Computer Modeling in Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2023.046826

ARTICLE

On Designs of Decentralized Reputation Management for Permissioned Blockchain Networks

Jinyu Chen[1], Long Shi[1,*], Qisheng Huang[2], Taotao Wang[3] and Daojing He[4]

[1]School of Electronic and Optical Engineering, Nanjing University of Science and Technology, Nanjing, 210094, China

[2]School of Mechanical Engineering and Automation, Harbin Institute of Technology, Shenzhen, 518055, China

[3]College of Electronics and Information Engineering, Shenzhen University, Shenzhen, 518060, China

[4]School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, 518055, China

*Corresponding Author: Long Shi. Email: slong1007@gmail.com

Received: 16 October 2023 Accepted: 10 November 2023 Published: 29 January 2024

ABSTRACT

In permissioned blockchain networks, the Proof of Authority (PoA) consensus, which uses the election of authorized nodes to validate transactions and blocks, has been widely advocated thanks to its high transaction throughput and fault tolerance. However, PoA suffers from the drawback of centralization dominated by a limited number of authorized nodes and the lack of anonymity due to the round-robin block proposal mechanism. As a result, traditional PoA is vulnerable to a single point of failure that compromises the security of the blockchain network. To address these issues, we propose a novel decentralized reputation management mechanism for permissioned blockchain networks to enhance security, promote liveness, and mitigate centralization while retaining the same throughput as traditional PoA. This paper aims to design an off-chain reputation evaluation and an on-chain reputation-aided consensus. First, we evaluate the nodes' reputation in the context of the blockchain networks and make the reputation globally verifiable through smart contracts. Second, building upon traditional PoA, we propose a reputation-aided PoA (rPoA) consensus to enhance security without sacrificing throughput. In particular, rPoA can incentivize nodes to autonomously form committees based on reputation authority, which prevents block generation from being tracked through the randomness of reputation variation. Moreover, we develop a reputation-aided fork-choice rule for rPoA to promote the network's liveness. Finally, experimental results show that the proposed rPoA achieves higher security performance while retaining transaction throughput compared to traditional PoA.

KEYWORDS

Blockchain; reputation management; PoA; throughput; security; decentralization

1 Introduction

As a vital component of blockchain technology, the distributed consensus mechanism is essential to achieve distributed consistency in blockchain data among trustless nodes in a decentralized manner. In general, the design of any consensus mechanism is faced to make a well-known trade-off that prevents it from achieving the following three aspects: security, scalability, and decentralization (i.e.,

known as the blockchain trilemma). Specifically, the significance of security in blockchain networks is to establish a secure environment for transactions. Once a block containing transactions is added to the chain, it cannot be retroactively altered, thus protecting the blockchain from fraud and double spending. The significance of decentralization is to mitigate the negative impact on blockchain security caused by a single point of failure. Therefore, the consensus mechanism should be well-designed to defend itself against malicious manipulation and disruption. Moreover, the consensus mechanism is expected to handle an increasingly growing number of transactions. Last but not least, the consensus mechanism is achieved without any coordination of centralized nodes. To date, extensive efforts have been dedicated to optimizing these three aspects in different blockchain networks.

In permissionless (or public) blockchain networks, any node can participate in the consensus without the coordination of a central authority. For public networks like Bitcoin [1], the Proof of Work (PoW) consensus has become popular to achieve distributed consensus among fully decentralized nodes. Under PoW, the nodes are required to solve an extremely difficult cryptographic puzzle by exhausting their computing power, and the winner who has solved this puzzle is eligible to publish a block. Apparently, PoW helps the decentralized network reach consensus securely at the cost of a lot of expensive computing power and energy spent. In contrast to PoW, the Proof of Stake (PoS) [2] consensus stipulates that the node with the highest stake (e.g., the Beacon [3] in Ethereum [4]) gains the authority to publish a block. As such, it is more likely for the nodes with a large number of tokens to monopolize the block generation, which exacerbates the issue of wealth gap.

The permissioned blockchain networks, in which participants must be authorized, typically demand higher throughput and more flexible network access but degraded centralization than their permissionless peers. Additionally, excessive computational work and energy consumption are undesirable in permissioned blockchain networks. Therefore, improving the throughput and scalability of blockchain networks under energy-efficient consensuses remains challenging. In this context, the aforementioned consensus mechanisms for permissionless networks are unsuitable for permissioned networks since both PoW and PoS maintain blockchain security by making it difficult and expensive for malicious nodes to misbehave. Currently, permissioned blockchain networks have been widely studied in the literature, such as the Internet of Underwater Things [5], Internet of Battlefield Things [6], Internet of Vehicles [7], and Crowdsourcing [8]. Furthermore, commonly used consensus mechanisms for permissioned blockchains include Practical Byzantine Fault Tolerance (PBFT) [9] and Proof of Authority (PoA) [10].

PBFT is the first practical Byzantine fault-tolerant consensus algorithm that can achieve consensus when a minority of nodes misbehave (i.e., Byzantine nodes). In the presence of f Byzantine nodes out of a total of $3f + 1$ nodes, this consensus protocol ensures the strong consistency of message transmission, as long as there are no fewer than $2f + 1$ honest nodes operating normally. The advantage of PBFT is that it reduces the complexity of Byzantine fault-tolerant algorithms from exponential to polynomial levels. However, as the number of nodes increases, the message complexity grows quadratically. Therefore, PBFT is more suitable for small-scale blockchain systems. In reality, permissioned blockchains like Hyperledger Fabric [11] and FISCO BCOS [12] adopt PBFT as their consensus algorithms.

The Proof of Authority (PoA) consensus, which integrates PoS and BFT consensus, has been widely applied in permissioned networks such as the Goerli Testnet [13] and POA Networks [14]. In this consensus, a set of authorized nodes is elected and takes turns to verify transactions and generate new blocks. Compared to other consensus algorithms, the PoA algorithm features higher throughput. Furthermore, since the identities of authorized nodes have been authenticated, PoA can achieve

enhanced security and resilience against malicious behavior. However, the round-robin consensus of authorized nodes could potentially result in centralization. The centralization issue in blockchain can give rise to mining pools and even enable the execution of a 51% attack [15]. Besides, the round-robin consensus also lacks anonymity. Due to the absence of anonymity, the blockchain nodes in traditional PoA can be easily predicted, tracked, and attacked. To address these problems inherent in PoA, current improvements include the introduction of the committee-endorsing mechanism [16] and verifiable random functions [17]. These improvements aim to optimize the security, efficiency, and adaptability of the PoA consensus. For example, VeChain [18] introduces two new mechanisms, i.e., Committee-Endorsing Mechanism and Block Finality Mechanism, to promote the security of traditional PoA. Nevertheless, these efforts still fail to address the security issues posed by the trustworthiness and anonymity of consensus nodes. It is worth noting that these works also introduce challenges such as consensus overload and reduced generalization.

