IEEEAccess

Delegated Proof of Stake With Downgrade: A Secure and Efficient Blockchain Consensus Algorithm With Downgrade Mechanism

FAN YANG[1], **WEI ZHOU**[1], **QINGQING WU**[1], **RUI LONG**[1],
NEAL N. XIONG[2], **(Senior Member, IEEE), AND MEIQI ZHOU**[1]
[1]School of Information and Safety Engineering, Zhongnan University of Economics and Law, Wuhan 430073, China
[2]College of Intelligence and Computing, Tianjin University, Tianjin 300073, China

Corresponding author: Rui Long (mr_longr@163.com)

ABSTRACT Blockchain technology has a wide range of applications in the fields of finance, credit reporting and intellectual property, etc. As the core of blockchain, consensus algorithm affects the security and performance of blockchain system directly. In the past 10 years, there have been about 30 consensus algorithms such as Proof of Work (PoW), Proof of Stake (PoS), Delegated Proof of Stake (DPoS), Ripple Protocol Consensus Algorithm (RPCA) and AlgoRand. But their security, stability and operating efficiency still lag far behind our actual needs. This paper introduces the computing power competition of PoW into DPoS to design an improved consensus algorithm named Delegated Proof of Stake with Downgrade (DDPoS). Through the further modification, the impact of both computing resources and stakes on generating blocks is reduced to achieve higher efficiency, fairness, and decentralization in consensus process. Then a downgrade mechanism is proposed to quickly replace the malicious nodes to improve the security. The simulation experiments in blockchain system show that the proposed consensus algorithm is significantly more efficient than PoW and PoS, but slightly lower than DPoS. However, its degree of centralization remains far below that of DPoS. And through the downgrade mechanism, the proposed consensus algorithm can detect and downgrade the malicious nodes timely to ensure the security and good operation of system.

INDEX TERMS Blockchain, consensus algorithm, delegated proof of stake with downgrade, downgrade mechanism, efficiency, fairness, decentralization.

I. INTRODUCTION

On 1 Nov. 2008, Satoshi Nakamoto has published a paper [1] on www.metzdowd.com, which not only marked the birth of Bitcoin but also showed people the blockchain. Blockchain is a data structure that is composed of data blocks in a manner similar to a linked list, using a distributed ledger with cryptography techniques to ensure the authenticity and security of transactions information. With the continuous development of blockchain technology, the blockchain can also play a good role in Artificial Intelligence [2], [3], Internet of Things [4]–[6], Education Reform [7], [8] and some other fields [9]–[12].

The associate editor coordinating the review of this article and approving it for publication was Luis Javier Garcia Villalba.

Consensus algorithm is the most important factor of the entire blockchain system, for the reason that its efficiency determines the blockchain's performance directly. With the continuous development of blockchain technology, the consensus algorithm is constantly adapting to the emerging requirements from the earliest Proof of Work (PoW) [1] to the later Proof of Stake (PoS) [13], Delegated Proof of Stake (DPoS) [14], Practical Byzantine Fault Tolerance (PBFT) [15] and some other improved consensus algorithms such as Proof of Burn (PoB) [16], Proof of Activity (PoA) [17], Proof of Luck (PoL) [18], and Stellar Consensus Protocol) (SCP)[19]. However, none of them is perfect.

In PoW, each node competes for the opportunity to generate blocks through owning more computing resources than others, that not only greatly consumes the computing power

resources but also leads to the inefficiency about only one block every 10min [1]. And nowadays it is no longer possible for an ordinary miner to get rewards from the Bitcoin network with a single device or a small amount of computing equipments because of the appearance of the Mine Pool, which can own most of the computing power. In order to reduce the waste of computing resources in the PoW, PoS uses stakes in competition for the opportunity of generating blocks [13]. However, when a node holds a large number of stakes for a long time, its probability of calculating the nonce value is nearly 100%. Although the time that generates a block is reduced to only 64s [13], PoS still relies on computing power and also needs to waste a lot of computing resources. To break the cycle of the competition in computing resources to generate blocks, DPoS introduces a voting mechanism based on PoS, and lower the time cost of generating a block to 3s [20]. DPoS divides the nodes in the blockchain system into three categories: witnesses, delegates, and workers. Witnesses are the core of the entire system and they were elected through resource voting by all nodes. The nodes winning the top-N number of votes become the witnesses and take turns to generate blocks. While they will not be paid, the delegates can initiate a request for updating the blockchain. Workers have the right to propose new projects and get reward from the elected projects voted for. However, for the reason that the witness will not be disqualified unless there are special reasons in the consensus process, they will be witnesses and hold the right of generating blocks for a long time, which leads to a security risk due to the higher degree of centralization of DPoS.

Moreover, in any blockchain, there may exist malicious nodes. The so-called malicious nodes are those that illegally violate the trusted consensus mechanism, tamper with transaction information, cause network congestion, and disrupt the normal operation of the network. As a result, the blockchain system may become insecure, unreliable, and inefficient.

To resolve above problems, we propose an improved consensus that introduces PoW's idea to improve fairness, and DPoS' idea to reduce the resource consumption and improve consensus efficiency of the blockchain system.

The key innovations of this improved consensus algorithm are:

- Using the PoW, the blockchain selects a set of nodes which own enough computational power to participate in the following election and block generation;
- Each node has only one vote for randomly voting, so that the impact of stakes on consensus nodes election can be decreased;
- A downgrade mechanism is applied to quickly downgrade the malicious nodes and upgrade the reliable nodes to maintain the security and good operation of the system.

The rest of this paper is organized as follows: Section 2 describes the concept, architecture, consensus algorithms, and some other core technologies of blockchain. Section 3 introduces the optimization idea and the process of the improved DDPoS algorithm, then describes it by pseudocode. Section 4 simulates the DDPoS algorithm and analyzes the experimental results. Section 5 concludes the paper and highlights the future directions.

II. RELATED WORK
A. BLOCKCHAIN

The blockchain system is built on a computer network and requires a computer network as a channel for transmitting information. The infrastructure of the blockchain system is divided into six layers: data layer, network layer, consensus layer, incentive layer, contract layer, and application layer. The blockchain infrastructure model [21] is shown in Fig.1. In the traditional blockchain system, each node records a unified and complete account of the whole network transactions. Each record has a timestamp and a unique cryptographic signature [22].
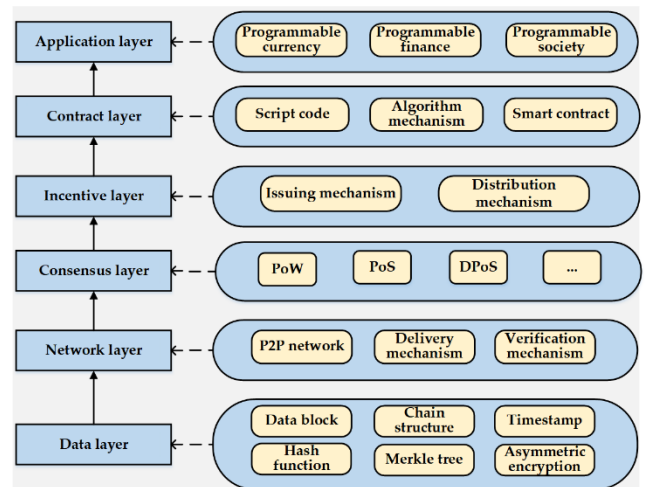

FIGURE 1. Blockchain infrastructure model.

The block is mainly composed of a block header and a block body. The block header includes a version number, a hash value of the previous block, a Merkel tree root of transactions, a timestamp, a current difficulty value, and a random number. The information about transactions are regularly stored in the block body. A distributed ledger based on blockchain allows both parties to effectively record transactions and verify the transaction without a trusted third party.

