T-PBFT: An EigenTrust-Based Practical Byzantine Fault Tolerance Consensus Algorithm

Sheng Gao[1,2,*], Tianyu Yu[1], Jianming Zhu[1], Wei Cai[3]

[1] School of Information, Central University of Finance of Economics, Beijing 100081, China
[2] Grain Information Processing and Control Key Laboratory of Ministry of Education, Henan University of Technology, Zhengzhou 450001, China
[3] School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen 518172, China
* The corresponding author, email: sgao@cufe.edu.cn

Abstract: Blockchain with these characteristics of decentralized structure, transparent and credible, time-series and immutability, has been considering as a promising technology. Consensus algorithm as one of the core techniques of blockchain directly affects the scalability of blockchain systems. Existing probabilistic finality blockchain consensus algorithms such as PoW, PoS, suffer from power consumptions and low efficiency; while absolute finality blockchain consensus algorithms such as PBFT, HoneyBadgerBFT, could not meet the scalability requirement in a large-scale network. In this paper, we propose a novel optimized practical Byzantine fault tolerance consensus algorithm based on EigenTrust model, namely T-PBFT, which is a multi-stage consensus algorithm. It evaluates node trust by the transactions between nodes so that the high quality of nodes in the network will be selected to construct a consensus group. To reduce the probability of view change, we propose to replace a single primary node with a primary group. By group signature and mutual supervision, we can enhance the robustness of the primary group further. Finally, we analyze T-PBFT and compare it with the other Byzantine fault tolerant consensus algorithms. Theoretical analysis shows that our T-PBFT can optimize the Byzantine fault-tolerant rate, reduce the probability of view change and communication complexity.

Keywords: blockchain; consensus protocol; Byzantine fault tolerance; trust model

I. INTRODUCTION

Blockchain as the underlying technology of Bitcoin [1] is essentially a distributed and append-only ledger, which are shared and maintained among distrustful nodes [2]. In a nutshell, it integrates with multiple technologies such as distributed ledger, cryptography, consensus protocol, and smart contract to achieve the characteristics of transparency, credibility, reliability and immutability [3]. Typically, the deployment types of blockchains can be roughly categorized into public blockchains, consortium blockchains and private blockchains, where consortium blockchains and private blockchains are collectively called permissioned blockchains [4],[5]. For public blockchains, any node with Internet access can join or exit the process of mining, access and submit transactions. There are no restrictions on read, write and commit operations (eg. Bitcoin [1], Ethereum [6]). For private blockchains, the read and write privileges are fully controlled by one node, which is a centralized system (eg. multichain [7]). Differ-

Received: Nov. 14, 2018
Revised: Apr. 11, 2019
Editor: Yueming Cai

ently, consortium blockchains are considered to be a multi-centralized system, where the read and write privileges are determined by a predefined list of nodes, and any node that wants to join the consensus process, access and submit the transactions must be authenticated by them (eg. Hyperledger [8]). Nowadays, the application scenarios of blockchains have been spread from cryptocurrency [9] and digital asset [10] to non-financial applications, such as energy [11], healthcare [12], education [13], and supply chain [14].

In order to reduce the probability of view change and improve the Byzantine fault-tolerant rate, a novel optimized practical Byzantine fault tolerance consensus algorithm based on EigenTrust named T-PBFT is proposed in this paper.

Apparently, consensus protocol is a core component of blockchains, which potentially affects the efficiency and scalability [15]. Specifically, a blockchain consensus protocol denotes how to make mutually distrusting nodes agree on a new block periodically to be added to the blockchain, which should meet the basic properties [16],[17]:(a) *consistency*: all the normal nodes should agree on a same block;(b) *validity*: a decided block should be proposed by a consensus node;(c) *liveness* (also named *termination*): every normal node should eventually decide some block. The *consistency* and *validity* properties define the safety property of the blockchain consensus. According to the different deployment types of blockchains, existing blockchain consensus algorithms can roughly be divided into *Proof-of-X* (PoX) consensus algorithms and *Byzantine Fault Tolerant* (BFT) consensus algorithms [18]. The PoX consensus algorithms such as PoW [1], PoS [19], are appropriate for public blockchains, which suffer from low efficiency and high computational power. The BFT consensus algorithms such as Practical BFT (PBFT)[20], Scalable BFT [21], Zyzzyva [22], HoneyBadgerBFT [23], are usually used in consortium blockchains. However, most of exiting BFT-type consensus algorithms are either with low scalability, e.g. the performance of PBFT consensus algorithm will decline sharply with the increase of the number of nodes, or with a low Byzantine fault-tolerant rate. In addition, the failure of primary node would incur the view change process and affect the whole consensus process.

In this paper, we propose a new Byzantine fault-tolerant consensus, named T-PBFT, which narrows the consensus consortium nodes down to a group of nodes with higher trust values. To get the appropriate consensus group, Biryukov et al.[24] proposed a reputation module Guru to select the committee based on the outcomes of consensus rounds. However, they do not consider individual ratings. Thus, we introduce the EigenTrust [25],[26] model that takes transaction relationship and ratings into account to evaluate the trust value of each node. After that we replace the primary node with a constructed primary group to reduce the probability of view change process, and use group signature and mutual supervision to enhance the robustness of T-PBFT further. Finally, we analyze our T-PBFT in details and compare it with the other related BFT-type consensus algorithms.

Overall, the main contributions of this paper are summarized as follows.

- We propose to use EigenTrust [25], [26] to build a trustworthy consensus group with higher trust values. It can prevent the nodes with lower trust values from participating in consensus, narrow down the number of the consensus consortium nodes and improve the consensus efficiency.
- We propose a new framework of our multi-stage consensus algorithm T-PBFT, and detail the stages of T-PBFT which contain node trust evaluation, consensus group building and consensus process.
- We compare T-PBFT with other related BFT-type algorithms and make a detailed analysis from the aspects of communication complexity, Byzantine fault tolerance and view change probability theoretically.

The remainder of this paper is organized as follows. Section II introduces related works. Section III introduces the preliminaries of T-PBFT. Section IV presents the system framework of T-PBFT and Section V describes the design process of T-PBFT. Section VI compares T-PBFT with related BFT-type algorithms. Section VII concludes this paper.