Driven by these challenges, we propose a decentralized reputation management mechanism (DRM) for permissioned blockchain networks. Our proposal features both the off-chain reputation evaluation and the on-chain reputation-aided PoA consensus. The goal is to enhance security compared to traditional PoA without sacrificing transaction throughput. Our primary contributions are as follows:

- We design a blockchain-based reputation management mechanism in a modular fashion, which builds upon the contract layer and does not tamper with the core components of the blockchain. This design ensures the generalization of the mechanism across various permissioned blockchain networks.
- We develop a smart contract-based reputation evaluation scheme for the blockchain nodes, wherein the nodes' reputation values are automatically generated, recorded, and verified through specifically designed on-chain contracts. Furthermore, it is shown that the scheme can defend against various on-chain attacks such as collusion attacks and data tampering attacks.
- We propose a reputation-aided PoA (rPoA) consensus to enhance security without sacrificing the throughput of traditional PoA. Particularly, rPoA can incentivize nodes to autonomously form committees based on reputation authority, which prevents block generation from being tracked through the randomness of reputation variation. Moreover, we develop a reputation-aided fork-choice rule for rPoA to promote the network's liveness.

Experimental results show that the proposed DRM can enhance security without sacrificing throughput and decentralization. The rest of the paper is organized as follows: Section 2 introduces the related works. Section 3 describes the proposed framework and workflow. The mechanism implementation is detailed in Section 4. Section 5 presents the experimental results of DRM. Finally, Section 6 concludes the paper.

2 Related Work

This section introduces related consensus designs and reputation schemes for permissioned blockchain networks from both community and academic research perspectives. Community research pays more attention to practical applications and implementations, while academic research mainly focuses on theoretical models and algorithm designs.

In community research, Aura [19] and Clique [20] are two mainstream implementations of PoA. Aura is a PoA consensus algorithm based on Rust and implemented in the Ethereum client Parity. Under this protocol, two local queues are maintained at each authority's side: one for transactions

and the other for pending blocks. The authority adds transactions to the new block and broadcasts the block to other authorities in its turn. If other authorities accept the proposed block, the new block is added to the blockchain. Clique implements PoA in the Ethereum client Geth based on the Go language. This algorithm sorts all authorized node accounts alphabetically, and they take turns proposing the blocks according to this order. Islam et al. analyzed the advantages and disadvantages of Aura and Clique in [21]. Clique provides higher availability and transaction throughput but lower security and consistency than Aura. The blockchains in the Aura network are immediately consistent, while those stored by Clique authorities eventually become consistent. Nevertheless, both Aura and Clique share similar drawbacks, such as the requirement for predetermined authorities and a loss of decentralization characteristics.

In academic research, the authors of [22] proposed a multi-stage optimized PBFT consensus algorithm (called T-PBFT) based on the EigenTrust model, which evaluates node trust according to the transactions between nodes. Compared with traditional PBFT, T-PBFT replaces a single primary node that proposes the block with a primary group, and membership dynamically changes based on reputation value. However, this algorithm faces a centralization problem due to the reduced number of consensus nodes. In addition, the authors of [23] proposed a blockchain-based trust management scheme in vehicular networks. In this scheme, roadside units (RSUs) calculate trust value offsets of vehicles and package data into a block. Then, RSUs compete to add their block to the blockchain under the joint PoW and PoS consensus mechanism. Recently, the authors of [24] proposed an improved PoW consensus to decentralize the computing power of centralized mining pools. Miners are rewarded with reduced mining difficulty if they have accumulated sufficient proof of work age. The authors of [25] proposed a secure and accountable data management scheme based on a redactable blockchain. This scheme allows trustworthy management of on-chain data in a decentralized manner without relying on trusted blockchain managers. In [26], the authors conceptualized a blockchain-based decentralized framework for crowdsourcing. In this framework, a crowd of workers can solve a requester's task without relying on any third trusted institution. Each worker is assigned a reputation, which serves as an important reference for requesters when selecting workers. A high reputation indicates good past performance in task completion, while a low reputation may limit their participation in certain tasks. High efforts and good performances can improve reputation. The authors of [27] proposed a trust management scheme to evaluate the trusts of all UAVs participating in decentralized spectrum sharing. In this scheme, trust increases depend on the legal use of spectrum allocation and honest reporting of spectrum sensing. This scheme incentivizes UAVs to comply with legal rules of spectrum sharing in blockchain and punishes malicious behavior that violates spectrum sharing rules or provides misleading spectrum sensing results. The authors of [28] proposed a blockchain-based trust management mechanism for crowdfunding to address fraudulent behavior in a zero-trust environment. The mechanism introduces an auditor committee to credibly audit crowdfunding projects. A high reputation value affects auditor income and auditor committee selection results.

3 System Model

In this section, we present the architecture of the proposed DRM mechanism for permissioned blockchain networks.

3.1 Architecture

As shown in Fig. 1, the proposed DRM mechanism consists of application, contract, and consensus layers. In the application layer, each user generates data from various applications (e.g., decentralized applications, Internet of Things, and mobile applications), which is aggregated by the Data Aggregation Contract (DAC) and then recorded on the blockchain. The evaluation module accesses this data for evaluation purposes and generates a score for the users. This score is securely stored within the DAC. Then, DRM transforms the DAC into a verifiable evaluation that serves as a foundational proof of reputation source (see Section 3.2). In the contract layer, the Reputation Management Contract (RMC) accesses the verifiable evaluation and changes the reputation value for the relevant participants based on this proof (see Section 3.3). In the consensus layer, the reputation value is used to determine which nodes are eligible to generate the block and the proof of forking choice. At the bottom of Fig. 1, we demonstrate how to resolve blockchain forks based on reputation values (see Section 3.4). It is worth noting that the proposed mechanism is applicable to decentralized reputation management for a wide range of permissioned networks.