The consensus mechanism is a collaborative process algorithm that specifies how consensus is reached among all consensus nodes and identifies the validity of records. Members of the blockchain use it to negotiate whether the transaction is valid and make the account stay in sync. Each consensus node in the blockchain verifies and confirms the data according to the algorithm. After confirmed by a certain number of nodes, the valid data can be written into the blockchain [23]. At present, the most common consensus algorithms are: PoW, PoS, DPoS and PBFT. From the emergence of Bitcoin to today, there are more than 30 consensus algorithms, most of which are based on the above four consensus algorithms.

Smart Contract is a set of commitments defined in digital form [24]. Now, it represents a computer agreement designed in a full life cycle that includes three parts: contract generation, contract release, and contract execution [25]. Particularly, blockchain based smart contracts include transaction processing, preservation mechanisms, and a complete state machine. The contract that meets the trigger conditions will be added into the contract queue to be verified through periodical traversal of the state and trigger conditions. After being verified successfully, the contract will be executed.

In order to establish a reliable distributed ledger, the blockchain system adopts the P2P (i.e., Peer to Peer) network [26] and public key cryptography [27]. The public-key cryptography involves two keys: a public key that is publicly disclosed and a private key that is privately secret. The two keys are mathematically related, but if you know one of them, you can not calculate another one.

B. CONSENSUS ALGORITHM

Proof of Work (PoW) mechanism, first used for spam filtering, now is applied to achieve the consistency of node data in Bitcoin. And if a participant wants to generate and write a new block into the blockchain in the Bitcoin system, it must solve the puzzle of proof-of-work given by the blockchain network. PoS was firstly implemented by Sunny King's Peercoin and its mining difficult is adjusted based on the number of stakes held by workers. Simply put, the more stakes you have, the easier it is to generate the block. Larimer designed the Delegated Proof of Stake (DPoS) and implemented it for the first time in its BitShares project [28]. The DPoS consensus process is divided into the earlier process of electing witness (i.e., block producers) and the later process of generating blocks. The witnesses are only responsible for witnessing the transaction, verifying the signature and timestamping the transaction, but not participating in the trading. They generate one block every 3s in turn, and if a witness did not complete the task at the specified time, it will be skipped and replaced by the next one. Each node on the network can vote for its own trusted witness, the more blockchain stakes he has, the higher possibility of him to be a witness. However, due to the mechanism that each witness node takes turns generating blocks, the identity of the witness is already known and always constant, which would make the blockchain system more vulnerable to collusion attacks. PBFT was proposed by Castro and Liskov in a paper published in 1999 to solve the problem of General Byzantine originally [29]. It works in the principle of state machine replication [30], ensuring the normal operation of the C/S system under the condition that the Byzantine replicas are less than 1/3 of all replicas.

In 2013, Ongaro and Ousterhout proposed the Raft [31] consensus algorithm, which is used by Quorum [32] to tolerate crashes. Raft was easier to be understood and utilized in practical systems than previous algorithms. Ripple, implemented in 2004 by Fugger, was detailed by Buterin on bitcoinmagazine.com in Feb. 2013. And in 2014, Schwartz *et al.* proposed the Ripple Protocol Consensus Algorithm (RPCA) [33] to solve the high communication latency in asynchronous network. In Ripple network, each node has its own unique node list (UNL), which includes its trusted nodes. Slimcoin [16], released in May 2014, draws on the design of Bitcoin and PPcoin, and proposes the PoB consensus algorithm based on PoW and PoS. And soon after, the PoA [17] consensus, in which some part of the tokens dug by PoW are distributed to all active nodes in a lottery, was also proposed. In PoA the number of stakes the node owns is proportional to the number of lottery tickets (i.e., probability of being drawn) it has. A major breakthrough in Tendermint [34] proposed in 2014 was the implementation of the first PBFT-based PoS consensus algorithm. In 2015, Professor Mazieres, Chief Scientific Officer of Stellar.org, proposed the SCP [19]. Based on the federal Byzantine agreement and the Ripple protocol, the SCP is the first provable security consensus mechanism with four key attributes: decentralized control, low latency, flexible trust, and progressive security. In 2016, Micali proposed a fast Byzantine fault-tolerant consensus algorithm called AlgoRand [35]. The blockchain applying this algorithm uses the password lottery technique to select the verifier and leader of the consensus process, and reaches a consensus on the new block through its designed BA∗ Byzantine fault-tolerant protocol. The Elastico [36] consensus mechanism proposed in 2016,which is the first Byzantine fault-tolerant security sharding protocol, has enhanced the scalability of the blockchain. The HoneyBadger [37] consensus proposed in 2016 is the first practical asynchronous Byzantine fault-tolerant consensus protocol that guarantees the liveness of the blockchain system without any network time assumptions. The 2-hop consensus [38] was proposed in Apr. 2017 and was originally designed to address the potential 51% computing power attack problem of PoW by introducing PoS's stakes. The security of 2-hop consensus is based on the fact that honest nodes occupy most of the joint resources. Ren proposed the Proof of Stake Velocity (PoSV) consensus algorithm in the white paper of Reddcoin [39], which is aimed at dealing with the problem that the coin age is a linear function of time in PoS, and is committed to decreasing the coin arbitrage. The Proof of Useful Work (PoUW) [40] proposed in 2017 is a consensus algorithm for solving the energy consumption problem of PoW. It transforms the meaningless SHA256 hash in the PoW consensus into difficult and valuable operations in real-world scenarios, such as computing orthogonal vectors, 3-SUM problems, and shortest path problems. The Ouroboros Consensus [41], introduced in August 2017, has proposed a new reward mechanism to drive the PoS consensus process, so that the behavior of honest nodes constitutes an approximate nano-equalization.

In China, one of the most well-known consensus algorithms is the delegated Byzantine Fault Tolerance (dBFT) of the NEO team used in its NEO (i.e., formerly AntShares) project [42]. The algorithm improves the election process in DPoS and consensus process in PBFT, and can tolerate up to 33% of faulty nodes. In 2017, Xiaofei Liu *et al.* improved the traditional PBFT consensus algorithm and proposed a

TABLE 1. Performance of PoW, PoS, and DPoS.

Consensus algorithms	PoW [1]	PoS [13]	DPoS [20]
Basis for assigning accounting rights	Computing power	Stake	Stake votes
Threat to security	Concentration of power	Lack of active nodes	Destruction of the witnesses
Resource consumption	The highest	Lower than PoW, higher than DPoS	The lowest
Average time to generate blocks	10min	64s	3s
Typical application	Bitcoin, Ethereum	Peercoin	Bitshare
Fairness	Relatively fair	Relatively unfair	Relatively unfair
Scalability	Good	Good	Good

TABLE 2. Performance of the existed improved consensus algorithms.

Other improved consensus algorithms	Advantages	Disadvantages
Raft [31]	More practical and easier to be understood	Lower security
RPCA [33]	A Low-latency and high-robust Byzantine fault-tolerance consensus	Limited the application areas
PoB [16]	Increased the efficiency	Lower practicality and security
PoA [17]	Increased the fairness and efficiency	Lower security
Tendermint [34]	Resisted double-spending attack and achieved the Byzantine fault-tolerance	Lower throughput
SCP [19]	A proven security consensus	Higher communication latency
AlgoRand [35]	An efficient Byzantine fault-tolerant algorithm	Lack of incentives
Elastico [36]	Increased the scalability of Byzantine fault-tolerant algorithm	Limited the application areas
HoneyBadger [37]	Increased the throughput	Lower scalability
The 2-hop consensus [38]	Resisted the 51%-power attack	Higher centralization and Resource consumption
PoSV [39]	Promoted financial flows and decentralized accounting right	Higher centralization
PoUW [40]	More practical	Lower security
Ouroboros [41]	Increased the security	More complicated
dBFT [42]	Increased the throughput and security	Lower scalability and more complicated
DDBFT [43]	Improved the throughput and scalability	Imperfect node scoring rules

dynamic authorization PBFT called DDBFT [43], which deployed in the P2P network topology instead of the classic C/S architecture of the PBFT. The replica nodes participating in the consensus algorithm can join and exit the network at any time. In combination with the mechanism in DPoS, a voting mechanism based on the holder's stakes is also designed in DDBFT.

