensus as

$$Reward(\Pr_h(G)) = (1 - \Pr_d(G)) \cdot \frac{1}{k}(1 - p_i) - \Pr_d(G) \cdot \frac{1}{m} p_i$$

if $m(1 - \Pr_d(G)) \cdot (1 - p_i) < k \cdot \Pr_d(G) \cdot p_i$, where $\Pr_d(G) = \Pr_v(G) \cdot (1 - \varepsilon)$; then $Reward(\varepsilon) = (1 - \Pr_v(G) \cdot (1 - \varepsilon)) \cdot \frac{1}{k}(1 - p_i) - \Pr_v(G) \cdot (1 - \varepsilon) \cdot \frac{1}{m} p_i < 0$. We need to prove that for any node in which $\Pr_h(G) < \varepsilon$, $Reward(\Pr_h(G))$ is smaller than $Reward(\varepsilon)$. We have $Reward(\Pr_h(G)) = (1 - \Pr_v(G) \cdot (1 - \Pr_h(G))) \cdot \frac{1}{k}(1 - p_i) - \Pr_v(G) \cdot (1 - \Pr_h(G)) \cdot \frac{1}{m} p_i$, for $\Pr_h(G)$ is smaller than ε; thus, $Reward(\Pr_h(G)) < Reward(\varepsilon)$ holds. Consequently, if $m(1 - \Pr_d(G)) \cdot (1 - p_i) < k \cdot \Pr_d(G) \cdot p_i$, where $\Pr_d(G) = \Pr_v(G) \cdot (1 - \varepsilon)$, a consensus group member i is honest with a probability of at least ε. □

Fault Tolerance: A blockchain is a distributed, decentralized system that maintains a shared state. The role of the consensus mechanism is to enable the network to reach a consensus on this state, but sometimes this consensus may not be achieved. Therefore, fault tolerance is an important part of blockchain technology.

The traditional PBFT mechanism requires the total number of nodes $n \geqslant 3f + 1$ (where f represents the number of malicious nodes); that is, the system's failure nodes must not exceed one-third of the nodes of the whole network. In RTChain, our consensus mechanism allows no more than one-third of the nodes to become faulty, or the total reputation of the failure nodes is less than one-third of the consensus group's total reputation. Therefore:

Definition 2. The system is safe when

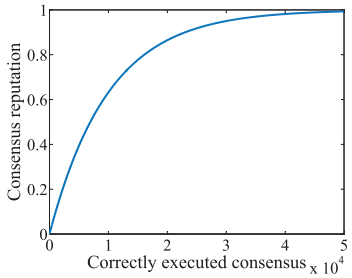

Fig. 6. The trend of consensus reputation changing with the number of times the consensus is correctly executed.

(1) the number of members of the consensus group is at least $3f' + 1$, where f' is the number of failure nodes in the consensus group, and
(2) the total reputation of the failure nodes is less than one-third of the consensus group's total reputation.

For the first condition, we assume that the number of failure nodes in the consensus group is f', and we have already demonstrated the compatibility of the system above; that is, the more legitimate the user's behavior, the higher his or her reputation score. Therefore, in a consensus group, we can get

$$\frac{|G|}{n} \geq \frac{f'}{f}. \tag{6.2}$$

That is to say, when there are f' invalid nodes in the consensus group, the number of members of the consensus group is at least $3f' + 1$.

For the second condition, we set the number of nodes to 100,000, and the members of the consensus group accounted for one-thousandth of the total number of nodes to simulate the reputation change of the nodes. Figure 6 shows how a node's consensus reputation changes with the number of times it correctly executes the consensus mechanism. It can be seen that when the nodes correctly execute the consensus, the consensus reputation grows more and more slowly. When the node is selected as a member of the consensus group and the consensus is correctly executed 10,000 times, its consensus reputation can reach about 0.5. Therefore, when the adversary wants to increase the voting weight, he or she will pay more to increase his or her reputation score, because it is difficult to obtain a high reputation score in a short time.

6.2 Security under Flash Attacks

In flash attacks [7], an attacker might purchase mining power (perhaps at a cost premium) for a short duration via bribery. This type of attack is feasible in networks such as Bitcoin. But in our system, the computing power is not available to the attacker to control the system, because each node in the system has the right to participate in the consensus through its proper system behavior. This allows our system to not only save computing power and reduce the waste of computing resources but also defend against attacks by any adversary who owns or aggregates large-scale computing power. RTChain prompts nodes in the transaction blockchain network to properly complete transactions and consensus work, because the reputation of all nodes is public, which will affect their later transactions.