II. Related Work

Blockchain consensus refers to the mutual distrust nodes come to an agreement on a new block that will be appended to blockchain in a distributed environment, which has been received extensive attentions [2],[15],[16]. Exiting consensus algorithms can be classified into different types on the basis of different principles [17],[27], such as the deployment types of blockchains, consensus leader election. In this paper, according to the efficiency of block confirmation, we recognized existing work as probabilistic finality blockchain consensus and absolute finality blockchain consensus.

2.1 Probabilistic finality blockchain consensus

Probabilistic finality blockchain consensus refers that any block in a blockchain would be reverted at a certain probability. When the probability is close to 0, the block in blockchain is determined. The most typical probabilistic finally blockchain consensus algorithm is Bitcoin's Nakamoto consensus named proof-of-work (PoW)[1], which stimulates nodes to compete with the right of bookkeeping by solving a cryptographic hash puzzle with a given different target. Specifically, the puzzle is $\mathcal{H}(blockhead) \leq Target$, where $\mathcal{H}$ is double SHA-256 in Bitcoin and *Target* is the difficulty target. Every consensus node changes the nonce in blockhead from 0 to infinity. Any node that makes this inequality true would broadcast this block to others to verify its effectiveness for getting reward. Once approved, the block would be append to the tail of all the nodes' blockchains. To ensure a generated block being reverted with a very low likelihood, it recommends to wait until at least 6 blocks have already been confirmed [9]. Based on the existing PoW-based Nakamoto consensus, there are lots of variations or extensions [15],[16]. For example, Ball et al.[28] proposed a *Proof of Useful Work* scheme, which utilizes the hashrate in PoW to solve those meaningful problems. Park et al.[29] proposed *Proof of Space*, which decides the bookkeeping node by disk space. Any node that devotes more disk space would mine a block at a high probability. To alleviate the cost overhead of computing and storage resources, while improving the consensus efficiency, King and Nadal [19] proposed a alternative consensus named *Proof of Stake* (PoS), which determines the right of bookkeeping based on the stake that each node owns. Any node with a larger stake at a higher probability to generate a new block, where the stake can be defined as Coin age, which is currency amount times holding period. To discourage hoarding and facilitate spending, Ren et al.[30] proposed *Proof of Stake Velocity* by changing the form of Coin age in PoS with an exponential decay function. Besides, Some blockchain algorithms mix PoW with others to improve the blockchain efficiency, such as *Proof of Activity* [31], 2-hop blockchain [32] and Bitcoin-NG [33].

2.2 Absolute finality blockchain consensus

Absolute finality blockchain consensus is regarded that any transaction is immediately determined once it is included in a block and appended to a blockchain. That is, a new block generated by a leader node should be committed by sufficient nodes before submitting to the blockchain. Currently, this type of blockchain consensus is mainly achieved by BFT-based protocols, which has received widely attentions in consortium blockchain [5],[17]. Traditional distributed consistency consensus algorithms, such as VR [34], Raft [35], Paxos [36], assume that all the consensus nodes are not Byzantine nodes. They are crash fault-tolerant consensus algorithms, which can tolerate a percentage of fault nodes caused by system crash or network anomaly. Castro and Liskov [20] first proposed PBFT, which tolerates the proportion of Byzantine nodes in all the consensus nodes less than 1/3. PBFT is a three-stage protocol, which has been adopted in Hyperledger Fabric v0.6 [37] and replaced by Raft and Kafka in Hyperledger Fabric v1.4 [38]. We would detailed the consensus process of PBFT in Section III. However, the efficiency of PBFT sharply drops with the expansion of network size [39]. Hereafter, there have been lots of work to improve PBFT. For example, Kotla et al.[22] proposed Zyzzyva to simplify PBFT by adopting the primary requests and the client responds immediately. Although Zyzzyva reduces the communication complexity, it would become dependent on the client. Tendermint [40] is the first PBFT-based PoS blockchain consensus, which includes: (1) *Propose*: it selects a proposer from validators to generate a new proposal block. (2) *Pre-vote*: all the validators verify the effectiveness of the proposal block. If true, then a validator would submit a pre-vote. Once a validator gets pre-votes more than 2/3, then it would submit a pre-commit. (3) *Pre-commit*: If a validator receives pre-commits more than 2/3, the new proposal block is committed to blockchain. Compared with PBFT, Tendermint simplifies the phase of view change and stores all the information in blockchain. Miller et al.[23] proposed the first practical asynchronous BFT protocol, namely HoneyBadgerBFT, to solve the time bound caused by network anomaly. However, it causes a large communication complexity. Besides, the other BFT blockchain consensus algorithms such as SBFT [41], dBFT [42], Thunderella [43], Algorand [44], and OuroborosBFT [45], are also proposed to improve PBFT. However, these BFT blockchain consensus algorithms are either with low efficiency and scalability, or strongly dependent on the primariy node in consortium blockchains.

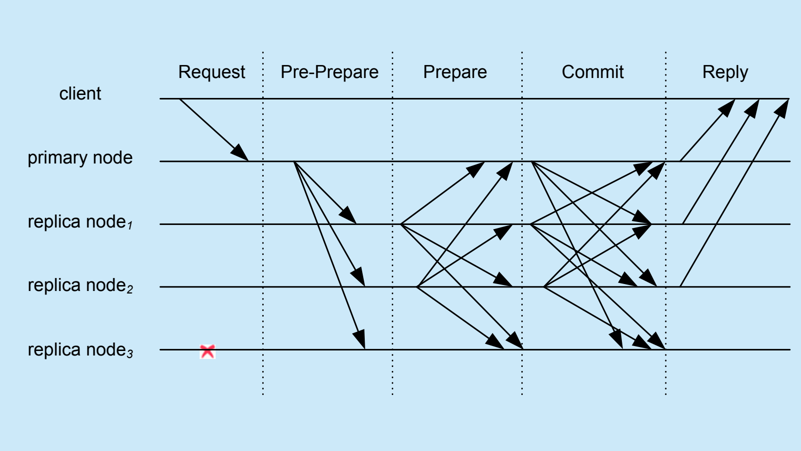

Fig. 1. *The process of practical byzantine fault tolerance [20].*

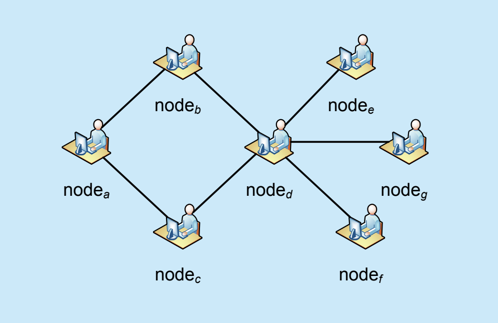

Fig. 2. *A relation graph between nodes.*

III. Preliminaries

In this section, we will introduce some foundations for our T-PBFT, including PBFT [20] and EigenTrust [25],[26].