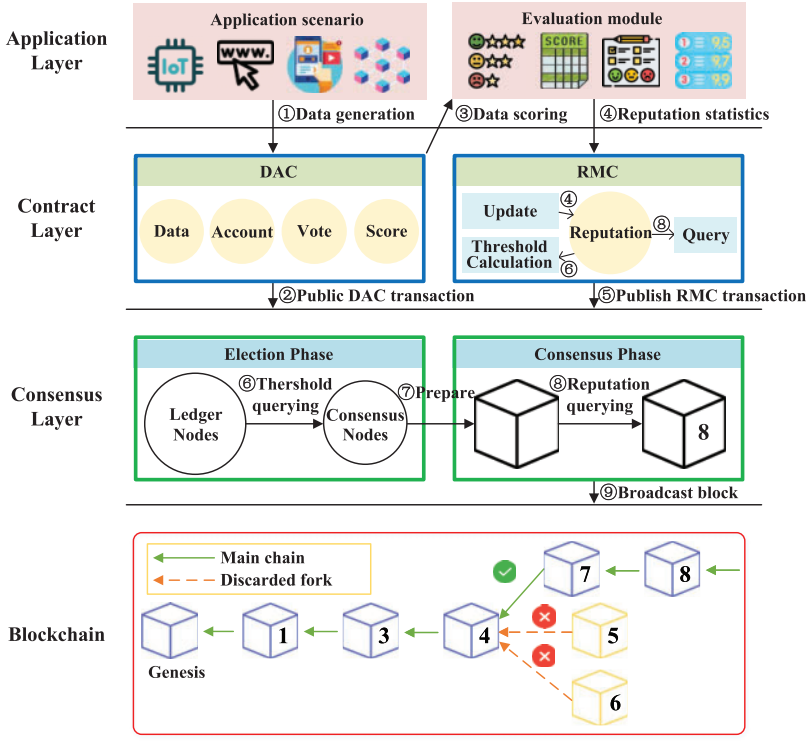


Figure 1: Layered structure of the proposed DRM mechanism and workflow

3.2 Application Layer

In the application layer, we define four types of roles according to their behaviors as follows:

- *Users* generate data from various applications.
- *Reviewers* access users' data for review and generate verifiable evaluations.
- *Validators* verify the legitimacy of evaluations from the reviewers.
- *Evildoers* are users, reviewers, or validators who engage in misconduct.

At the user side, the usage of various applications continuously generates data. For example, in machine learning applications [29,30], users generate data for identification and classification tasks by using advanced learning algorithms. In crowdsourcing applications [26], users cooperate to finish a common task by contributing their local data.

The evaluation module is designed for reviewers to evaluate the data from the application scenarios and for validators to verify the scores from the reviewer. For example, in machine learning applications, the user's data can be scored according to the levels of learning accuracy, while in crowdsourcing applications, the user's data can be scored based on the quality of the contributed data. This entire evaluation process is recorded on the blockchain via smart contracts (see Section 3.3).

3.3 Contract Layer

In the contract layer, we deploy two smart contracts to facilitate off-chain reputation evaluation. These smart contracts provide account reputation statistics, queries, and reputation threshold calculation capabilities. Using Algorithms 4.1, 4.2, 4.3, and 4.4 in the consensus layer, node reputation can be globally verified under the RMC's Application Binary Interface (ABI) and bytecode.

The *Data Aggregation Contract (DAC)* serves as a repository for storing verifiable evaluations. It includes five essential attributes: user account, user data, user score, reviewer account, and validator signature. Upon its creation, this contract aggregates the user's account and data, then deploys it onto the blockchain network and broadcasts the contract address.

The *Reputation Management Contract (RMC)* is deployed on the permissioned blockchain during the genesis block generation. It is used to manage the reputation of all participants. This contract is used to calculate the user's reputation based on verifiable evaluations and query their reputation values. Additionally, the contract can determine the reputation threshold for the ledger node to become a consensus node (see Section 3.4).

Validators also engage with the scoring process through the contract account. Their role involves scrutinizing and either accepting or rejecting the scores. Consensus is achieved when more than two-thirds of the validators accept the score. Subsequently, the reputation management contract adjusts reputations based on verifiable evaluations. Furthermore, if a user disagrees with the scoring result, the verifiable evaluation will serve as an important basis for appeal.

3.4 Consensus Layer

Building upon traditional PoA, we design an on-chain reputation-aided consensus protocol named Reputation-aided Proof of Authority (rPoA) to enhance security without sacrificing throughput. Since DRM is modular and built upon the contract layer, this mechanism has no impact on the performance (i.e., throughput) of rPoA. Additionally, rPoA incentivizes nodes to autonomously conduct distributed consensus based on reputation authority, thereby addressing the centralization issue of traditional PoA in permissioned blockchain networks, i.e., diminishing the control of authorities over consensus nodes. Specifically, rPoA comprises the election phase and the consensus phase, where we refer to participating nodes in the blockchain network as ledger nodes, and nodes with the ability to generate blocks are referred to as consensus nodes. Notably, the election of consensus nodes is based on reputation and is conducted by all ledger nodes.

We call the interval between two consecutive checkpoint blocks an epoch. Consider that a checkpoint block is broadcasted at the beginning of each election phase. The checkpoint block contains a list of accounts known as signers, and this list is sorted by the dictionary order of their accounts. The signer is a ledger node whose reputation value surpasses the predesigned threshold and is eligible to

become a consensus node. When this block is accepted and verified by all ledger nodes, it implies that the election phase terminates. In the next epoch, new consensus nodes will generate the block.

The iterative sequence of the consensus phase is known as a round. At the conclusion of each round, each consensus node broadcasts its respective block containing an individual reputation value. It is stressed that rPoA determines whether the reputation value in the current block is the highest among all nodes. For the blocks with smaller reputation values, Fig. 2 shows that a fixed delay is introduced during block broadcasting. The fixed latency allows for the rapid confirmation of temporary forks, thereby enhancing the network's overall liveness. In contrast to random latency, fixed latency reduces the lifetime of forks. Please refer to Algorithm 4.3 for the details of the implementation of fixed latency.

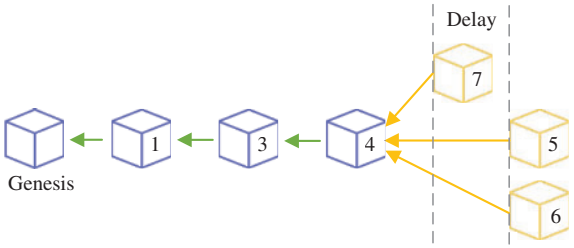
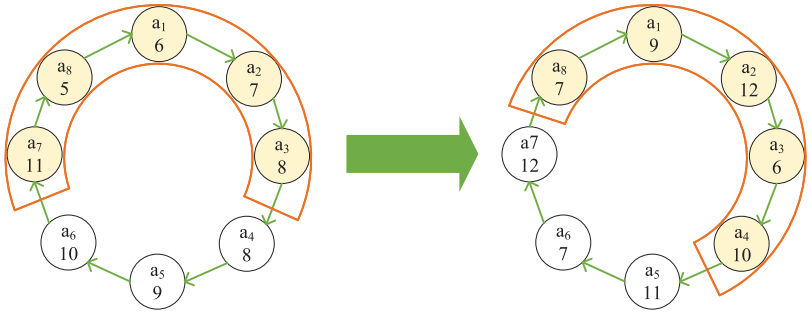


Figure 2: Delay for reputation-aided block proposal

As shown in Fig. 3, rPoA is designed to prevent evildoers from destroying the network by proposing a large number of blocks (e.g., Replay attacks). Therefore, each consensus node is only allowed to propose a single block per $\frac{|\text{signers}|}{2} + 1$ rounds, where |signers| denotes the total number of signers. Meanwhile, when other ledger nodes in the network receive a new block, they will check whether the account of the block proposer exists in the list of consensus nodes, whether the node has recently submitted a block, and whether the block carries the same reputation value as that of its proposer. It is notable that the reputation value variation of each consensus node is utterly random in each round because the reputation value sources are unpredictable. Thanks to this, DRM can prevent the next block generation node from being predicted, thus increasing the security of the blockchain network.