6.3 Security under Selfish Mining Attacks

Selfish mining attacks [13] allow a pool of sufficient size to obtain a revenue larger than its ratio of mining power. It divides the miners into two categories: a colluding minority pool that follows the selfish mining strategy and a majority that follows the honest mining strategy. The selfish mining pool privatizes the blocks it mines, secretly bifurcating the blockchain and creating a private branch, and honest miners continue to mine in the shorter public branch. Therefore, selfish mining judiciously reveals blocks from the private branch to the public, such that the honest miners will continue to mine on the longer blocks recently revealed, and their previous efforts on the shorter public branch have been wasted. That is to say, in selfish mining attacks, selfish miners make honest miners' work ineffective by selectively exposing the blocks they are mining. Selfish mining is an attack against Bitcoin mining and incentive mechanism. Its purpose is not to destroy the operation mechanism of Bitcoin, but to obtain additional rewards and make honest miners perform invalid calculations. At present, many efforts have been carried out to optimize and expand the strategy of selfish mining [5, 15, 30, 33]. In RTChain, after each block is created by the leader of the round, it needs to be submitted to the consensus group. After the consensus team members reach a consensus, they can add a certificate *CERT* to the block and then publish it to the blockchain. Each block is created based on the last block that has been added to the blockchain. The attacker cannot predict the next round of leaders, consensus group members, and consensus results, so it is impossible to attack the blockchain system through selfish mining.

6.4 Security under Eclipse Attacks

An Eclipse attack [17] is when an attacker "isolated" a victim node from the normal blockchain network. When a node is attacked by Eclipse, most of the external contacts of the node are controlled by the malicious node, and the malicious node can further implement attacks such as route spoofing, denial of service, and ID hijacking. Therefore, the Eclipse attack poses a serious threat to the blockchain network. In RTChain, the choice of leader is randomly selected by a verifiable random function in cryptography, and the attacker does not know which of the leader candidates is the true leader in this round. This makes it almost impossible for an attacker to accurately isolate the real leader, and the cost of the attack is very large. On the other hand, eclipse attacks may indeed reduce consensus efficiency and system throughput by isolating consensus group members, but the generation and verification of blocks in the system are separate, so the attacker still cannot initiate attacks such as double payment.

6.5 Security under Double Spending Attacks

Double spending attacks [19, 32] refer to trading one digital asset twice or even multiple times. That is to say, when a transaction has been issued and has passed through n blocks, the attacker regenerates a new blockchain in a very short time, making the new chain longer than the previous blockchain. This allows the attacker to retrieve the spent currency from the previous transaction and use it for a secondary transaction, because in a blockchain system like Bitcoin, the system automatically recognizes the longest chain as a valid chain.

RTChain provides strong consistency guarantees. A transaction is released into the block after it is generated, and once it is added to the chain, it is deterministic and irreversible. No matter how powerful the attacker's computing power is, it will not enable our blockchain system to produce longer branches to complete the double spending attack.

7 EVALUATION

For our prototype, we built an experimental network. We implemented our protocol and we deploy the nodes on client machines with Intel i7-4600U 2.70GHz CPU, and 16GB RAM. In the experiment

Table 3. Comparison of Different Blockchain Schemes

System	Block Interval	Confirmation Time
Bitcoin	10 min	60 min
Litecoin	2.5 min	70 min
Dogecoin	1 min	50 min
Ethereum	0.4 min	15 min
Ourscheme	0.26 min	0.26 min

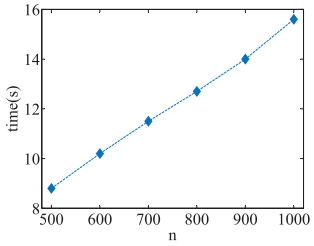

Fig. 7. The consensus time.

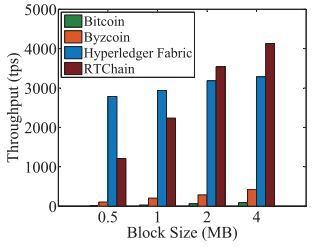

Fig. 8. Throughput.

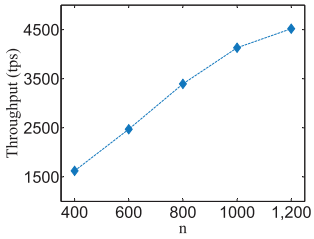

Fig. 9. Throughput vs. number of nodes.

we set up 1,000 nodes; each node has a bandwidth of 20 Mbps, and the block size is 2MB. According to the number of nodes, we set the initial transaction reputation score and consensus reputation score of the node to $\frac{1}{n}$, which is 0.001. We set the values of α and β to 0.5; that is to say, in our experiment, the transaction reputation and consensus reputation have the same weight.