3.1 Overview of PBFT

PBFT [20] is a state machine replica copy algorithm that can be applied to synchronous network environments. There are three important components in PBFT, namely view, primary, and replica. The primary node is the initiator of the voting process. The replica node is used to guarantee the effectiveness of the voting process. When the primary node fails, the view rotation function is called to change the current primary node.

Here, we brief review the three-stage process of PBFT including pre-prepare, prepare, and commit, depicted by figure 1, where replica node$_3$ is considered to be the Byzantine node. More details is available in [20].

1) In the pre-prepare phase, the client first sends a request to the primary node. The primary node checks and processes the client's request, and then broadcasts a pre-prepare message to the replica nodes. Each replica node receives and verifies the validity of the pre-prepare message. Once it is authenticated, a replica node will accept the request and start the prepare phase.

2) In the prepare phase, the replica nodes broadcast prepared messages to each other. Meanwhile, they also receive the prepared

messages from others, and then check its validity. When a replica node gets $2f$ valid prepared messages from different replica nodes, the prepare phase is finished, where f is the number of Byzantine nodes in the system.

3) In the commit phase, each node broadcasts a commit message to others for validation. Once the number of the commit messages including itself that are received by a node in the prepare state is equal or greater than $2f+1$, it will send a reply message to the client. When the client has received $f+1$ of the same reply message, the consensus is achieved.

3.2 EigenTrust Model

The EigenTrust [25],[26] is one of the most authoritative trust models, which can be applied in the P2P environment. The EigenTrust model can effectively evaluate the trust value for every node while enhancing security by supervising the dishonest nodes. It obtains a unique global trust value for every node in the system by recording the transaction history between nodes. The global trust value T_i in our model can be computed by taking

$$T_i = C_{1i}T_1 + ... + C_{ni}T_n,$$

where T_i is the global trust value of $node_i$ and C_{ij} is the local trust value of $node_i$ to $node_j$.

Figure 2 shows the relationship between nodes. Nodes connected by lines mean that they have the direct transaction. We can easily divide the relationship between nodes into two types: the nodes with transactions and the nodes without transactions. Take $node_a$ as an example, there will be three types of trust values in EigenTrust model [25], [26]:

1) Direct trust value C_{ab}: it can be evaluated between $node_a$ and $node_b$ that conducts transactions directly. We define $S_{ab} = sat(node_a, node_b)\ unsat(node_a, node_b)$, where *sat* and *unsat* respectively represent the number of satisfactory and unsatisfactory transactions between $node_a$ and $node_b$. Then $C_{ab} = \frac{max(S_{ab},0)}{\sum_x max(S_{ax},0)}$, where x=b and c.

2) Recommended trust value C_{ad}: it can be evaluated between $node_a$ and $node_d$ that do not conduct any transaction. It is built on the basis of transitive trust and its value is related to the direct trust value. The C_{ad} can be taken by $C_{ad} = \sum_k C_{ak}C_{kd}$, where $k = b$ and c.

3) Global trust value T_a^{k+1}: it is the quantifiable degree of trust that the system evaluates for nodes. The global trust value of $node_a$ integrates the trust value of each nodes in the network and combines the current global trust value of each node, which can be used as an evaluation index for the trust degree of $node_a$.

IV. System Overview

4.1 T-PBFT framework

As a typical P2P network architecture application, blockchain has received extensive attention due to its anonymity and decentralization characteristics. However, due to its own characteristics, blockchain is vulnerable to be attacked by malicious nodes. How to reduce the impact of malicious nodes or dishonest nodes in the system has always been an important research content in the consensus field. Therefore, we propose a new multi-stage consensus algorithm based on the EigenTrust model named T-PBFT, and the overview framework shown by figure 3.

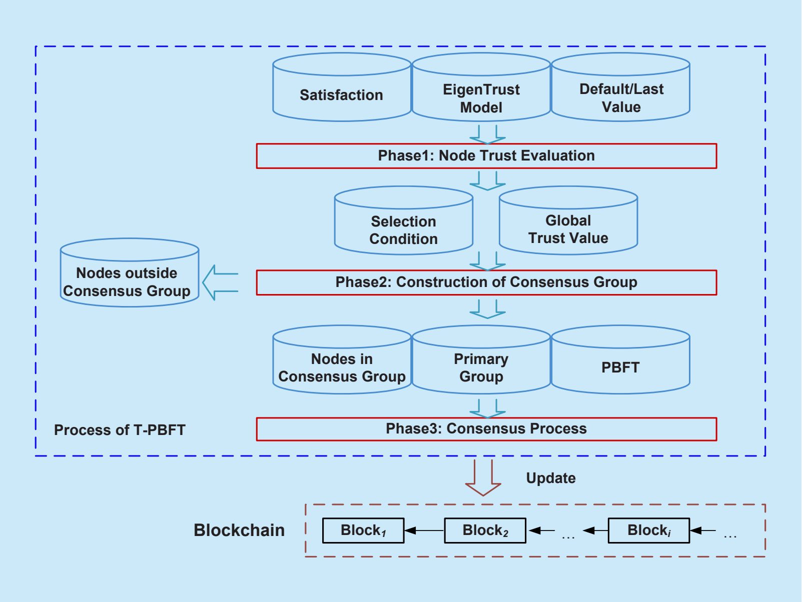

Fig. 3. *T-PBFT Framework.*

Our consensus algorithm T-PBFT includes three phases: node trust evaluation, construction of consensus group and consensus process, which would be described in Section V in detail. First, we will introduce the EigenTrust model to compute the global trust value for nodes which will be the basis to select consensus group. And then, the nodes with higher trust values will be selected into a consensus group. In the process of consensus, because of the consensus group, the number of nodes participating in the consensus will be decreased, which makes consensus process more efficient in a large-scale network environment. After that, there will be a new block connected to the blockchain and new transactions between nodes, so the global trust value can dynamically change with blocks and the T-PBFT can start a new round.