The performance analysis of some representative blockchain consensus algorithms we mentioned above can be seen in Table 1 and Table 2.

From the analysis above, we know, PoW competes to generate blocks through owning more computing resources than others. And PoS is an improved version of PoW, based on the assumption that the people with more stakes will not attack the network. Although PoS is more energy saving and efficient than PoW, the assignment mechanism to generate blocks is extremely unfair, for the reason that the assignment

in PoS is based on the number of stakes the node owning [44]. The DPoS mechanism further cut down the number of nodes which generate blocks to 101. Under the premise of ensuring network security, the energy consumption of the entire network is further reduced, and the network operating cost is the lowest. However, the voting enthusiasm of nodes in DPoS is not high, because a small number of vote holders occupy the vast majority of voting rights, resulting in the low influence from ordinary nodes even if they participate in voting. According to statistics in [20], more than 90% of shareholders have never participated in the vote. This has led to that the witnesses selected by a few votes' holders becoming more and more fixed (i.e., as long as the witness does not act as a malicious node). And there are also a lot of security risks in the blockchain system because of the difficulty in dealing with malicious nodes [45]. The three consensus algorithms of PoW, PoS, and DPoS are widely used in the application of

public blockchains. PBFT is mostly used in the consortium blockchain with a fixed number of nodes, but not applicable in a public blockchain with a large number of nodes because of its poor scalability. As mentioned above, some later PoW-based improved consensus algorithms such as PoUW try to make good use of computing power. Some PoS-based consensus algorithms such as Tendermint, Ouroboros, and HoneyBadger attempt to solve the "Nothing at Stake" problem in the traditional PoS. Some PBFT-based consensus algorithms such as SCP, Algorand, and Elastico aim to enhance the scalability or security of the blockchain. Some "PoW+PoS" consensus algorithms such as PoA, PoB, PoSV, and the 2-hop consensus alleviate the energy consumption and security problems of PoW and PoS. However, although these improved consensus algorithms have a better performance in terms of security, scalability or power reduction of the blockchain, their efficiency of generating blocks are still lag far behind the DPoS consensus algorithm. Therefore, under the premise of ensuring safety and fairness, we introduce the DPoS into our improved algorithm to enhance the efficiency of the blockchain. We combine the PoW's computing competition with the DPoS's mechanism, for the reason that PoW's competitive strategy can improve fairness while the DPoS's voting idea can reduce the resource consumption and improve the efficiency. We divide the consensus process into two steps to reduce the impact of stakes on node's right of generating blocks. Then we design a downgrade mechanism to detect and downgrade the malicious nodes to ensure blockchain system security. Finally, we use the Java language to simulate the blockchain node and operate the blockchain system based on the DDPoS mechanism.

III. THE PROPOSED IMPROVED DDPoS
A. THE MAIN IDEA OF DDPoS ALGORITHM
According to the above analysis, it is found that the biggest problem the public blockchain consensus algorithm facing is that the blockchain is becoming more and more centralized and insecure, and its efficiency of generating blocks is still far behind our actual needs. Therefore, this paper raises the following questions:

- How to achieve high efficiency of the blockchain system and reduce the resource consumption?
- How to assign the right of generating blocks fairly and ensure that most nodes can participate in the consensus so as to avoid overcentralization and reduce the risk of collusion attack?
- How to detect and stop the witness nodes' malicious behavior in the DPoS consensus algorithm?

Table 1 and previous analysis showed that the DPoS authorization idea can solve the first problem, because the DPoS consensus mechanism can greatly improve the consensus efficiency and greatly reduce resource consumption. At the same time, PoW's power competition can deal with the second problem well, as the idea of PoW can ensure that everyone has the right to generate blocks as much as possible.

However, it is difficult to achieve fairness through simply using DPoS, which will lead to that only those who have more resources can become electors. And when simply using the PoW, the block interval is about 10min, and it would waste a large amount of computing and electric power resources. For the third problem, this paper introduces a downgrade mechanism similar to that in the modern sports league to descend the level of the witness node which damages the consensus and replace them.

In our research, there are two kinds of nodes in consensus process. The one is the witness node, which generates or verifies blocks and participates in the consensus process. While the other kind is the candidate node, which acts as candidate for the witness node so as to replace the witness node once it fails. And, these two types of nodes are collectively referred to as the consensus nodes.

The main ideas are as follows:

- We divide the consensus process into two steps. In the first step, use the idea of PoW to "screen" a certain number (more than 201) of better nodes from the whole network, and then select 201 consensus nodes by stake voting, in which the top 101 nodes are witness nodes and the remaining nodes are candidates. In the second step, the witness nodes record the transactions in a block sequentially and then broadcast it to all consensus nodes for consensus. If more than 50% of the consensus nodes verify the block successfully, it will be added to the blockchain.
- We introduce a downgrade mechanism to replace the malicious node as soon as it was found. On the one hand, when the witness node is found to be a malicious node, it falls into the candidate nodes set, and the rank of all existing witness nodes are decremented by one. On the other hand, the node with sequence number 1 in the set of candidate nodes is upgraded to the witness nodes set and ranked last in the witness nodes set. while the sequence number of all the remaining candidate nodes are decremented by 1, the malicious node is sorted at the end of the set of candidate nodes.

B. DESIGN OF DDPoS ALGORITHM
In DDPoS, there are 101 nodes with the most votes can be elected as witness nodes, and the witness nodes' shortlist will be updated every 24 hours as the same as in DPoS. If someone is found to have a low rate of generating blocks or malicious behavior, the witness node will lose its credibility and witness identity.

The consensus process of DDPoS algorithm consists of three modules: selecting a certain number of consensus nodes, reaching a consensus on the verification of the block, and downgrading malicious nodes.

The module of selecting consensus nodes: all the nodes in the blockchain system are functionalized in order to assign different tasks to different kinds of nodes, which are mainly divided into consensus nodes (which are composed of witness
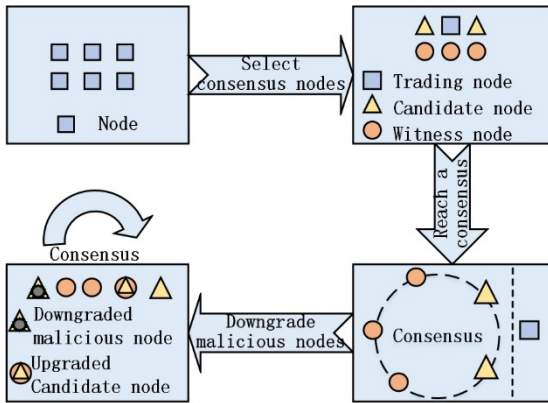
FIGURE 2. State transition of the nodes in a blockchain environment based on the DDPoS algorithm.

nodes and candidate nodes) and trading nodes. The main function of the trading nodes is to generate the transaction while the consensus nodes is to generate and verify the blocks.

The module of reaching a consensus (i.e., consensus module): completing the whole process from the generation to the verification of the blocks.

The module of downgrading malicious nodes: When a malicious node is found, the DDPoS algorithm turns into the module of downgrading the malicious node.

Fig. 2 shows the state transition of the nodes in a blockchain environment based on the DDPoS algorithm. Table 3 describes the symbols appearing in the DDPoS algorithm expression.

1) SELECTION OF CONSENSUS NODES

The blockchain network environment is defined as the P2P network composed of all the nodes in the blockchain system, and all nodes in this environment are divided into two types: consensus nodes and trading nodes. The subnetwork environment composed of consensus nodes is called the consensus network, which will vary with the change of the consensus nodes in the DDPoS algorithm. Meanwhile, the network environment composed of trading nodes is called the trading subnetwork, which is also not static for the reason that every round for consensus nodes selection will also lead to the variation in trading nodes. Trading network will be stable just in the time period from the generation of the current consensus nodes to the start of the next round of consensus node election.