Due to the large span of the WAN, in order to make our experimental results closer to the WAN, we have increased the round-trip network latency of 200 ms between every two nodes. Although the network round-trip delay is unlikely to be constant in WAN, adding an average network delay can also help the following evaluation be closer to the actual operation.

Transaction Confirmation Time. Table 3 is a comparison of our scheme with the four most widely used blockchain systems in terms of block interval and transaction confirmation time. Experiments show that the Bitcoin system takes 10 minutes to generate a block, but because of the possible fork, the system needs to wait for the generation of six blocks to reduce the possibility of invalid transactions. Although the block interval of Bitcoin is artificially set to 10 minutes, in order to balance the security and efficiency in the blockchain network, it is the best choice after trade-off. Similarly, Litecoin takes 2.5 minutes to generate a block, and it takes 70 minutes for a transaction to be confirmed; Dogecoin takes 1 minute and 50 minutes, respectively; Ethereum takes 24 seconds and 15 minutes, respectively. In our scheme, once the block is generated, the transactions in the block will be confirmed; the block interval and transaction confirmation time is 0.26 minute. Therefore, our blockchain system using a reputation-based consensus mechanism represents a significant performance advantage over these existing blockchain systems.

Consensus Time. Figure 7 shows the consensus time of our scheme. Let n be the number of the nodes in the system, and we vary n from 500 to 1,000. We have observed that it takes about 8.8 seconds to reach a consensus with 500 participants in the system. When the number of nodes is increased to 1,000, the time cost of RTChain is 15.6 seconds.

Throughput. In terms of throughput, Figure 8 compares RTChain with Bitcoin, ByzCoin, and Hyperledger Fabric. We set the block sizes of these systems to 0.5MB, 1MB, 2MB, and 4MB. From

the figure we can see that the throughput of the Bitcoin system is 2TPS, 3TPS, 7TPS, and 14TPS under different block sizes; the ByzCoin system [22] is 105TPS, 205TPS, 287TPS, and 428TPS, respectively. Hyperledger Fabric [3] is a permissioned blockchain system that is advantageous in terms of throughput. Under the above four block sizes, its throughput is approximately 2,785TPS, 2,940TPS, 3,185TPS, and 3,285TPS, respectively. In RTChain, when the block size is 0.5MB, the throughput is about 1,208TPS, and when the block size is increased to 4MB, the system throughput can reach 4,131TPS. Figure 9 shows the relationship between the number of nodes and throughput. We set the block size to 4MB, and experiments show that when the number of nodes increases from 400 to 1,200, the throughput increases from 1,623TPS to 4,519TPS. The scale of the system increases with the number of nodes.

From the preceding experiment results, we can see that our protocol is effective and its efficiency is acceptable in transaction blockchain. In addition, the efficiency of the solution will be greatly improved after optimizing our code in future work.

8 CONCLUSION AND FUTURE WORK

In this article, we have proposed a reputation system with transaction and consensus incentives, called RTChain, for e-commerce blockchain. In RTChain, both the consensus behavior and transaction behavior of a node will affect its reputation score with an incentive mechanism. We have improved the consensus mechanism to satisfy the high throughput requirement of e-commerce. Moreover, RTChain is secure against most of the existing attacks such as flash attacks, self-mining attacks, eclipse attacks, and double spending attacks and allows nodes that participate in the consensus to fail no more than one-third of the total reputation. Finally, we have built a prototype of RTChain, and the experimental result shows that RTChain is promising and deployable for e-commerce blockchains.

This article has shown how to use the reward system to encourage the legitimate behavior of users. However, this is just the beginning of the reward system research, and there are still many avenues of research left to pursue in this area, including the number of awards and the mode of payment, how awards change with the operation of the system, and the detailed description of smart contracts, which are left to our future work.

REFERENCES

[1] Bitshare. [n.d.]. *Delegated Proof-of-Stake Consensus*. Retrieved from https://bitshares.org/technology/delegated-proof-of-stake-consensus/.

[2] Mustafa Al-Bassam, Alberto Sonnino, Shehar Bano, Dave Hrycyszyn, and George Danezis. 2018. Chainspace: A sharded smart contracts platform. In *25th Annual Network and Distributed System Security Symposium (NDSS'18)*. The Internet Society. Retrieved from http://wp.internetsociety.org/ndss/wp-content/uploads/sites/25/2018/02/ndss2018_09-2_Al-Bassam_paper.pdf.