Algorithm 1. DivideNodes

```
Input: $node_i$, node set $N\ odes$
Output: $TxNodes$, $NonTxNodes$
$TxNodes \leftarrow \varnothing$, $NonTxNodes \leftarrow \varnothing$;
    for $node_j \in Nodes$ do
        if $node_j$ trades with $node_i$ then
            $TxNodes \leftarrow node_j$;
        else
            $NonTxNodes \leftarrow node_j$;
        end
    end
```

Algorithm 2. CalcTxNodeTrust.

```
Input: $node_i$, $TxNodes$ of $node_i$
Output: Direct trust value $C_{ij}$
$C_{ij} \leftarrow 0$;
for $node_j \in TxNodes$ do
    $S_{ij} = sat(i, j) - unsat(i, j)$;
    $S_{total} = \sum max(S_{ij}, 0)$
end
if $S_{total} = 0$ then
    set $C_{ij} = \frac{1}{N}$; ▷$N$ is the size of Nodes
else
    for $node_j \in TxNodes$ do
        $C_{ij} = \frac{max(S_{ij}, 0)}{S_{total}}$;
    end
end
```

4.2 Assumptions

a) Behavior consistency: In T-PBFT, we consider that those nodes with higher global trust values are more trustworthy under the personal profit-driven considering. That is, any node with a higher global trust value is reasonable that would behave honestly at a high probability.

b) Limited transaction time: Time play an important role in the EigenTrust model. To ensure the effectiveness of EigenTrust, the transaction set should be stable. Therefore, we take the amount of transactions in a limited transaction time into considerations, such as an hour, a day, or the other time slot.

V. THE PROCESS OF T-PBFT

5.1 Node trust evaluation

In this phase, we depict the process of node trust evaluation based on EigenTrust model [25], [26]. Support there are N nodes in the network and we initialize the global trust values of all nodes to $1/N$, and then compute the direct trust value between nodes with directly trading. For those nodes without transactions directly with each other, we compute the recommended trust value. Finally, the global trust can be computed based on these.

To compute the direct trust value and recommended trust value, Algorithm 1 classifies all the nodes according to the transaction relationships among nodes. For a given $node_i$, any node would be divided into the set named *TxNodes* if it conducts a transaction with $node_i$; otherwise it would be put into another set named *NonTxNodes*.

Now we present the computational processes of direct trust value and recommended trust value by Algorithm 2 and Algorithm 3 respectively. As depicted by Algorithm 2, it takes $node_i$ and its direct transaction set *TxNodes* as inputs, and then computes the absolute satisfaction value S_{ij} by querying the historical records including satisfied and dissatisfied transactions. Finally, we can get the direct trust value C_{ij} between $node_i$ and $node_j$.

Algorithm 3 describes the computational process of the recommended trust value. Following the instructions above, it take $node_i$, *TxNodes* of all the nodes, and those nodes in *NonTxNodes* that do not conduct transactions with $node_i$ as inputs. The main idea is to find the transaction paths and compute it synthetically based on these direct trust values. Specifically, for $node_j$ that do not have conducted any transaction with $node_i$, to compute the recommended trust value, $node_i$ needs to get $node_k \in TxNodes$ in which have conducted transactions with the target $node_j$. Then, the recommended trust value between $node_i$ and $node_j$ can be computed by the product of C_{ik} and C_{kj}. If none exists, we will iteratively compute the recommended trust value by the different transaction paths.

Afterwards, all nodes establish a local trust relationship and have an accurate local trust value. To obtain the trust value that can fully represent the trust level of the node, we need to compute the global trust value. The value of all nodes in the initial phase is $1/N$, where N is the number of nodes in the system. When a new block is generated, the node needs to reevaluate its global trust value. Algorithm 4 shows the computing method of the global trust value. For $node_i$, its global trust value should be the sum of the product of the local trust value and the other node's corresponding global trust value. We can see that the global trust value is dynamic, which is affected by all the other nodes. By using this comprehensive dynamic assessment method, we can get accurate node trusts, which can help to reduce some low credit nodes for consensus.

5.2 Construction of consensus group

In this phase, we present the process of blockchain consensus group construction. As we discuss in node trust evaluation, the global trust values of all the nodes in the system are quantificationally determined by EigenTrust model. Recall the behavior consistency we assumed, those nodes with higher global trust values are more trustworthy under the personal profitdriven considering. To improve the efficiency and scalability of consortium blockchain consensus algorithms, we can construct the blockchain consensus group by selecting some nodes with higher trust values instead of all the consortium blockchain nodes. On the one hand, it can enhance the Byzantine fault-tolerant rate by excluding the lower credit nodes. On the other hand, because the scope

Algorithm 3. CalcNonTxNodeTrust.

```
Input: $node_i$, $TxNodes$, $NonTxNodes$ of $node_i$
Output: Recommended trust value $C_{ij}$
$C_{ij} \leftarrow 0$;
Find the transaction paths between $node_i$ and $node_j$;
for $node_j \in NonTxNodes$ do
    if $node_k \in TxNodes\ of\ node_i\ \&\ node_k \in TxNodes\ of\ node_j$ then
        $C_{ij} = \sum_k C_{ik} C_{kj}$;
    else
        Compute $C_{ij}$ iteratively;
    end
end
```

Algorithm 4. CalcGlobalTrust.

```
Input: $node_i$, node set $Nodes$
Output: Global trust $T_i$ of nodei
$T_i \leftarrow 0$;
for $node_j \in Nodes$ do
    $T_i = \sum C_{ji} T_j$;
end
```

Algorithm 5. getConsensusGroup.

```
Input: Node set $Nodes$, Global trust set $T$, a constant percentage of nodes $d$
       ($0<d\leq 1$)
Output: ConsensusGroup
ConsensusGroup $\leftarrow \varnothing$;
Sort $Nodes$ by $T$;
for $node_i \in Nodes$ do
    if $T_i$ is in the top $d$ then
        Add $node_i$ into ConsensusGroup;
    else
        Exclude $node_i$ from ConsensusGroup;
    end
end
```

of blockchain consensus nodes is narrowed down, we can reduce the number of messages to be delivered and accelerate progress of blockchain consensus consistency.