Consensus nodes: the nodes, which involve in the consensus process, rather than generating transactions, comprise two subsets: witness nodes and candidate nodes, as mentioned above.

Trading nodes: the nodes that generate, encrypt and sign its transaction, store blockchain data, but do not participate in the consensus process of the block.

In the blockchain environment, the set of all nodes is expressed as N whose size is k ($k \in N^*$), the number of each node N_i ($i \in [1, k]$) in the set N is $\{1, 2, 3, 4 \ldots k\}$, and

TABLE 3. Definition of the representation symbol in DDPOS algorithm.

| Symbol	Representative meaning
N	The set of all nodes in a blockchain network
N_i	The node numbered with i
N^C	The set of consensus nodes
N^T	The set of trading nodes
N^W	The set of witness nodes
N^A	The set of candidate nodes
N^W_y	The malicious witness node
$PreBlock$	The previous block
$PreBlockHead$	The header of the previous block
$nonce$	The random number
D	Value of difficulty
txs	Trading information
$GOOD$	Good state
$NORMAL$	Normal state (default)
$ERROR$	Abnormal state (a malicious node appears)
PK_{N_i}	Public key of node N_i
SK_{N_i}	Private key of node N_i
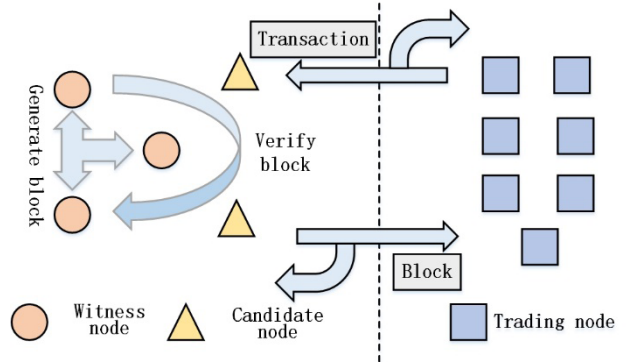
FIGURE 3. Blockchain consensus environment based on DDPoS algorithms.

in brief:

$$\forall N_i, \quad N_i \in N \tag{1}$$

We use N^C to represent the set of all consensus nodes, and its size is l ($l \in N^*$, $l < k$). The symbol N^T represents the trading nodes set with its size (k-l):

$$N = N^C \cup N^T, \quad N^C \cap N^T = \emptyset \tag{2}$$

To the set of all consensus nodes N^C, N^W represents the set of all witness nodes with the size m ($m \in N^*$, $m < l$), and the symbol N^A represents the candidate nodes with its size (l-m):

$$N^C = N^W \cup N^A \tag{3}$$

Hence:

$$\begin{cases} N_i \in N, & i \in [1, k] \\ N_i \in N^C, & i \in [1, l] \\ N_i \in N^T, & i \in [1, (k\text{-}l)] \\ N_i \in N^W, & i \in [1, m] \\ N_i \in N^A, & i \in [1, (l\text{-}m)] \end{cases} \tag{4}$$

The process of the consensus nodes selection is as following:

Step 1: The blockchain system broadcasts message $\langle nonce, D \rangle$ to every node in the entire network.

Step 2: After receiving the message $\langle nonce, D \rangle$, node N_i ($i \in [1, k]$) obtains the previous block header *PreBlockHead* from the blockchain and calculates the value of $HASH(HASH(PreBlockHead), nonce)$.

Step 3: Node N_i observes the value of the comparison equation $HASH(HASH(PreBlockHead), nonce) < D$, if the value is *false*, then $nonce = nonce + 1$ and Step2 is repeated. Else if it's *true*, then N_i broadcasts $\langle HASH(HASH(PREBlockHead), nonce), N_i \rangle$.

Step 4: After receiving l verified information messages of $\langle HASH(HASH(PREBlockHead), nonce), N_i \rangle$, the system renumbers nodes and broadcasts N^C_i ($i \in [1, l]$).Then the node N_i is to be assigned into the set of consensus nodes N^C.

Step 5: All of the trading nodes N^T vote for N^C_i, a random number $(int)(Math.random() * l)$ calculated by N^T_i will be the subscript number of vote for N^C_i.

Step 6: N^T_i signs the $\langle (int)(Math.random() * l), N^T_i \rangle$ through $Encrypt\langle (int)(Math.random() * l), N^T_i \rangle, SK_i \rangle$ with its private key SK_i and gets the $\langle (int)(Math.random() * l), N^T_i \rangle_{SK_i}$, then N^T_i broadcast $\langle \langle (int)(Math.random() * l), N^T_i \rangle_{SK_i}, PK_i \rangle$ which is encrypted with its public key P_k.

Step 7: After receiving the $\langle \langle (int)(Math.random() * l), N^T_i \rangle_{SK_i}, PK_i \rangle$ broadcasted from N^T_i, the system decrypts it through $Decrypt\left(\langle (int)(Math.random() * l), N^T_i \rangle_{SK_i}, PK_i \right)$ with N^T_i's public key PK_i in order to get the plaintext message $\langle (int)(Math.random() * l), N^T_i \rangle$.

Step 8: The system counts the total quantity of the same random number $(int)(Math.random() * l)$ it has received, the computing equation is expressed as follows:

$$NUMBER_{(int)(Math.random()*l)} = Count((int)(Math.random() * l)) \quad (5)$$

Step 9: Ranking the $N^C_{(int)(Math.random()*l)}$ according to the rule that the bigger the $NUMBER_{(int)(Math.random()*l)}$, the higher the ranking of $N^C_{(int)(Math.random()*l)}$. And then the nodes N^C_i would be renumbered so that the top m nodes in N^C with the highest number of votes become witness nodes N^W_i ($i \in [1, m]$) in N^W, and the nodes ranked from $m + 1$ to l become candidate nodes N^A_i($i \in [1, (l-m)]$) in N^A.

2) ACHIEVEMENT OF THE CONSENSUS

From the blockchain consensus environment of DDPoS algorithm shown in Fig. 3, we can see that the trading nodes are responsible for the creation, broadcasting and storage of transaction's information, the witness nodes take turns to record transactions into a block and broadcast the block to the other consensus nodes for validating.

We define a state tag that may be *GOOD*, *NORMAL* or *ERROR* for the block in its header. $BLOCK_{NORMAL}$ is a sign that the block is waited to be verified, $BLOCK_{GOOD}$ is a sign that the block has been verified successfully, and

$BLOCK_{ERROR}$ signs that the block has not been verified successfully. N^W_{NORMAL} denotes the witness nodes' default state, while N^W_{GOOD} denotes the witness node that has generated blocks successfully, *and* N^W_{ERROR} denotes the witness node that has generated a fault block.

The module of reaching a consensus can be described as following:

Step 1: The consensus node N^C_i generates a block that consists of a state tag *NORMAL*, the difficulty D sent by the system, the random value *nonce*, the $HASH(PreBlock)$ of the previous block, some recent transactions *txs* as well as its *MerkleTreeRoot*, and the time of the block *TimeStamp*. The expression of the block is given by (6):

$$BLOCK_i = \langle \langle NORMAL, HASH(PreBlock), MerkleTreeRoot, TimeStamp, D, nonce \rangle, txs \rangle \quad (6)$$

Step 2: Witness nodes that take turns to account will broadcast their own generated blocks to the blockchain network in turn.

Step 3: The trading node N^T_i will forward the $BLOCK_{N^W_i}$, while the consensus node N^C_i compares that with the block generated by themselves after receiving it.

Step 4: If $BLOCK_{N^W_i} = BLOCK_{N^C_i}$, the consensus node N^C_i will produce a piece of encrypted text $\langle BLOCK_{N^W_i} \rangle_{SK_{N^C_i}}$ by signing the $BLOCK_{N^W_i}$ through $Encrypt(BLOCK_{N^W_i}, SK_{N^C_i})$ with its private key $SK_{N^C_i}$ and then broadcasts a $\langle \langle \langle BLOCK_{N^W_i} \rangle_{SK_{N^C_i}} \rangle, PK_{N^C_i} \rangle$. Otherwise, N^C_i will produce $\langle BLOCK_{N^C_i} \rangle_{SK_{N^C_i}}$ by signing its own $BLOCK_{N^C_i}$ through $Encrypt(BLOCK_{N^C_i}, SK_{N^C_i})$ with $SK_{N^C_i}$, and then broadcasts a $\langle \langle \langle BLOCK_{N^C_i} \rangle_{SK_{N^C_i}} \rangle, PK_{N^C_i}$.