(a) Round t1 (b) Round t2

Figure 3: Reputation-aided round-robin block proposal of rPoA

We implement a novel fork-choice rule for rPoA, which adopts the highest reputation value principle, different from the longest chain principle used in traditional PoA. Specifically, each

consensus node prioritizes the chain with the highest cumulative reputation as the main chain. If the node with the highest reputation value is offline during a given round, the node with the next highest reputation value will be accepted by other nodes. If there exist multiple chains with the same reputation value, the consensus node randomly follows a branch. When a fork occurs, honest nodes within the network actively maintain the chain with the highest reputation, ultimately leading to stability. This rule offers significant benefits in reducing a single point of failure and promoting the liveness of the blockchain network. For instance, as illustrated in Fig. 4, a block should have been proposed by a node with a reputation value of 7 in round 3. However, due to the offline status of this node, a fork emerges in the subsequent round 4. Nevertheless, the highest reputation value chain can be finalized since honest nodes diligently follow the chain with the highest reputation value.

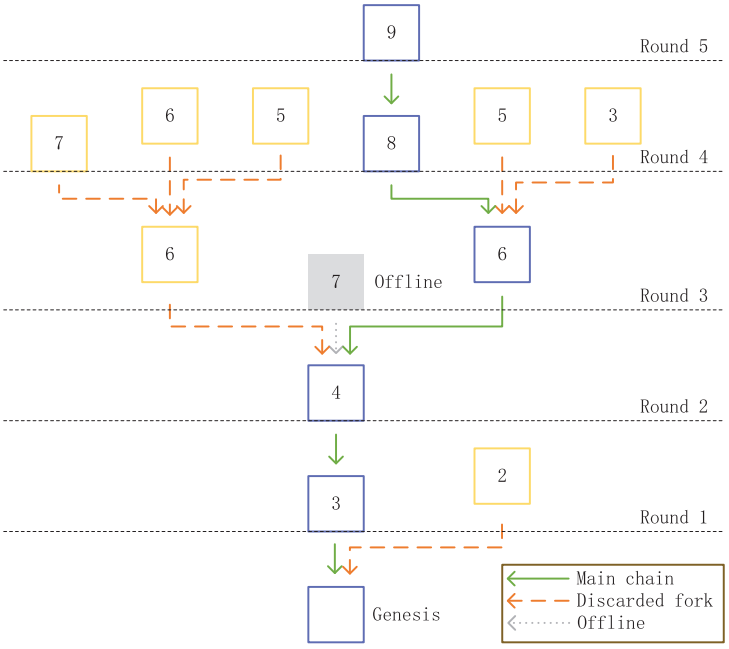


Figure 4: Reputation-aided fork-choice rule of rPoA

4 Implementation of DRM

In this section, we implement the DRM mechanism on the Ethereum blockchain platform. First, we introduce the blockchain platform. Then, we deploy two types of contracts (i.e., Data Aggregation Contract and Reputation Management Contract). Finally, we design four algorithms in the consensus layer.

4.1 Overview of Ethereum Blockchain

Bitcoin and Ethereum, while both utilizing blockchain technology, exhibit distinct characteristics in usage, underlying consensus, transaction speed, and programming language functionality. In particular, Ethereum distinguishes itself as a versatile decentralized application development platform by offering Turing completeness, which allows the creation of complex smart contracts. In this context, we select Ethereum as the blockchain platform for the proposed DRM mechanism. In order to ensure compatibility with Ethereum code and leverage its established data structures, we adopt its core data structure in our implementation. Fig. 5 depicts the block data structure utilized in Ethereum.

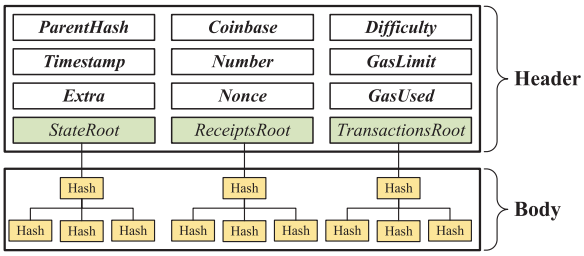


Figure 5: Block structure of Ethereum

In the rPoA consensus, we propose to share some fields of the block structure in Ethereum. The sharing fields can significantly reduce the amount of code modifications on the Ethereum client (i.e., Geth), which has the benefit of improving the efficiency of DRM execution and reducing potential DRM bugs. For instance, the *Difficulty* field is one of the sharing fields, which represents mining difficulty in the *Ethash* consensus. Notably, the proposed rPoA does not require any mining. Thus, we redefine this field as a block reputation value. Thanks to this fine-tuning, we innovate the implementation of the reputation-aided fork-choice rule compared with the weight-aided fork-choice rule of GHOST [31] in Ethereum. It is known that each block has a weight. For example, in PoW, the weights are the difficulty of solving the puzzle in PoW and the number of tokens the holder owns in PoS respectively. In contrast, the reputation value represents the block's weight in rPoA.

4.2 Contract Layer Implementation

We provide a comprehensive overview of the specific implementation solutions employed within the contract layer, encompassing the data aggregation and reputation management contract. We use Solidity as the programming language of smart contracts on the Ethereum platform. Solidity offers robust security features and reliability for writing smart contracts and enables the execution of smart contracts on the decentralized Ethereum Virtual Machine (EVM) [32]. In Section 4.2.3, we display the workflow for generating verifiable evaluations.

4.2.1 Data Aggregation Contract (DAC)

In Fig. 6, the DAC contract stores the user data, account, score, reviewer account, and validator vote and is finally stored as verifiable evaluations on the blockchain. It has five attributes: *UserData*, *UserAccount*, *UserScore*, *ReviewerAccount*, and *ValidatorsVote*. The *UserData* stores the data generated by users. The *UserAccount* records the user's account for reputation management. The *UserScore* represents the reviewer's score of the user data. The *ReviewerAccount* identifies the reviewer account. The *ValidatorsVote* is implemented through a key-value structure that stores the validator's account and whether they approve the score given by the reviewer to the user.