To achieve this goal, we present the criteria for the construction of blockchain consensus group. A direct idea is to set a global trust threshold, a node will be selected to construct the blockchain consensus group once its global trust value exceeds the pre-defined trust threshold. However, because the global trust values of nodes are dynamic in a limited time, it would make the number of blockchain consensus nodes vary in a large range, which is not benefit for the stability of blockchain consensus consistency. Therefore, we adopt a different strategy to construct the consensus group in this paper. That is to select a constant percentage of nodes with higher global trust values, which are described by Algorithm 5 in details.

In Algorithm 5, we first initialize an empty ConsensusGroup and sort all the nodes by their global trust values. Given a constant percentage of nodes *d* and a $node_i$ in the node set *Nodes*, if the global trust of $node_i$ is in the top *d*, we will add $node_i$ into ConsensusGroup; otherwise, it would be excluded from the ConsensusGroup. Finally, we can select the *d* percentages nodes of the node set *Nodes* with higher global trust values to construct the blockchain consensus group ConsensusGroup. Only these nodes in ConsensusGrop are allowed to participate in the following blockchain consensus process. In this way, we select the subset with the higher global trust values from the consortium blockchain nodes, which can improve the blockchain consensus process greatly.

Algorithm 6. getPrimaryGroup.

```
Input: ConsensusGroup, fixed proportion $m (0 < m \leq 1)$
Output: PrimaryGroup
PrimaryGroup ←∅;
for $node_i \in$ ConsensusGroup do
    if $node_i$ with the global trust value in top $m$ then
        Add $node_i$ to PrimaryGroup;
    else
        Exclude $node_i$ from PrimaryGroup;
    end
end
```

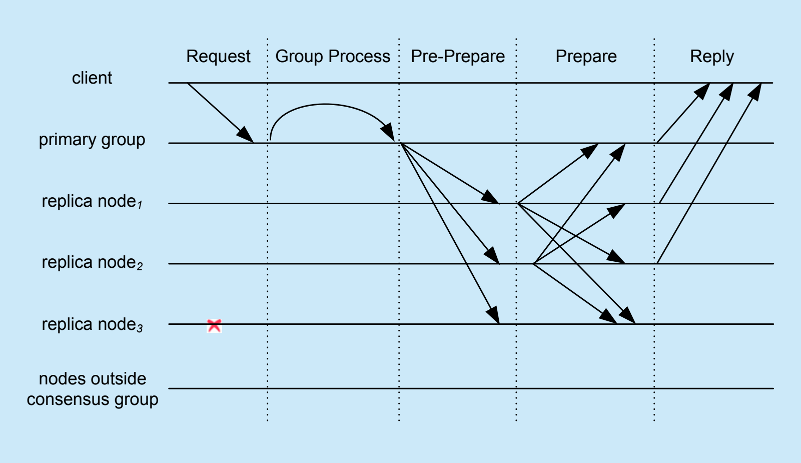

Fig. 4. *The operation process of our T-PBFT.*

5.3 Consensus process

In this phase, we propose an optimized PBFT blockchain consensus algorithm to improve the fault-tolerate rate further. After the consensus group ConsensusGroup is established, the new block will be generated by voting within ConsensusGroup. Recall the process of PBFT [20], when the primary node fails, such as a Byzantine node that behaves arbitrarily or network failure [46], the replica nodes whose timers expired would detect and start a process of view change. However, view change is complex and expensive, which should be avoid as much as possible [47]. To resist against the Byzantine behavior or fail-stop fault of the primary node, and reduce the probability of view change process, we further select a few nodes with the higher trust values from ConsensusGroup to form the primary group to replace the primary node, which is described by Algorithm 6. It shows the selection process of the primary group, which is based on the global trust value of the node. The primary group is responsible for building, recording and confirming the correctness of the new generated block.

Intuitively, by introducing the concept of the primary group, we can see that our T-PBFT can reduce the risk of view change process caused by the single primary node failure or its Byzantine behavior. Specifically, our T-PBFT optimizes PBFT [20], which can be divided into the phases of group process, pre-prepare, prepare and reply in figure 4.

1) In the group process phase, a node in the primary group will package transactions into a pre-generated block and broadcast it to the primary group member nodes for mutual supervision and verification. Once approved, each primary group member node will record the pre-generated block temporarily with the same view. Even if a node in the primary group fails, it could be replaced immediately and would not trigger the process of view change.

2) In the pre-prepare phase, the primary group will broadcast a pre-prepare message including the pre-generated block and the group signature to the replica nodes in the consensus group for audit and verification. Note that we use group signature [48] to enhance the primary group member nodes privacy and reduce the probability of being attacked, which also lower the probability of view change. That is, any node can verify the validity of the primary group signature, but cannot find which primary group member has made it.

3) In the prepare phase, the replica nodes will verify the validity of the pre-generated block. Specifically, each replica node would simulate the execution of the packaged transactions in the pre-generated block with the prearranged transaction order, and then compute the block hash. If it is consistent with the current block hash, the validity check is passed. Once verified, they will broadcast a prepare message with their signatures to each other. Once the number of the prepare messages received by the consensus nodes exceeds $2f$, it will send a reply to the client, where f is the number of the Byzantine nodes in the consensus group that can be computed in Section VI.

4) In the reply phase, when the client has received $f+1$ of the same reply message, the pre-generated block is confirmed and will be added to the end of blockchain. After that, every node in the blockchain network will update their local records.