Step 5: The witness node N^W_i will decrypt the $\langle \langle \langle BLOCK_{N^W_i} \rangle_{SK_{N^C_i}} \rangle, PK_{N^C_i}$ through $Decrypt\left(\langle \langle BLOCK_{N^C_i} \rangle_{SK_{N^C_i}} \rangle, PK_{N^C_i} \right)$ to get $BLOCK_{N^W_i}$ or decrypt $\langle \langle \langle BLOCK_{N^C_i} \rangle_{SK_{N^C_i}} \rangle, PK_{N^C_i}$ through $Decrypt\left(\langle \langle BLOCK_{N^C_i} \rangle_{SK_{N^C_i}} \rangle, PK_{N^C_i} \right)$ to get $BLOCK_{N^C_i}$.

Step 6: The witness node N^W_i has a responsibility to count the total number of obtained $BLOCK_{N^W_i}$ and $BLOCK_{N^C_i}$ statistically, shown in (7) and (8). The calculation is based on the premise that all consensus nodes can reply $Encrypt\left(BLOCK_{N^W_i}, SK_{N^C_i} \right)$ or $encrypt(BLOCK_{N^C_i}, SK_{N^C_i})$.

$$NUMBER_{BLOCK_{N^W_i}} = \sum BLOCK_{N^W_i} \quad (7)$$

$$NUMBER_{BLOCK_{N^C_i}} = \sum BLOCK_{N^C_i} \quad (8)$$

Step7: If $NUMBER_{BLOCK_{N^W_i}} > NUMBER_{BLOCK_{N^C_i}}$, the state of the $BLOCK_{N^W_i}$ will be *GOOD* and N^W_i will broadcast $BLOCK_{GOOD}$ expressed as $\langle \langle GOOD, HASH(PreBlock), MerkleTreeRoot, TimeStamp, D, nonce \rangle txs \rangle$ to the entire blockchain network so that all of the nodes N_i can add $BLOCK_{GOOD}$ to their own blockchain. Otherwise, the state will be *ERROR*, N^W_i broadcast $BLOCK_{ERROR}$ expressed as
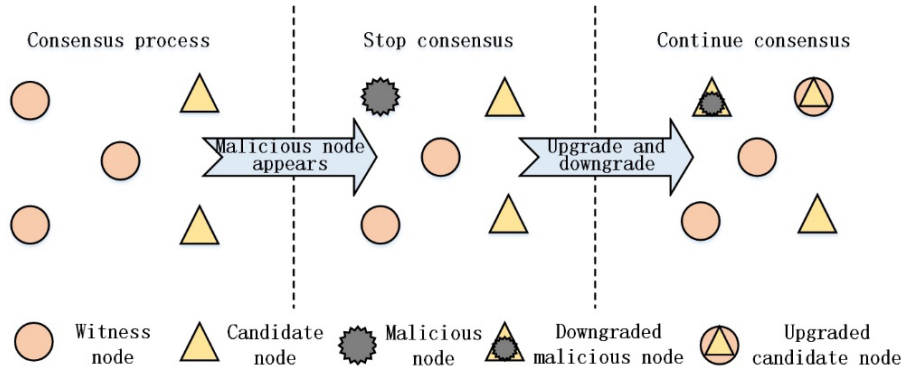
FIGURE 4. Module of eliminating malicious nodes based on DDPoS algorithm.

$\langle\langle ERROR, HASH\ (PreBlock)\ , MerkleTreeRoot, TimeStamp,$ $D, nonce, \rangle txs \rangle$ so that the algorithm can stop generating blocks and enter the module of downgrading the malicious node.

3) DOWNGRADE OF THE MALICIOUS NODES

When the state tag of a block is ERROR, the generator of the block will be judged as malicious node, the witness node N^W_i will stop the consensus module, and DDPoS will enter the module of downgrading malicious nodes. Then DDPoS algorithm will complete the transition of the state tag of consensus nodes N^C_i in nodes set N^W and N^A, and deprive the right of generating block of the faulty witness node N^W_{ERROR} through downgrading it into the candidate nodes set. After that, the top-1 ranked candidate node N^A_1 is upgraded from N^A into N^W. The downgrade mechanism is implemented between the witness nodes set N^W and the candidate nodes set N^A to guarantee the integrity of the witness nodes set N^W, which can avoid the situation where there is no witness node in N^W due to too many wrong nodes.

The state of the downgrading malicious nodes module is shown in Fig. 4, and the steps of downgrading malicious nodes are expressed as follows:

Step 1: When the system finds that the state tag in the block is ERROR, the witness node N^W_i stops the consensus module and the DDPoS algorithm enters the module of downgrading malicious node, performing a downgrade and upgrade operation between N^W and N^A.

Step 2: The blockchain system marks the faulty witness node N^W_γ as N^W_{ERROR}.

Step 3: The node N^W_i ($i \in [\gamma + 1, m]$) ranked after N^W_γ raises its ranking by one in the witness nodes set N^W, that is, reduces its subscript by one through operating N^W_{i-1}, and the witness nodes set becomes $N^W = \{N^W_i | i \in [1, m-1]\}$.

Step 4: The first candidate node N^A_1 in the candidate nodes set N^A is added to the witness nodes set and becomes N^W_m. Thus, the witness nodes set becomes $N^W = \{N^W_i | i \in [1, m]\}$ and the candidate nodes set is $N^A = \{N^A_i | i \in [2, l-m]\}$.

Step 5: All candidate nodes in set $N^A = \{N^A_i | i \in [2, l-m]\}$ raise their ranking by one, that is, reduces their subscript by

one through operating N^A_{i-1}, and the candidate nodes set becomes $N^A = \{N^A_i | i \in [1, l-m-1]\}$.

Step 6: The blockchain system transfers the faulty witness node N^W_{ERROR} to the candidate nodes set N^A, and sets it as N^A_{l-m}. Then the candidate nodes set returns to normal size as $N^A = \{N^A_i | i \in [1, l-m]\}$.

Step 7: After the downgrade operation between the witness nodes set N^W and the candidate nodes set N^A is completed, and the N^W and N^A are regenerated, the blockchain system sets the state to *NORMAL*, the witness node N^W_i starts to generate blocks, and the consensus node N^C_i performs verification on them.

4) THE OVERALL FLOW OF THE DDPoS ALGORITHM

The overall flow of the DDPoS algorithm can be depicted in Fig. 5.

C. IMPLEMENTATION OF DDPoS ALGORITHM

Based on the detailed description of the DDPoS algorithm in the previous section, the pseudocode of the three modules in the DDPoS algorithm is shown in following Algorithm 1, Algorithm 2, and Algorithm 3.

Algorithm 1 is to select the consensus nodes, here ξ denotes the random number generated by the candidate node, and V denotes the number of votes the witness node in N^C owns. Then the module of reaching a consensus is expressed as Algorithm 2, the *NUMBER* represents the total number of blocks. Finally, we use Algorithm 3 to show the module of downgrading malicious nodes.