[3] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, Srinivasan Muralidharan, Chet Murthy, Binh Nguyen, Manish Sethi, Gari Singh, Keith Smith, Alessandro Sorniotti, Chrysoula Stathakopoulou, Marko Vulou, Sharon Weed Cocco, and Jason Yellick. 2018. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference, EuroSys 2018, Porto, Portugal, April 23-26, 2018*. 30:1–30:15. https://doi.org/10.1145/3190508

[4] Pierre-Louis Aublin, Rachid Guerraoui, Nikola Knezevic, Vivien Quéma, and Marko Vukolic. 2015. The next 700 BFT protocols. *ACM Trans. Comput. Syst.* 32, 4 (2015), 12:1–12:45. DOI: https://doi.org/10.1145/2658994

[5] Juan Beccuti and Christian Jaag. 2017. *The Bitcoin Mining Game: On the Optimality of Honesty in Proof-of-work Consensus Mechanism*. Working Papers 0060. Swiss Economics. Retrieved from https://ideas.repec.org/p/chc/wpaper/0060.html.

[6] Iddo Bentov, Rafael Pass, and Elaine Shi. 2016. Snow white: Provably secure proofs of stake. *IACR Cryptology ePrint Archive* 2016 (2016), 919. http://eprint.iacr.org/2016/919

[7] Joseph Bonneau. 2016. Why buy when you can rent? - Bribery attacks on bitcoin-style consensus. In *Financial Cryptography and Data Security (FC'16) International Workshops, BITCOIN, VOTING, and WAHC, Revised Selected Papers*. 19–26. DOI: https://doi.org/10.1007/978-3-662-53357-4_2

[8] Davide Carboni. 2015. Feedback based reputation on top of the bitcoin blockchain. *CoRR* abs/1502.01504 (2015). arxiv:1502.01504 http://arxiv.org/abs/1502.01504

[9] Jing Chen and Silvio Micali. 2019. Algorand: A secure and efficient distributed ledger. *Theor. Comput. Sci.* 777 (2019),155–183. https://doi.org/10.1016/j.tcs.2019.02.001

[10] Nxt Community. 2014. *Nxt Whitepaper*.

[11] Cynthia Dwork and Moni Naor. 1992. Pricing via processing or combatting junk mail. In *Advances in Cryptology (CRYPTO'92), 12th Annual International Cryptology Conference, Proceedings (Lecture Notes in Computer Science)*, Ernest F. Brickell (Ed.), Vol. 740. Springer, 139–147. DOI: https://doi.org/10.1007/3-540-48071-4_10

[12] Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert Van Renesse. 2016. Bitcoin-NG: A scalable blockchain protocol. In *Usenix Conference on Networked Systems Design & Implementation*.

[13] Ittay Eyal and Emin Gün Sirer. 2018. Majority is not enough: Bitcoin mining is vulnerable. *Commun. ACM* 61, 7 (2018), 95–102. DOI: https://doi.org/10.1145/3212998

[14] Fangyu Gai, Baosheng Wang, Wenping Deng, and Wei Peng. 2018. Proof of reputation: A reputation-based consensus protocol for peer-to-peer network. In *Database Systems for Advanced Applications - 23rd International Conference (DASFAA'18), Proceedings, Part II*. 666–681. DOI: https://doi.org/10.1007/978-3-319-91458-9_41

[15] J. Göbel, H. P. Keeler, A. E. Krzesinski, and P. G. Taylor. 2016. Bitcoin blockchain dynamics: The selfish-mine strategy in the presence of propagation delay. *Perform. Eval.* 104 (2016), 23–41.

[16] Guy Golan-Gueta, Ittai Abraham, Shelly Grossman, Dahlia Malkhi, Benny Pinkas, Michael K. Reiter, Dragos-Adrian Seredinschi, Orr Tamir, and Alin Tomescu. 2018. SBFT: A scalable decentralized trust infrastructure for blockchains. *CoRR* abs/1804.01626 (2018). arxiv:1804.01626 http://arxiv.org/abs/1804.01626

[17] Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. 2015. Eclipse attacks on bitcoin's peer-to-peer network. In *24th USENIX Security Symposium (USENIX Security'15)*.129–144. Retrieved from https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/heilman.

[18] Markus Jakobsson and Ari Juels. 1999. Proofs of work and bread pudding protocols (extended abstract). In *Ifip Tc6/tc11 Joint Working Conference on Secure Information Networks: Communications & Multimedia Security*.