It is worth noting that in PBFT [20], the pre-prepare and prepare phase are used for ordering requests in the same view, and the commit phase serves for ensuring that requests that commit are totally ordered across views. Obviously, our T-PBFT greatly simplifies the operation process of PBFT by reducing the probability of view change with the primary group. Besides, the member nodes of the primary group will change dynamically for the valid block generation in the consensus process. That is because after a new block is generated, there will be new transactions confirmed, which will lead to the change of the global trust value of all nodes. The global trust value is the key criteria for the selection of the consensus group and the primary group. Therefore, to participate in the consensus process, each node should enhance the global trust value.

VI. Evaluation

In this section, we first theoretically analyze the effectiveness, efficiency and scalability of our T-PBFT, and then compare it with other related BFT-type consensus algorithms.

6.1 Effectiveness

Our T-PBFT is also a multi-stage consensus algorithm, which is based on PBFT [20] and EigenTrust model [25],[26]. Generally, our T-PBFT is different from PBFT in the following aspects. Firstly, consider that those nodes with high global trust values would be more trustworthy at a high probability driven by personal interests and profits, we propose to construct a more trustworthy consensus group based on EigenTrust model to improve the efficiency and scalability of PBFT. Secondly, we reduce the risk of view change process by replacing the single primary node with a primary group. In PBFT, it will randomly selects only one node as the primary node. Once the primary node fails, the process of view change will be triggered to select another primary node, which would increase the consensus complexity. In our T-PBFT, we will choose more than one node with higher trust values to form a primary group. It can conduct mutual supervision actions among member nodes to resist against the Byzantine behavior and reduce the probability of view change process caused by the single primary node fails. For one thing, we use group signature to enhance anonymity of the primary group member nodes, which can reduce the probability of these nodes being attacked. For another thing, because the member nodes in the primary group store the pre-generated block with the same view, if a node fails, it can be replace by any other in the primary group immediately, which avoids the process of view change.

6.2 Efficiency

1) Number of messages delivered: In our T-PBFT, the consensus process is limited within the bounds of ConsensusGroup rather than all the nodes in *Nodes*. Apparently, the number of messages during the consensus process is reduced. Assume that the total number of nodes in the blockchain system is N. The nodes with top d ($0 < d \leq 1$) trust value in the system participate in the consensus process, and the number of the nodes in the primary group is determined as x ($1 \leq x \leq dN$). When the size of the primary group $x = 1$, it will degrade into the process of PBFT with the number of consensus nodes dN . Thus, the number of messages is $O((dN)^2)$. If $d = 1$, it means that all the nodes in *Nodes* would take part in the consensus process, and the number of the messages will be $O(N^2)$, which is the biggest number during the consensus process. Thus it can be seen that the communication complexity is reduced on the whole and only in the worst case, the communication complexity is $O(N^2)$, which is the same with PBFT.

2) Byzantine Fault Tolerance: As we all know, the Byzantine fault-tolerant rate of PBFT algorithm is $\frac{N-1}{3}$. In our T-PBFT, the Byzantine fault-tolerant rate will be optimized. When the system selects the nodes with the trust value in top d ($0<d\leq 1$), the nodes excluded from the consensus group will have no effect on the consensus whose scale is $(1-d)N$. During the consensus process, in order to ensure the correctness of the consensus, every node needs to receive at least $dN-f$ messages from different nodes and verify that the correct messages received are more than f , where f is the number of the Byzantine nodes in the consensus group. Thus, it is necessary to make $dN-f-f \geq f+1$. We can deduce $f \leq \frac{dN-1}{3}$ in the consensus group. The other nodes outside the consensus group can behavior arbitrarily, which has no impact on T-PBFT. In the worse case, all of them are Byzantine nodes. Thus, from the aspect of the whole system, we take both the aspects into accounts, so the maximum number of the Byzantine nodes in the system is the sum of the maximum number of Byzantine nodes f in the consensus group and the maximum number of nodes outside the consensus group, that is $\frac{dN-1}{3}+(1-d)N=(1-\frac{2}{3}d)N-\frac{1}{3}$. When all nodes participate in the consensus, that is d=1, the Byzantine fault-tolerant rate reaches minimum $\frac{N-1}{3}$.

6.3 Scalability

As we all know, most of the consensus algorithms based on Byzantine fault tolerance have poor scalability. When the number of nodes in the algorithm reaches a certain scale, the performance will drop sharply. In T-PBFT, we do not limit the number of the nodes in the system, but by introducing the EigenTrust model and setting of the consensus group, it can reduce the number of the nodes that participate in the consensus process. This will make our T-PBFT be more suitable for a large-scale network environment.

6.4 Comparisons of T-PBFT with other BFT-type consensus

The comparisons of our T-PBFT with the other typical BFT-type consensus algorithms is shown in Table I. Because both PBFT [20] and Tendermint [40] are three-phase commit consensus, their communication complexity are $O(N^2)$. Zyzzyva [22] simplifies PBFT [20] by

Table I. *Comparisons of our T-PBFT with other BFT-type consensus algorithms.*

	PBFT [20]	Zyzzyva [22]	HoneyBadgerBFT [23]	Tendermint [40]	XFT [46]	T-PBFT
Communication complexity	$O(N^2)$	$O(N)$	$O(N^2+N^3\log N)$	$O(N^2)$	$O(N)$	$O(N^2)$
Byzantine fault tolerance	$\frac{N-1}{3}$	$\frac{N-1}{3}$	$\frac{N-1}{3}$	$\frac{N-1}{3}$	$\frac{N-1}{2}$	$(1-\frac{2}{3}d)N-\frac{1}{3}$
Primary node	Single	Single	Single	Single	Single	Group
View change probability	High	High	High	High	High	Low

adopting the primary requests and the client responds immediately, whose communication complexity is $O(N)$. The communication complexity of XFT [46] is the same as crash-tolerant replication protocols, such as Raft [35], Paxos [36], which is $O(N)$. The total communication complexity of HoneyBadgerBFT [23] is about $O(N^2 + N^3logN)$, Our T-PBFT narrows the consensus consortium nodes down to a group of nodes with higher trust values. Theoretically, the number of messages of our T-PBFT delivered among nodes is $O((dN)^2)$ $(0 < d \leq 1)$. At worst, when $d = 1$, that is all the consortium nodes participate in the blockchain consensus group, our T-PBFT degrades into PBFT [20] with the communication complexity is $O(N^2)$.