IV. SIMULATION AND RESULT ANALYSIS
A. SIMULATION DESIGN

We use the Java language to simulate the DDPoS algorithm in the experimental environment built in Eclipse. Blockchain nodes are simulated by some threads with different priorities through multithreading technique. In our simulation experiment, the size of N is changed by setting the parameters of multithreading. For example, by setting the parameters of multithreading to 500, 1000, 2000, 3000, 4000, and 5000, there is a corresponding number of nodes in the simulated blockchain environment. By controlling the running time

Algorithm 1 The Algorithm of Selecting Consensus Nodes

Input: D, $nonce$
Output: N^W, N^A
1: $Broadcast\langle nonce, D\rangle$
2: N_i : $HASH\,(HASH\,(PreBlockHead)\,, nonce)$
3: *while* $(HASH\,(HASH\,(PreBlockHead)\,, nonce) > D)$
4:     $nonce \leftarrow nonce + 1$
5:     $PN^C_i \leftarrow N_i$
6:     $Broadcast\langle N^C\rangle$
7:     $N^T \leftarrow N - N^C$
8:     $\xi_{N^T_i} \leftarrow (int)\,(Math.random() * l)$
9:     N^T_i *vote for* $N^C_{\xi_{N^T_i}}$
10:    $V_{N^C_{\xi_{N^T_i}}} = Count(\xi_{N^T_i})$
11:    $quicksort(V_{N^C_{\xi_{N^T_i}}})$
12:    $N^W_i \leftarrow N^C_i (i \in [1, m])$
13:    $N^A_i \leftarrow N^C_i (i \in (m, l - m])$
14:*end*

Algorithm 2 The Algorithm of Reaching a Consensus

Input: N^W, N^A
Output: $BLOCK_{GOOD} or BLOCK_{ERROR}$
1: N^C_i : $BLOCK_{N^C_i} \leftarrow$
    $\langle\langle NORMAL, HASH(PreBlock), MerkleTreeRoot,$
    $TimeStamp, D, nonce\rangle, txs\rangle$
2: $N^W_i\,Broadcast\langle BLOCK_{N^W_i}\rangle$
3: *if* $HASH\left(BLOCK_{N^C_i}\right) = HASH\left(BLOCK_{N^W_i}\right)$
    $N^C_i\,Broadcast\langle BLOCK_{N^C_i}\rangle$
    else $N^C_i\,Broadcast\langle BLOCK_{N^C_i}\rangle$
4:     N^W_i : $TOTAL(NUMBER_{BLOCK_{N^W_i}})$
    $= \sum BLOCK_{N^W_i} and$
    $NUMBER_{BLOCK_{N^C_i}} = \sum BLOCK_{N^C_i}$
5: *if* $NUMBER_{BLOCK_{N^W_j}} > NUMBER_{BLOCK_{N^C_i}}$
    $Broadcast\langle BLOCK_{GOOD}\rangle$
    else
    $Broadcast\langle BLOCK_{ERROR}\rangle$
6: *end*

Algorithm 3 The Algorithm of Eliminating Malicious Nodes

Input: $BLOCK_{ERROR}$
Output: N^W, N^A
1: $N^W_\gamma\,Broadcast\langle BLOCK_{ERROR}\rangle$
2: $N^W_{ERROR} \leftarrow N^W_\gamma$
3: *for* $i \leftarrow (\gamma + 1)$ *to* m
    N^W_{i-1}
4:    $N^W_m \leftarrow N^A_1$
5: *for* $i \leftarrow 2$ *to* $(l-m)$
    N^A_{i-1}
6:    $N^A_{l-m} \leftarrow N^W_{ERROR}$
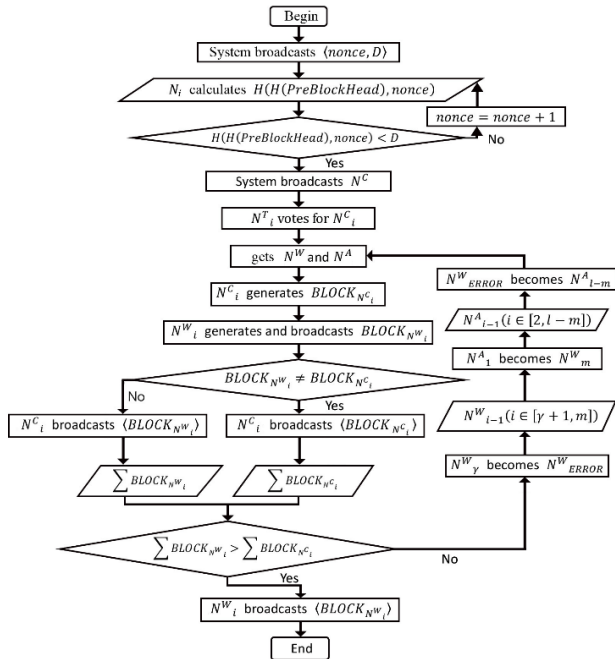7: *get* N^W *and* N^A
8: *end*

FIGURE 5. Flowchart of the DDPoS algorithm.

of the DDPoS algorithm (10, 20, 30, 40, 50, and 60min), we observe the running results of the DDPoS algorithm, and perform analysis on them. In order to enhance the persuasiveness of experimental data, the traditional PoW [1], PoS [13], and DPoS [14] consensus algorithms are also simulated in the experimental environment and their experimental results are compared with DDPoS algorithm objectively.

B. RESULT ANALYSIS

In this section, we compare the experimental results of the PoW, PoS, DPoS, and the proposed DDPoS consensus algorithms. We mainly analyze the performance of DDPoS consensus algorithm from three aspects: efficiency of generating blocks (i.e., accounting efficiency), probability of generating blocks and performance of fault-tolerant. Efficiency of generating blocks is expressed by *SPB* (Seconds Per Block, the time taken by each block), as shown as (9).

$$SPB = \frac{TotalTime}{TotalBlocks} \tag{9}$$

Here *TotalTime* is the running time of the consensus algorithm in each single simulation experiment, and we have set it to 10, 20, 30, 40, 50, and 60min respectively. *TotalBlocks* is the number of blocks generated during the running time of the consensus algorithm. When calculating the *SPB*, we convert the running time unit from minutes to seconds and represent the unit of the accounting efficiency by seconds per block.

We use the *POB* to denote the probability of generating blocks, and expressed it as (10).

$$POB_{\Delta t} = \frac{\left(NUMBER_{N^W_{GOOD}}\right)_{\Delta t}}{(NUMBER_N)_{\Delta t}} \tag{10}$$

Here $POB_{\Delta t}$ indicates the probability of generating blocks during the unit of running time, where Δt represents a unit

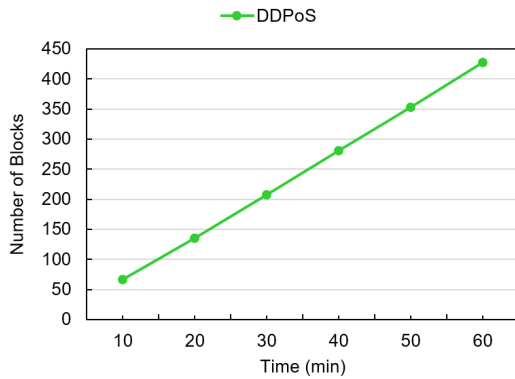**FIGURE 6.** Number of blocks generated by the DDPoS algorithm in size of 1000 nodes.

FIGURE 7. Number of blocks generated by the DDPoS algorithm in 10min.

of running time and may take values of 10, 20, 30, 40, 50, and 60min. And $\left(NUMBER_{N^W_{GOOD}}\right)_{\Delta t}$ indicates the number of consensus nodes which have generated blocks, while $\left(NUMBER_N\right)_{\Delta t}$ is the number of all nodes in N during the Δt.

Performance of fault-tolerant is the ability to find errors and recover from them. In this paper, it refers to the ability to detect malicious nodes in the blockchain system and quickly downgrade them to ensure that the system can operate safely and smoothly.

1) EFFICIENCY OF GENERATING BLOCKS

Firstly, we calculate the number of blocks generated by the DDPoS consensus algorithm in different time periods with the size of 1000 nodes, and the accounting efficiency of this consensus algorithm. A line graph of the number of blocks generated by the 1000-node size DDPoS consensus algorithm over different time periods is shown in Fig. 6.