In addition, this contract consists of three functions: *ScoreEvaluating*, *Accept*, and *Decline*. The *ScoreEvaluating* function, designed for reviewers (which means restricting the user's use of the function), evaluates the user data and stores the result in the *UserScore* attribute. At the same time, the account of the reviewer is automatically filled in the *ReviewerAccount*. The main implementation of this function allows a flexible design of the permissioned blockchain with different application scenarios. In DRM, we randomly generate data for the users. Specifically, given randomly generated data, we compute its hash value and count the number of zero values within the hash. Each occurrence of a zero value is attributed with one point as a reward. The *Accept* (or *Decline*) function is designed

for validators (which means restricting both the user and reviewer's use of the function). In the same evaluation function, the score for the same type of data is the same. For example, in DRM, the number of zeros calculated by the validator is the same as that calculated by the reviewer. However, a malicious reviewer may not be scoring based on the rules. The *Accept* (or *Decline*) function is called when the validator believes the score is reasonable (or unreasonable), and it adds the validator's account to the validator's list. If the validator's account already exists in this list, then the validator is not allowed to vote again, which can prevent the malicious validator from "standing on both sides" by agreeing with and disagreeing with the score.

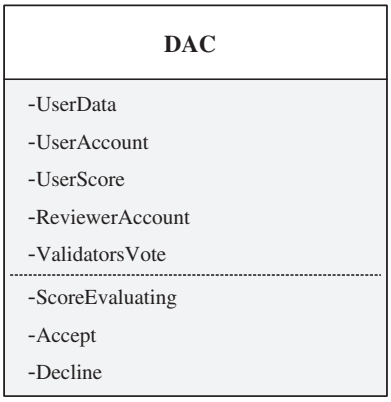


Figure 6: Data structure of DAC

4.2.2 Reputation Management Contract (RMC)

In Fig. 7, the RMC contract is designed to manage the reputation of all participants (i.e., *Users*, *Reviewers*, and *Validators*). It contains a key-value structure for recording account reputation and three functions. The *ReputationValues* attribute employs a key-value data structure to store and manage the reputation associated with accounts. The *ReputationUpdating* function plays a crucial role in updating the reputation of specific accounts. It accepts the score obtained from verifiable evaluations (i.e., DAC) and utilizes predefined rules to modify the participants' reputation value. In DRM, the reputation value is updated as

$$R = \sum \Delta r_{\text{user}} + \sum \Delta r_{\text{reviewer}} + \sum \Delta r_{\text{validator}} \tag{1}$$

where each participant is tagged with a reputation R, and Δr denotes the change of reputation. The different types of participants are identified by subscripts (e.g., the reputation values of user, reviewer, and validator are identified by r_{user}, $r_{reviewer}$, and $r_{validator}$, respectively).

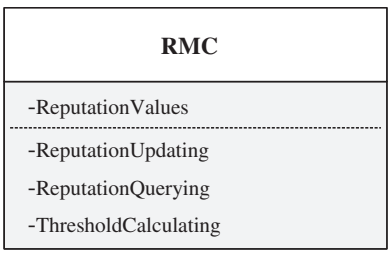


Figure 7: Data structure of RMC

The reputation values of the participants in the permissioned blockchain come from the accumulation of each verifiable evaluation, which is permanently stored on the blockchain in the form of DAC. There are three types of roles in verifiable evaluation, and their respective reputation values are updated as

$$\Delta r_{\text{user}} = \begin{cases} a \text{ in DAC, if DAC is reasonable} \\ 0, \text{ else} \end{cases} \tag{2}$$

$$\Delta r_{\text{reviewer}} = \begin{cases} b, \text{ if DAC is reasonable} \\ -c, \text{ else} \end{cases} \tag{3}$$

$$\Delta r_{\text{validator}} = \begin{cases} d, \text{ if the validator accepts reasonable DAC or declines unreasonable DAC} \\ -e, \text{ if the validator declines reasonable DAC or accepts unreasonable DAC} \end{cases} \tag{4}$$

where a, b, c, d, and e are the positive values. These values can be designed according to different applications.

The *ReputationQuerying* function is used to retrieve the reputation associated with a specified account. The function requires an account and returns the reputation value of this account.

The *ThresholdCalculating* function calculates the reputation threshold (see Section 4.3.2). In DRM, the threshold is defined as the average reputation value of all the ledger nodes:

$$\text{threshold} = \frac{\sum_{i=1}^{N} R_i}{N} \tag{5}$$

where N is the number of ledger nodes.

4.2.3 The Workflow of Application and Contract Layer

Given DAC and RMC, we are ready to elaborate on the off-chain process of the proposed DRM mechanism. As shown in Fig. 8, the off-chain process of DRM includes the following steps:

Step 1: Applications enter the user's account and data into DAC.

Step 2: This application deploys DAC on the blockchain and broadcasts the contract address.

Step 3: Upon receiving this address, the evaluation module of the application layer assigns it to one of the reviewers.

Step 4: The evaluation module notifies the validators to vote.

Step 5: The validators call either the *Accept* or *Decline* function according to the scoring rules. If most validators believe the user score in DAC is reasonable or unreasonable, the DAC is stored on the blockchain. If this DAC is reasonable, perform step 6; otherwise, perform step 3.

Step 6: RMC calculates the reputation values of the participants from (2), (3), and (4).

Step 7: The reputation values of the participants are updated according to (1).

An extra step: Users, reviewers, or validators can use DAC as evidence to appeal incorrect results because it is a verifiable evaluation and permanently stored on the blockchain.

4.3 Consensus Implementation

The rPoA consensus consists of election and consensus phases. First, we present the election phase through two algorithms in Section 4.3.1. Then, we show the consensus phases in Section 4.3.2. For ease of exposition, we introduce two vectors: *SignerAccounts* and *RecentAccounts*. The *SignerAccounts* vector stores all the consensus nodes' accounts, which are configured by the administrator in the

configuration file (such as genesis.json) of the permissioned chain ahead of the genesis block. Later, this vector can be modified during the consensus phase. The *RecentAccounts* vector represents the accounts of the consensus nodes that have proposed blocks in the last $\frac{|\text{signers}|}{2} + 1$ rounds, which is used to prevent consecutive block proposal by the same consensus node. Each ledger node maintains these data structures locally.