In terms of Byzantine fault tolerance, these BFT-type consensus algorithms can deal with Byzantine nodes that behave arbitrarily. In PBFT [20], the Byzantine fault-tolerant rate is $\frac{N-1}{3}$. Others such as Zyzzyva [22], HoneyBadgerBFT [23], and Tendermint [40] are extensions of PBFT [20] with the same Byzantine fault-tolerant. XFT [46] assumes that a BFT node cannot control the network and Byzantine nodes simultaneously, which allows to tolerate up to $\frac{N-1}{2}$ Byzantine nodes. As we analyze above, the Byzantine fault-tolerant rate of our T-PBFT is related to the value d. As d increases, the Byzantine fault-tolerant rate tends to decrease. When $d = 1$, the Byzantine fault tolerance gets the minimum value, that is the same with PBFT [20]. For the number of the primary nodes, we conclude that most consensus algorithms in Table I choose a single node as the primary node. T-PBFT chooses a primary group instead of the single primary node to enhance the robustness and reduce the probability of the primary node's Byzantine behavior by mutual supervision. It can not only guarantee the privacy of the nodes in primary group and lower the the risk of being attacked by group signature, but also reduce the probability of view change process.

VII. Conclusion

In this paper, we propose a new consensus algorithm named T-PBFT to optimize PBFT consensus algorithm. In our TPBFT, we construct a trustworthy consensus group based on EigenTrust model to narrow down the number of consensus nodes to improve the consensus efficiency and reduce the communication complexity. Then we also replace the primary node in PBFT with a primary group with higher trust values. By group signature and mutual supervision in T-PBFT, we can enhance the robustness and lower the probability of the view change process. Compared with the other related BFT-type consensus algorithms, theoretical analysis show that our T-PBFT can improve the efficiency and Byzantine fault tolerance. Our T-PBFT can be used as a supplement for the BFT-type consensus algorithms, which will be deployed in a blockchain prototype.

Acknowledgment

This work was supported by Nature Key Research and Development Program of China (2017YFB1400700), the National Natural Science Foundation of China (61602537, U1509214), the Central University of Finance and Economics Funds for the Youth Talent Support Plan (QYP1808) and First-Class Discipline Construction in 2019, open fund of Key Laboratory of Grain Information Processing and Control (KFJJ-2018-202).

References

[1] S. Nakamoto,"Bitcoin: A peer-to-peer electronic cash system," Available on:https://bitcoin.org/bitcoin.pdf/, 2009.

[2] T. T. A. Dinh, R. Liu, M. Zhang *et al.*,"Untangling blockchain: A data processing view of blockchain systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 7, pp. 1366–1385, 2018.

[3] Y. Yuan and F.-Y. Wang,"Blockchain and cryptocurrencies: Model, techniques, and applications," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 9, pp. 1421–1428, 2018.

[4] V. Buterin,"On public and private blockchains," *Ethereum blog*, vol. 7, 2015.

[5] Z. Zheng, S. Xie, H. Dai *et al.*,"An overview of blockchain technology: Architecture, consensus, and future trends," in *2017 IEEE International Congress on Big Data*, 2017, pp. 557–564.

[6] V. Buterin *et al.*,"A next-generation smart contract and decentralized application platform," *white paper*, 2014.

[7] G. Greenspan,"Multichain private blockchainłwhite paper," Available on: http://www.multichain.com/download/MultiChain-White-Paper.pdf, 2015.

[8] E. Androulaki, A. Barger, V. Bortnikov *et al.*,"Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the 13th EuroSys Conference*, 2018, pp. 1–15.

[9] T. F. and S. B.,"Bitcoin and beyond: A technical survey on decentralized digital currencies," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 2084–2123, 2016.

[10] H. R. Hasan and K. Salah,"Proof of delivery of digital assets using blockchain and smart contracts," *IEEE Access*, vol. 6, pp. 65 439–65 448, 2018.

[11] N. Z. Aitzhan and D. Svetinovic,"Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 5, pp. 840–852, 2018.

[12] Z. Shae and J. J. Tsai,"On the design of a blockchain platform for clinical trial and precision medicine," in *IEEE 37th International Conference on Distributed Computing Systems*, 2017, pp. 1972–1980.

[13] M. Turkanovic, M. Ho¨lbl, K. Kosic, M. Hericko, and A. Kamisalic,"Eductx: A blockchain-based higher education credit platform," *IEEE Access*, vol. 6, pp. 5112–5127, 2018.

[14] G. Perboli, S. Musso, and M. Rosano,"Blockchain in logistics and supply chain: A lean approach for designing real-world use cases," *IEEE Access*, vol. 6, pp. 62 018–62 028, 2018.

[15] W. Wang, D. T. Hoang, Z. Xiong *et al.*,"A survey on consensus mechanisms and mining management in blockchain networks," *IEEE Access*, vol. 7, pp. 22 328 – 22 370, 2019.

[16] S. Bano, A. Sonnino, M. Al-Bassam *et al.*,"Sok: Consensus in the age of blockchains," *arXiv preprint arXiv:1711.03936*, 2017.

[17] V. Gramoli,"From blockchain consensus back to byzantine consensus," *Future Generation Computer Systems*, 2017.

[18] M. Vukolic´,"The quest for scalable blockchain fabric: Proof-of-work vs. bft replication," in *International Workshop on Open Problems in Network Security*, 2015, pp. 112–125.

[19] S. King and S. Nadal,"Ppcoin: Peer-to-peer crypto-currency with proofof-stake," *self-published paper*, 2012.

[20] M. Castro and B. Liskov,"Practical byzantine fault tolerance," in *Proceedings of the 13rd Symposium on Operating Systems Design and Implementation*, vol. 99, 1999, pp. 173–186.

[21] J. Behl, T. Distler, and R. Kapitza,"Scalable bft for multi-cores: Actorbased decomposition and consensus-oriented parallelization," in *The 10th Workshop on Hot Topics in System Dependability*, 2014.

[22] R. Kotla, L. Alvisi, M. Dahlin *et al.*,"Zyzzyva: speculative byzantine fault tolerance," *ACM SIGOPS Operating Systems Review*, vol. 41, no. 6, pp. 45–58, 2007.