It can be seen from Fig. 6 that the number of blocks generated by the DDPoS algorithm increases as time increases, and the slope of the block number curve also increases slightly. The possible reason is that in the DDPoS consensus algorithm, the time spent in selecting the consensus nodes is almost unchanged, so as the running time of DDPoS increases, the proportion of time spent by the module of selecting consensus nodes in the total consensus time is decreasing, resulting in a slight increase in the number of generated blocks. Putting the data of Fig. 6 into the above mentioned (9), it can be concluded that the efficiency of generating blocks *SPB* of the DDPoS consensus algorithm at 1000-node size is approximately 9s/block.

Next, we count the number of blocks generated by the DDPoS consensus algorithm with different nodes set size (500, 1000, 2000, 3000, 4000, and 5000) in 10min, and draw a line chart showing the number of blocks changing with the number of all blockchain nodes, as shown in Fig. 7.

It can be seen from Fig. 7 that, the number of blocks generated by the DDPoS algorithm is decreasing as the size of nodes set becomes larger, but the absolute value of slope of the *SPB* curve in Fig. 7 increases slightly. It is speculated
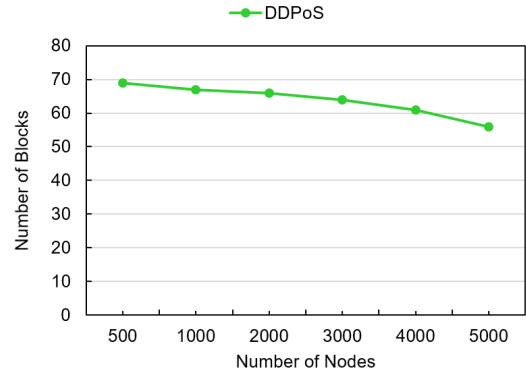
that as the size of N increases sharply, the number of votes becomes larger, and the time spent in this phase becomes longer, which inevitably affects the efficiency of generating blocks. As a result, the accounting efficiency *SPB* of DDPoS consensus algorithm is decreasing due to the number of nodes becoming larger. It can also be found that when the nodes number is less than 3000, the *SPB* of the DDPoS consensus algorithm is about 9s/block, but when the nodes number exceeds 4000 the *SPB* will decrease, and when the number reaches 5000, the *SPB* will be reduced to 10s/block.

Finally, Fig. 8 describes the changes in the number of blocks generated by the three consensus algorithms PoW, PoS, and DDPoS with different nodes number (500, 1000, 2000, 3000,4000, and 5000) in different lengths of time periods (10, 20, 30, 40, 50, and 60min). It can be seen that the accounting efficiency of the PoW consensus algorithm is hardly affected by time and nodes number, and the number of generated blocks increases with time. This is because PoW takes a long time to generate a block (about 10min), and there is an adjustable difficulty value in PoW. Whenever 2016 blocks are generated, all nodes will automatically adjust the difficulty: if a block is generated less than 10min, the difficulty value will increase, and if more than 10min, the difficulty value will decrease.

For the DDPoS consensus algorithm in any length of simulation time, the number of blocks generated by the DDPoS consensus algorithm is decreasing as the nodes number increases, but the absolute value of slope of the *SPB* curve in Fig. 8 decreases slightly. This is further consistent with our analysis of Fig. 7, which shows that this problem is not a special case, but one of the problems faced by the DDPoS consensus algorithm.

2) PROBABILITY OF GENERATING BLOCKS

The witness nodes of the DPoS and DDPoS algorithms are set to be updated every 10min to keep consistent with the experimental parameters of DDPoS in our experiment. And we define several data symbols to provide a detailed and clear description of the experimental results: λ denotes the number of witness nodes that only generate one block over a period
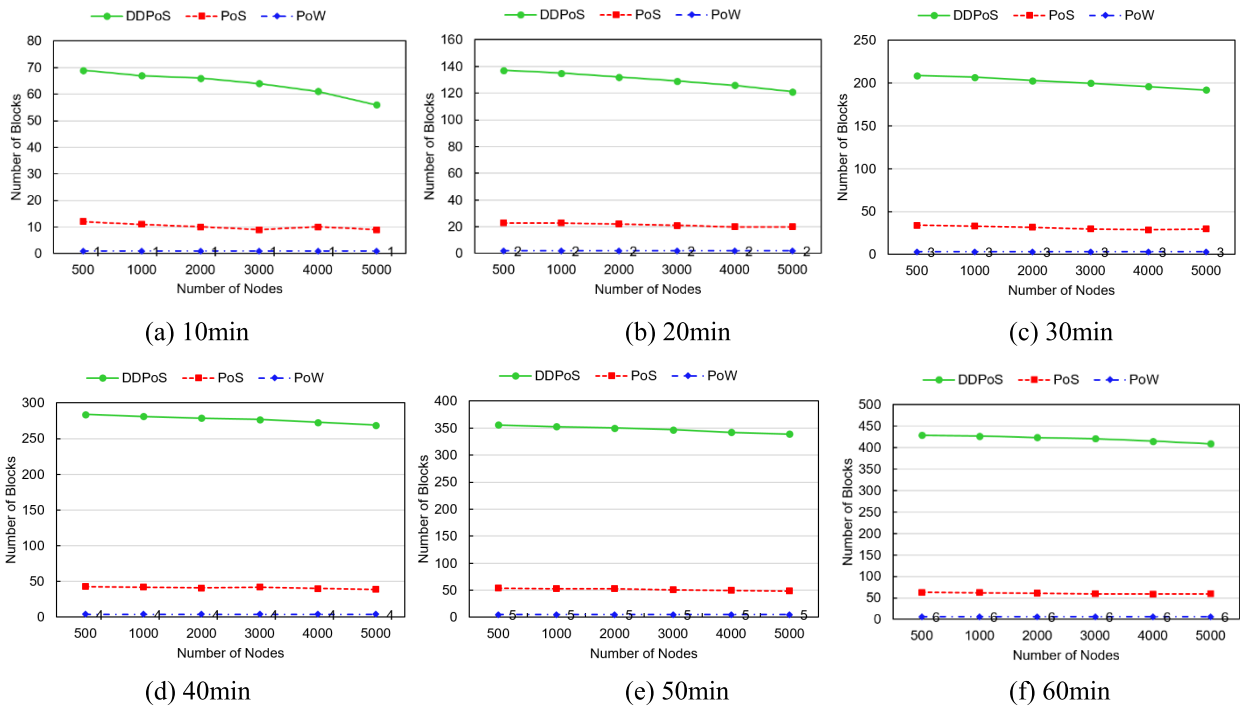
FIGURE 8. Comparison of the number of blocks generated by the three consensus algorithms. (a) 10min. (b) 20min. (c) 30min. (d) 40min. (e) 50min. (f) 60min.
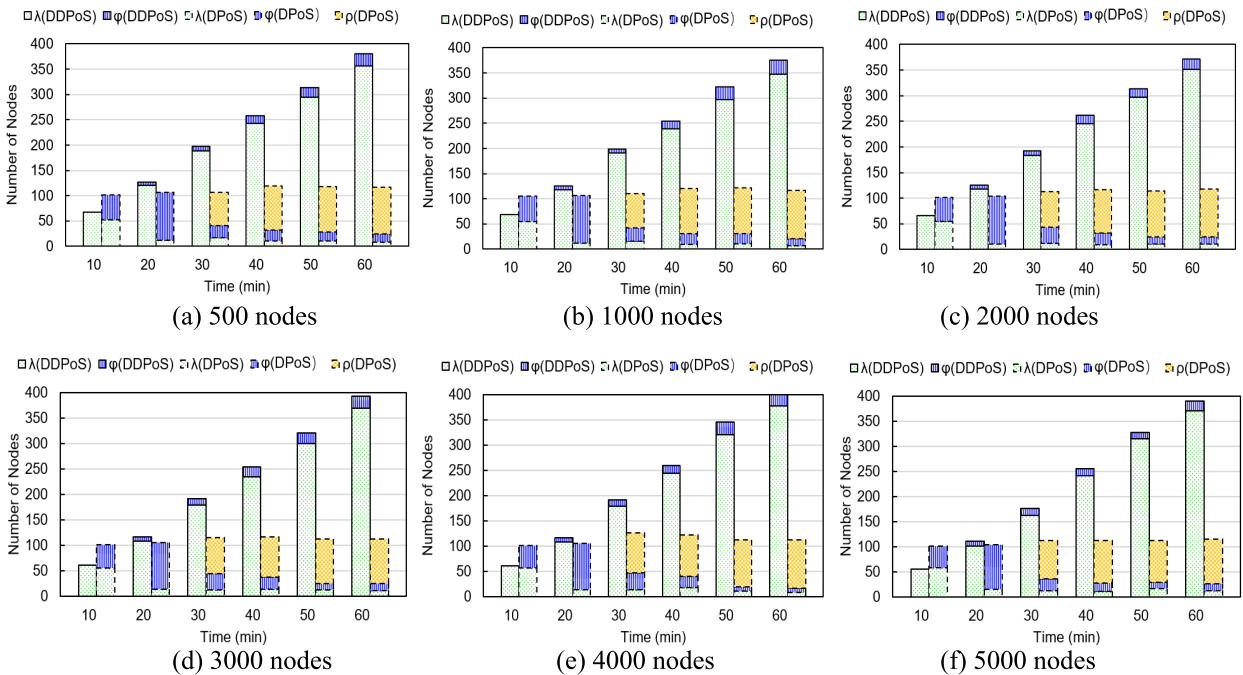