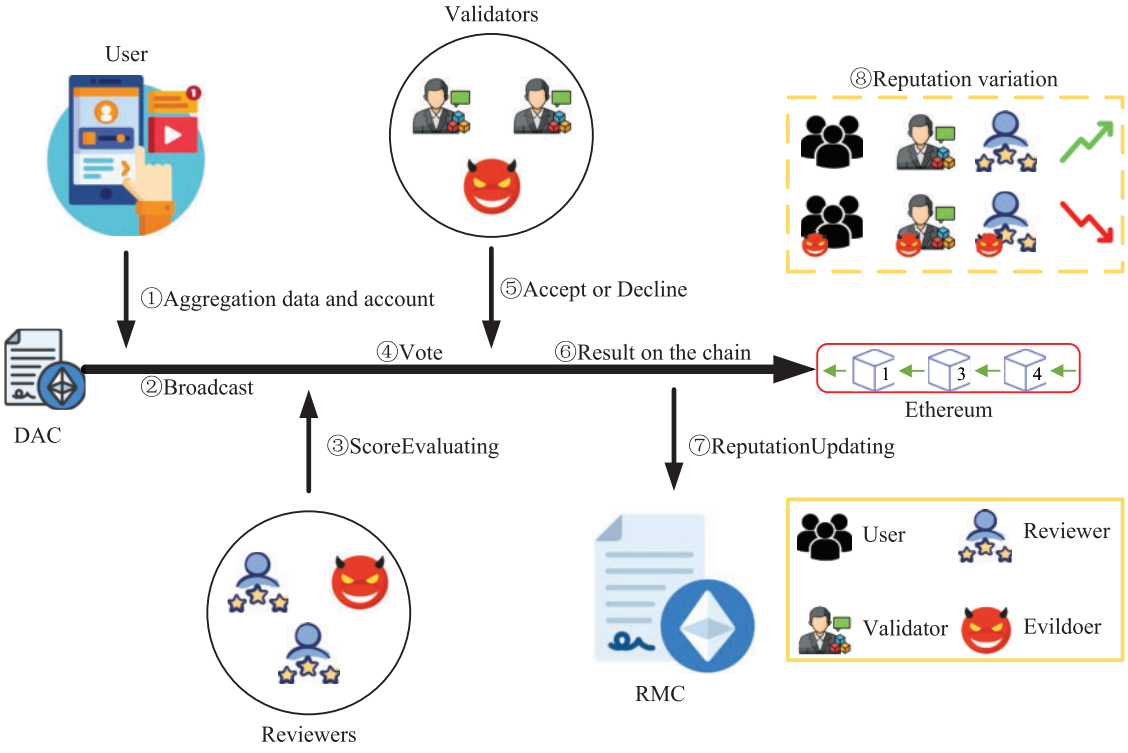


Figure 8: The workflow between application and contract layers

4.3.1 Election Phase

At the beginning of each epoch, the generated block is a checkpoint block, which differs from ordinary blocks in two ways. First, the block's state changes to the *ChangeAuthorization* state, notifying all ledger nodes to enter the Election Phase. Second, the block carries the accounts of ledger nodes with reputation values greater than the reputation threshold (i.e., the candidates for consensus nodes). Algorithm 4.1 describes the process of checkpoint block proposal. In the election phase, the *CheckpointBlockProposal* function uses the *ThresholdCalculating* function to calculate the threshold as a baseline reputation value for the ledger node to become a consensus node. The reputation value of all ledger nodes is compared to this threshold, and nodes that exceed it are added to the Signers vector. This vector is included in the *Extra* attribute of the block structure, and *ChangeAuthorization* is set in the Nonce attribute.

Algorithm 4.1.

```
Procedure CHECKPOINTBLOCKPROPOSAL(BlockNumber,LedgerNodes)
    if BlockNumber mod Epoch = 0 then
        Threshold ← ThresholdCalculating()
        Signers ← ∅
        for node_i ∈ LedgerNodes do
            if ReputationQuerying(node_i) > Threshold then
                Add nodei into Signers
            end if
        end for
        if Signer ≠ ∅ then
            Block_Extra ← Signers
            Block_Nonce ← ChangeAuthorization
        end if
    end if
    return Checkpoint Block
end Procedure
```

Upon receiving the checkpoint block, ledger nodes check its validity as shown in Algorithm 4.2. When this block is accepted and verified by all the ledger nodes, it implies that the election phase terminates. In the next epoch, new consensus nodes will generate the block.

Algorithm 4.2.

```
procedure VERIFYCHECKPOINTBLOCK(CheckpointBlock)
    if Block_Nonce = ChangeAuthorization&Block_Number mod Epoch = 0 then
        Threshold ← ThresholdCalculating()
        Signers ← Block_Extra
        for node_i ∈ Signers do
            if ReputationQuerying(node_i) > Threshold then
                return error
            end if
        end for
        SignerAccounts ← Signers
        RecentAccounts ← ∅
    end if
    return SignerAccounts
end Procedure
```

4.3.2 Consensus Phase

During the consensus phase, each consensus node generates and broadcasts a block. Algorithm 4.3 shows the *BlockProposal* process. The process starts by verifying the node's block proposal. Then, the *ReputationQuerying* function is used to query the reputation value of the node and fill it in the *Difficulty* field of the block header. Let R_{max} denote the maximum reputation value among all the ledger nodes in the current round.

Algorithm 4.3.

procedure BLOCKPROPOSAL(*LedgerNodeAccount, Rmax*)
    $Coinbase \leftarrow LedgerNodeAccount$
    if $Coinbase \notin SignerAccounts$ **then**
        return error
    end if
    if $Coinbase \in RecentAccounts$ **then**
        return error
    end if
    othercode
    $Block_{Coinbase} \leftarrow Coinbase$
    $Block_{Difficulty} \leftarrow ReputationQuerying(Coinbase)$
    othercode
    if $Block_{Difficulty} < R_{max}$ **then**
        $Broadcasts = RoundTime + FaxTime$
    end if
    Add coinbase into RecentAccounts
    Broadcasts Block
end procedure

Upon receiving a new block, ledger nodes verify it as follows. First, the *BlockVerifying* algorithm verifies the node's eligibility for block proposal. Then, the algorithm judges if the block's reputation value is the same as its proposer by querying RMC. Finally, as long as the verification is completed, the qualified block is appended to the blockchain, and the unqualified block is discarded. The verification steps are depicted in Algorithm 4.4.

Algorithm 4.4.

procedure BLOCKVERIFYING(*Block*)
    if $Block_{Coinbase} \notin SignerAccounts$ **then**
        return error
    end if
    if $Block_{Coinbase} \in RecentAccounts$ **then**
        return error
    end if
    if $Block_{Difficulty} \neq ReputationQuerying(Block_{Coinbase})$ **then**
        return error
    end if
    Add Block into Blockchain
end procedure

5 Experimental Results

In this section, experimental results show the performance of DRM from the aspects of throughput, decentralization, and security. The experiments are conducted on a computer running on the Windows 11 Pro operating system and with a 2.1 GHz Intel Core i7-12700 processor, 16 GB of DDR4 RAM, and 256 GB of NVMe SSD.