[19] Ghassan Karame, Elli Androulaki, and Srdjan Capkun. 2012. Double-spending fast payments in bitcoin. In *the ACM Conference on Computer and Communications Security (CCS'12)*. 906–917. DOI: https://doi.org/10.1145/2382196.2382292

[20] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. 2017. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Advances in Cryptology (CRYPTO'17) - 37th Annual International Cryptology Conference, Proceedings, Part I*. 357–388. DOI: https://doi.org/10.1007/978-3-319-63688-7_12

[21] Aggelos Kiayias, Alexander Russell, Bernardo David, Roman Oliynykov, Iddo Bentov, Charles Lee, Alex Mizrahi, et al. 2017. PPCoin: Peer-to-peer crypto-currency with proof-of-stake. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS'16)*. 1–27. Retrieved from http://peerco.in/assets/paper/peercoin-paper.pdf%0A.

[22] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Nicolas Gailly, Ismail Khoffi, and Bryan Ford. 2016. Enhancing bitcoin security and performance with strong consistency via collective signing. *Appl. Math. Modell.* 37, 8 (2016), 5723–5742.

[23] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ewa Syta, and Bryan Ford. 2018. OmniLedger: A secure, scale-out, decentralized ledger via sharding. In *2018 IEEE Symposium on Security and Privacy (SP'18), Proceedings*. IEEE Computer Society, 583–598. DOI: https://doi.org/10.1109/SP.2018.000-5

[24] Leslie Lamport. 1998. The part-time parliament. *ACM Trans. Comput. Syst.* 16, 2 (1998), 133–169. DOI: https://doi.org/10.1145/279227.279229

[25] Leslie Lamport, Robert Shostak, and Marshall Pease. 1982. *The Byzantine Generals Problem*. 382–401 pages.

[26] Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, and Prateek Saxena. 2016. A secure sharding protocol for open blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi (Eds.). ACM, 17–30. DOI: https://doi.org/10.1145/2976749.2978389

[27] Oom Temudo De Castro Miguel. 2002. Practical byzantine fault tolerance. *ACM Trans. Comput. Syst.* 20, 4 (2002), 398–461.

[28] Andrew Miller, Xia Yu, Kyle Croman, Elaine Shi, and Dawn Song. 2016. The honey badger of BFT protocols. In *ACM SIGSAC Conference on Computer & Communications Security*.

[29] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. *Consulted* (2008).

[30] Kartik Nayak, Srijan Kumar, Andrew Miller, and Elaine Shi. 2016. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In *IEEE European Symposium on Security and Privacy (EuroS&P'16)*. IEEE, 305–320. DOI: https://doi.org/10.1109/EuroSP.2016.32

[31] Diego Ongaro and John K. Ousterhout. 2014. In search of an understandable consensus algorithm. In *2014 USENIX Annual Technical Conference (USENIX ATC'14)*. 305–319. https://www.usenix.org/conference/atc14/technical-sessions/presentation/ongaro.

[32] Meni Rosenfeld. 2014. Analysis of hashrate-based double spending. *Eprint Arxiv* (2014).

[33] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. 2016. Optimal selfish mining strategies in bitcoin. In *Financial Cryptography and Data Security - 20th International Conference (FC'16), Revised Selected Papers*. 515–532. DOI: https://doi.org/10.1007/978-3-662-54970-4_30

[34] Derek Sorensen. 2019. Establishing standards for consensus on blockchains. In *Blockchain (ICBC'19) - 2nd International Conference, Held as Part of the Services Conference Federation (SCF'19), Proceedings*. 18–33. DOI: https://doi.org/10.1007/978-3-030-23404-1_2

[35] Pavel Vasin. 2017. *BlackCoins Proof-of-Stake Protocol v2*. https://www.blackcoin.co.

[36] Maofan Yin, Dahlia Malkhi, Michael K. Reiter, Guy Golan-Gueta, and Ittai Abraham. 2019. HotStuff: BFT consensus with linearity and responsiveness. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing (PODC'19)*, Peter Robinson and Faith Ellen (Eds.). ACM, 347–356. DOI: https://doi.org/10.1145/3293611.3331591

[37] Jiangshan Yu, David Kozhaya, Jeremie Decouchant, and Paulo Jorge Esteves Veríssimo. 2019. RepuCoin: Your reputation is your power. *IEEE Trans. Comput.* 68, 8 (2019), 1225–1237. DOI: https://doi.org/10.1109/TC.2019.2900648

[38] Mahdi Zamani, Mahnush Movahedi, and Mariana Raykova. 2018. RapidChain: Scaling blockchain via full sharding. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS'18)*, David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang (Eds.). ACM, 931–948. DOI: https://doi.org/10.1145/3243734.3243853

Received December 2019; revised September 2020; accepted October 2020