[23] A. Miller, Y. Xia, K. Croman *et al.*,"The honey badger of bft protocols," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 31–42.

[24] A. Biryukov, D. Feher, and D. Khovratovich,"Guru: Universal reputation module for distributed consensus protocols," *IACR Cryptology ePrint Archive*, pp. 1–20, 2017.

[25] S. D. Kamvar, M. T.Schlosser, and H. Garcia-Molina,"The eigentrust algorithm for reputation management in p2p networks," in *Proceedings of the 12th international conference on World Wide Web*. ACM, 2003, pp. 640–651.

[26] K. Lu, J. Wang, and M. Li,"An eigentrust dynamic evolutionary model in p2p file-sharing systems," *Peer-to-Peer Networking and Applications*, vol. 9, no. 3, pp. 599–612, 2016.

[27] N. Alzahrani and N. Bulusu,"Towards true decentralization: A blockchain consensus protocol based on game theory and randomness," in *Proceedings of the 9th International Conference on Decision and Game Theory for Security*, 2018, p. 465.

[28] M. Ball, A. Rosen, M. Sabin *et al.*,"Proofs of useful work." *IACR Cryptology ePrint Archive*, pp. 1–28, 2017.

[29] S. Park, K. Pietrzak, A. Kwon *et al.*,"Spacemint: A cryptocurrency based on proofs of space," pp. 480–499, 2018.

[30] L. Ren,"Proof of stake velocity: Building the social currency of the digital age," *Self-published white paper*, 2014.

[31] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld,"Proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract]," *ACM SIGMETRICS Performance Evaluation Review*, vol. 42, no. 3, pp. 34–37, 2014.

[32] T. Duong, L. Fan, and H.-S. Zhou,"2-hop blockchain: Combining proofof-work and proof-of-stake securely," *IACR Cryptology ePrint Archive*, pp. 1–45, 2016.

[33] I. Eyal, A. E. Gencer, E. G. Sirer *et al.*,"Bitcoin-ng: A scalable blockchain protocol," in *Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation*, 2016, pp. 45–59.

[34] B. M. Oki and B. H. Liskov,"Viewstamped replication: A new primary copy method to support highly-available distributed systems," in *Proceedings of the 7th ACM Symposium on Principles of Distributed Computing*, 1988, pp. 8–17.

[35] D. Ongaro and J. K. Ousterhout,"In search of an understandable consensus algorithm." in *USENIX Annual Technical Conference*, 2014, pp. 305–319.

[36] L. Lamport,"Paxos made simple," *ACM Sigact News*, vol. 32, no. 4, pp. 18–25, 2001.

[37] "Hyperledger fabric v0.6.0," Available on: https://github.com/ hyperledger/fabric/releases/tag/v0.6.0-preview, 2007.

[38] "Hyperledger fabric v1.4.0," Available on: https://hyperledger-fabric. readthedocs.io/en/latest/whatis.html, 2018.

[39] A. Clement, E. L. Wong, L. Alvisi *et al.*,"Making byzantine fault tolerant systems tolerate byzantine faults," in *Proceedings of the 6th USENIX Symposium on Networked. Systems Design and Implementation*, vol. 9, 2009, pp. 153–168.

[40] J. Kwon,"Tendermint: Consensus without mining," *Draft v. 0.6, fall*, 2014.

[41] G. Golan-Gueta, I. Abraham, S. Grossman *et al.*,"SBFT: a scalable decentralized trust infrastructure for blockchains," *arXiv preprint arXiv:1804.01626*, 2018.

[42] N. W. Paper,"Delegated byzantine fault tolerance consensus algorithm," Available on: https://docs.neo.org/en-us/basic/consensus/consensus. html, 2016.

[43] R. Pass and E. Shi,"Thunderella: Blockchains with optimistic instant confirmation," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2018, pp. 3–33.

[44] Y. Gilad, R. Hemo, S. Micali *et al.*,"Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th Symposium on Operating Systems Principles*, 2017, pp. 51–68.

[45] A. Kiayias and A. Russell,"Ouroboros-bft: A simple byzantine fault tolerant consensus protocol," *IACR Cryptology ePrint Archive*, pp. 1– 21, 2018.

[46] S. Liu, P. Viotti, C. Cachin *et al.*,"Xft: Practical fault tolerance beyond crashes." in *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation*, 2016, pp. 485–500.

[47] A. Dinh,"Subtle details in pbft," Available on: https://ug93tad.github. io/pbft/, 2017.

[48] D. Chaum and E. van Heyst,"Group signatures," in *Advances in Cryptology-EUROCRYPT91*, 1991, pp. 257–265.

Biographies

Sheng Gao, received the B.S. degree in information and computation science from Xi'an University of Posts and Telecommunications in 2009, and the Ph.D. degree in computer science and technology from Xidian University in 2014. He is now an Associate Professor in the School of Information at Central University of Finance and Economics. He has published more than 30 research papers in refereed international journals and conferences. His current research interests include data security, privacy computing, and blockchain technology.

Tianyu Yu, received the B.S. degree in computer science and technology from Central University of Finance and Economics in 2017. He is now pursuing his M.S. degree in Central University of Finance and Economics. His current research interests mainly focus on scalability and performance in blockchain.

Jianming Zhu, received the M.S. degree in computer application from Taiyuan University of Technology in 1998, and the Ph.D. degree in computer application technology from Xidian University in 2004. He is now a Professor in the School of Information at Central University of Finance and Economics. He has published 5 books and over 100 research papers in refereed international journals and conferences. His research interests include financial information security, wireless network security, and blockchain.

Wei Cai, received the B.Eng. degree in software engineering from Xiamen University, China, in 2008, the M.S. degree in electrical engineering and computer science from Seoul National University, South Korea, in 2011, and the Ph.D. degree in electrical and computer engineering from The University of British Columbia (UBC), Vancouver, Canada, in 2016. From 2016 to 2018, he was a Post-Doctoral Research Fellow with UBC. He joined the School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen, in 2018, where he is currently an Assistant Professor. His recent research interests include software systems, cloud and edge computing, blockchain systems, and networked video games.