5.1 Throughput Performance

Fig. 9 compares the transaction throughput (i.e., transactions per second) between the proposed rPoA consensus and Clique over different benchmarking transaction throughputs provided by Caliper v0.5.0 [33]. Caliper is a standard benchmarking tool used to assess the performance of various blockchain solutions with predefined use cases, such as benchmark transaction throughputs in Fig. 9. Specifically, we use Caliper to create and send transaction requests to rPoA from 10 to 100 tx/s and test the number of transactions that rPoA and Clique can handle, respectively. In this figure, we simulate 4 ledger nodes that perform the consensus phase every 100 rounds and configure a block proposal time of 10 s. It is observed that the throughput of rPoA is very close to that of Clique across various benchmarking transaction throughputs.

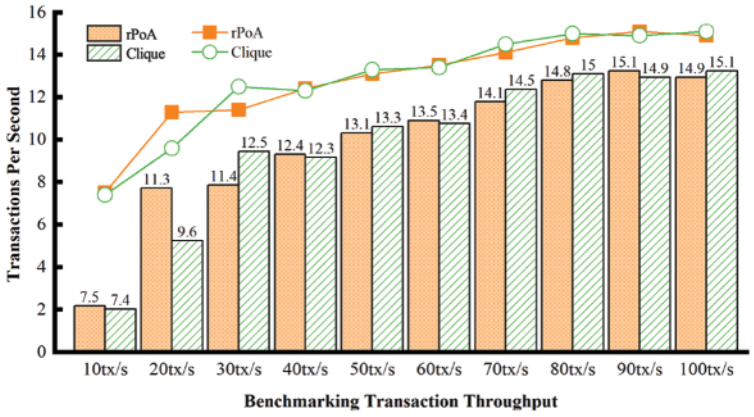


Figure 9: Transaction throughput comparison between rPoA and clique

5.2 Security Performance

In this part, we consider potential attacks from different participants in DRM:

- *Malicious Users*: A user may generate harmful or false data, which inevitably degrades the trustworthiness of the off-chain reputation evaluation.
- *Malicious Reviewers*: A reviewer may provide an unreasonable score for the user.
- *Malicious Validators*: A validator may be incentivized to "stand on both sides" by agreeing with and disagreeing with the score given by the reviewer to the user, and the validator may also exhibit biased behavior or make arbitrary judgments.

Moreover, our experiments build upon the following underlying assumptions:

1. Most nodes actively uphold their reputation.
2. Data submissions across different layers are reliable.

According to Section 4.2.3, we consider a simple application to generate random data as sources of reputation for the user. Particularly, we set a specific evaluation rule for the reviewer to calculate the hash value of the random data and count the number of zeros in the hash. That is, the value of a in (2) is the number of zeros within the hash, and the values of b, c, d, and e in (3) and (4) are 5, 3, 2, and 1, respectively. In the experiments, we simulate 200 rounds of consensus among 8 ledger nodes. In each round, each ledger node submits data for scoring, and a single node from the remaining 7 nodes is selected by equal probability to be the reviewer, while the remaining 6 nodes act as validators to verify the scores given by the reviewer.

Fig. 10 shows the variation in the reputation values of all nodes in DRM over 200 rounds. In this figure, an honest node (e.g., node 1) turns into a malicious node at round 100. First, it is observed

that the reputation value of each honest node gradually increases as the number of rounds goes up. Second, the reputation value of node 1 degrades immediately once it becomes malicious. This is due to the fact that the scores and votes submitted by any evildoer are rejected by the honest nodes.

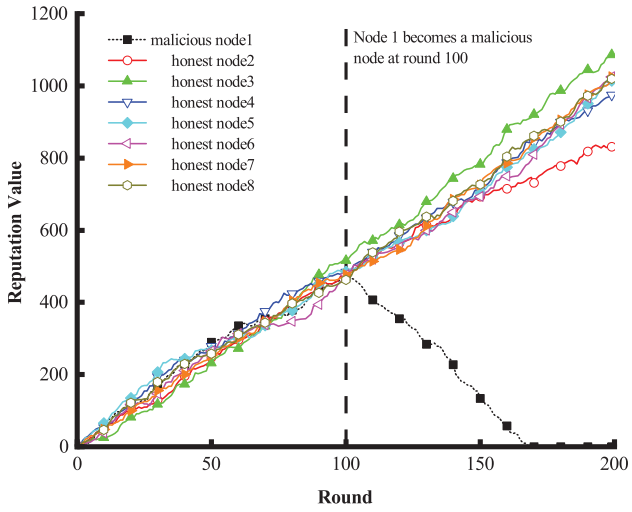


Figure 10: The variation of reputation values of malicious and honest nodes in DRM

5.3 Decentralization Performance

Fig. 11 shows the variation in the reputation values of 8 ledger nodes concerning the reputation thresholds over 200 rounds. First, it is observed that nodes 1, 2, 3, 4, and 5 are all the consensus nodes at round 100 since their reputation values are beyond the threshold. Second, the node with the largest reputation value for a particular round (e.g., node 5 at round 100) proposes the block, and this block will be accepted by the others if node 5 has not proposed any blocks in the last 3 rounds.

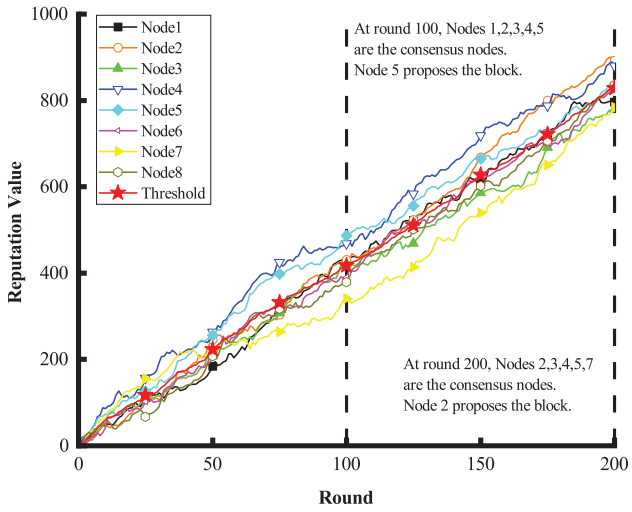


Figure 11: The variation of reputation values with reference to the reputation thresholds in DRM

6 Conclusion

In this paper, we have proposed a decentralized reputation management mechanism for permissioned blockchain networks. In this mechanism, we first designed an off-chain reputation evaluation, in which the user is evaluated by a reviewer according to their generated data, and the evaluation is stored on the blockchain. We then developed an on-chain reputation-aided consensus to enhance security without sacrificing throughput.

Several interesting directions follow this work. First, this paper shows that verifiable evaluations are stored on the blockchain as smart contracts. It is of interest to extend the storage capacity of DRM through decentralized storage (such as IPFS). Second, the rPoA consensus retains transaction throughput compared with traditional PoA. It is worth studying to improve the throughput of rPoA through sharding in the future. In this context, how to design the sharding-aided rPoA with the aid of DRM remains challenging.

Acknowledgement: The authors would like to express their gratitude to the anonymous reviewers and journal editors for their valuable assistance in improving the logical organization and content quality of this paper.

Funding Statement: This research is supported by the Shenzhen Science and Technology Program under Grants KCXST20221021111404010, JSGG20220831103400002, JSGGKQTD2022110111565 5027, JCYJ 20210324094609027, the National Key R&D Program of China under Grant 2021YFB270 0900, the National Natural Science Foundation of China under Grants 62371239, 62376074, 72301083, the Jiangsu Specially-Appointed Professor Program 2021.

Author Contributions: The authors' contributions to the paper are as follows: study conception and design: Jinyu Chen, Long Shi, Taotao Wang; methodology: Jinyu Chen, Long Shi, Taotao Wang, Daojing He; analysis and interpretation of results: Jinyu Chen, Long Shi, Qisheng Huang; supervision: Daojing He; draft manuscript preparation: Jinyu Chen, Long Shi, Qisheng Huang, Daojing He. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Not applicable.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf (accessed on 13/07/2023).
2. King, S., Nadal, S. (2012). PPCoin: Peer-to-peer crypto-currency with proof-of-stake. *Self-Published Paper, August, 19(1)*.
3. The beacon chain Ethereum 2.0 explainer you need to read first. https://ethos.dev/beacon-chain (accessed on 09/07/2023).
4. Wood, G. (2014). Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper, 151,* 1–32.
5. Bhattacharya, S., Victor, N., Chengoden, R., Ramalingam, M., Selvi, G. C. et al. (2022). Blockchain for Internet of Underwater Things: State-of-the-art, applications, challenges, and future directions. *Sustainability, 14(23),* 15659.

6. Wang, H., Wang, T., Shi, L., Liu, N., Zhang, S. (2023). A blockchain-empowered framework for decentralized trust management in Internet of Battlefield Things. *Computer Networks, 237,* 110048.
7. Mollah, M. B., Zhao, J., Niyato, D., Guan, Y. L., Yuen, C. et al. (2020). Blockchain for the Internet of Vehicles towards intelligent transportation systems: A survey. *IEEE Internet of Things Journal, 8(6),* 4157–4185.
8. Wang, M., Zhu, T., Zuo, X., Yang, M., Yu, S. et al. (2023). Differentially private crowdsourcing with the public and private blockchain. *IEEE Internet of Things Journal, 10(10),* 8918–8930.
9. Castro, M., Liskov, B. (2002). Practical Byzantine fault tolerance and proactive recovery. *ACM Transactions on Computer Systems, 20(4),* 398–461.
10. ProofofAuthority. https://github.com/paritytech/parity/wiki/Proof-of-Authority-Chains (accessed on 11/07/2023).
11. Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K. et al. (2018). Hyperledger fabric: A distributed operating system for permissioned blockchains. *Proceedings of the Thirteenth EuroSys Conference*, New York, USA.
12. FISCO BCOS. http://www.fisco-bcos.org (accessed on 14/07/2023).
13. Goerli testnet. https://goerli.net/ (accessed on 13/07/2023).
14. POA Networks. https://www.poa.network (accessed on 16/07/2023).
15. Aponte-Novoa, F. A., Orozco, A. L. S., Villanueva-Polanco, R., Wightman, P. (2021). The 51% attack on blockchains: A mining behavior study. *IEEE Access, 9,* 140549–140564.
16. Vechain whitepaper. https://www.vechain.org/whitepaper/ (accessed on 19/07/2023).
17. Çulha, D. (2022). A random and scalable blockchain consensus mechanism. *Asian Journal for Convergence in Technology, 8(1),* 47–55.
18. VeChain. https://www.vechain.org (accessed on 03/07/2023).
19. Parity technologies. https://www.parity.io (accessed on 03/07/2023).
20. Clique PoA protocol and Rinkeby PoA testnet. Available at: https://github.com/ethereum/EIPs/issues/225 (accessed on 01/07/2023).
21. Islam, M. M., Merlec, M. M., In, H. P. (2022). A comparative analysis of proof-of-authority consensus algorithms: Aura vs clique. *2022 IEEE International Conference on Services Computing (SCC)*, Barcelona, Spain, IEEE.
22. Gao, S., Yu, T., Zhu, J., Cai, W. (2019). T-PBFT: An EigenTrust-based practical Byzantine fault tolerance consensus algorithm. *China Communications, 16(12),* 111–123.
23. Yang, Z., Yang, K., Lei, L., Zheng, K., Leung, V. C. (2018). Blockchain-based decentralized trust management in vehicular networks. *IEEE Internet of Things Journal, 6(2),* 1495–1505.
24. Shi, L., Wang, T., Li, J., Zhang, S., Guo, S. (2023). Pooling is not favorable: Decentralize mining power of pow blockchain using age-of-work. *IEEE Transactions on Cloud Computing, 11(3),* 2756–2769.
25. Xu, Y., Xiao, S., Wang, H., Zhang, C., Ni, Z. et al. (2023). Redactable blockchain-based secure and accountable data management. *IEEE Transactions on Network and Service Management*. https://doi.org/10.1109/TNSM.2023.3255265
26. Li, M., Weng, J., Yang, A., Lu, W., Zhang, Y. et al. (2018). CrowdBC: A blockchain-based decentralized framework for crowdsourcing. *IEEE Transactions on Parallel and Distributed Systems, 30(6),* 1251–1266.
27. Wang, T., Shi, L., Wang, J. H., Wang, Z., Deng, X. et al. (2022). Building trust via blockchain in UAV-assisted ultra-dense 6G cellular networks. *IET Blockchain, 2(3–4),* 67–76.
28. Xu, Y., Li, Q., Zhang, C., Tan, Y., Zhang, P. et al. (2023). A decentralized trust management mechanism for crowdfunding. *Information Sciences, 638,* 118969.
29. Li, J., Shao, Y., Wei, K., Ding, M., Ma, C. et al. (2021). Blockchain assisted decentralized federated learning (BLADE-FL): Performance analysis and resource allocation. *IEEE Transactions on Parallel and Distributed Systems, 33(10),* 2401–2415.

30. Gao, M., Shen, R., Shi, L., Qi, W., Li, J. et al. (2023). Task partitioning and offloading in DNN-task enabled mobile edge computing networks. *IEEE Transactions on Mobile Computing, 22(4),* 2435–2445.
31. Buterin, V., Hernandez, D., Kamphefner, T., Pham, K., Qiao, Z. et al. (2020). Combining ghost and casper. arXiv preprint arXiv:2003.03052.
32. ETHEREUM VIRTUAL MACHINE (EVM). https://ethereum.org/en/developers/docs/evm/ (accessed on 11/07/2023).
33. Caliper. https://github.com/hyperledger/caliper (accessed on 13/07/2023).