FIGURE 9. Number of consensus nodes generated by DPoS and DDPoS algorithms. (a) 500 nodes. (b) 1000 nodes. (c) 2000 nodes. (d) 3000 nodes. (e) 4000 nodes. (f) 5000 nodes.

of time, that is, $\lambda(DDPoS)$ denotes the number of witness nodes that generated one block over a period of time by the DDPoS, $\lambda(DPoS)$ denotes the number of witness nodes that generated one block by DPoS. The symbol φ represents the number of witness nodes that generate two to four blocks during a period of time. The symbol ρ denotes the number of witness nodes generating more than four blocks. The reason why we divide the nodes in this way is that the experimental
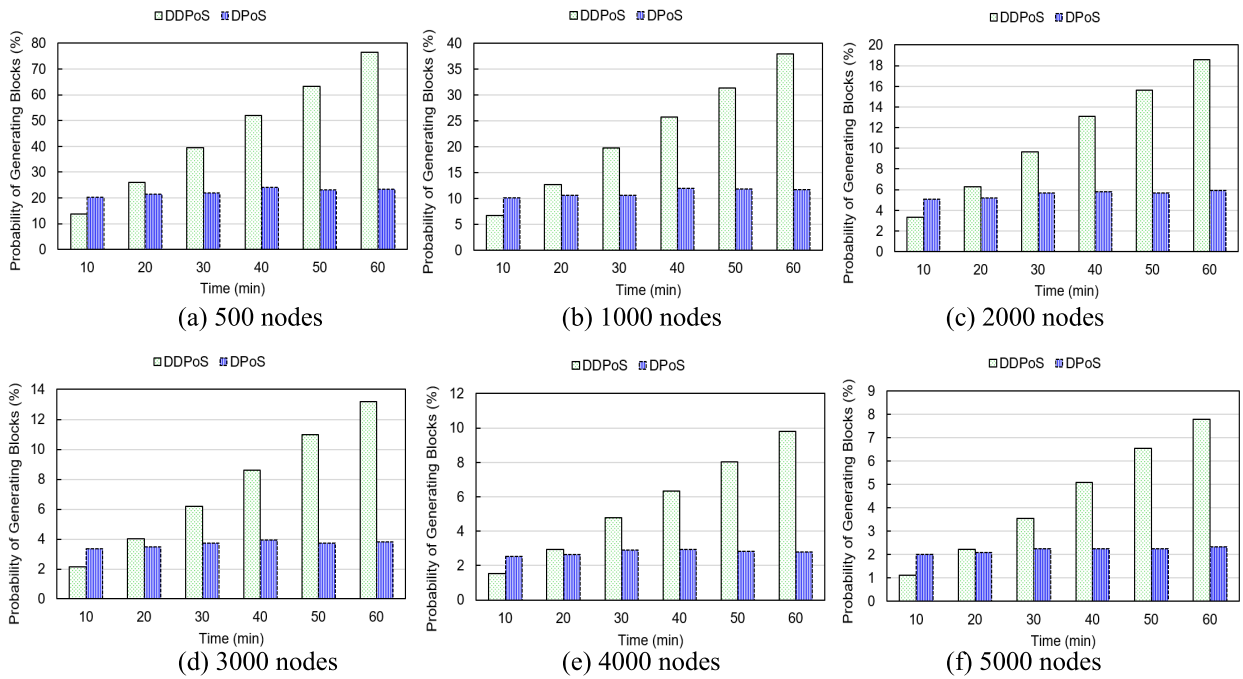
FIGURE 10. Probability of generating blocks through DPoS and DDPoS algorithms. (a) 500 nodes. (b) 1000 nodes. (c) 2000 nodes. (d) 3000 nodes. (e) 4000 nodes. (f) 5000 nodes.

data shows that this division method can better evaluate the *POB* by comparing the number of these three types of nodes.

Firstly, the number of witness nodes that generate different numbers of blocks in DPoS and DDPoS over different period of time (10, 20, 30, 40, 50, and 60min) with different size of nodes (500, 1000, 2000, 3000, 4000, and 5000) will be displayed by histogram. Then the *POB* of the two consensus algorithms will be calculated according to (10), and the differences between them will be compared. The specific experimental results are reflected in Fig. 9 and Fig. 10.

As previous depiction, when the system runs under DPoS and DDPoS consensus algorithms for 10 minutes, the blocks and witness nodes generated under the DDPoS are fewer than DPoS. This is because: the *SPB* of the experimental DPoS and DDPoS are about 4s/block and 9s/block respectively, so that it can be deduced that the DPoS consensus algorithm can generate about 150 blocks while the DDPoS 66 blocks in 10min. In our experiments we set the number of witness nodes in each round of the DPoS and the DDPoS consensus algorithms to 101. It is known that within 10 minutes, each witness node in DPoS algorithm generates at least one block while each in DDPoS less than one. Therefore, the number of witness nodes elected by the DPoS is more than that of the DDPoS. This conclusion also explains why the *POB* of the DDPoS consensus algorithm calculated by (10) is lower than the DPoS within 10min.

From (a) in Fig. 9, it is found that in DPoS algorithm, the $\left(NUMBER_{N^W{}_{GOOD}}\right)_{10min}$ is equal to 101 and the $\left(NUMBER_{N^W{}_{GOOD}}\right)_{60min}$ is less than 130. But in the DDPoS algorithm, the $\left(NUMBER_{N^W{}_{GOOD}}\right)_{\Delta t}$ has risen from 66 in

10min to more than 350 in 60min. Moreover, because there was no node having generated more than four blocks, so the $\rho\,(DPoS)$ is always equal to zero and has not been displayed in Fig. 9. While the $\lambda(DDPoS)$ in the DDPoS consensus algorithm increases rapidly with time, and $\varphi\,(DDPoS)$ basically remains stable. When the time exceeds 20min, $\lambda\,(DPoS)$ remains relatively stable and accounts for a small proportion of the total number of nodes. When the time exceeds 30min, $\varphi\,(DPoS)$ remains basically stable and accounts for a small proportion, while $\rho\,(DPoS)$ accounts for a large of more than 50%. This is because the DPoS consensus algorithm uses stake as the basis for selecting witness nodes. But there are only a small number of nodes have a large number of stakes in a system, so the witness nodes will always be generated in those nodes, and as time goes on, the proportion of $\rho\,(DPoS)$ is getting bigger and bigger. In contrast, the DDPoS is an improvement based on the DPoS, which reduces the influence of the stake factor and keeps the proportion of $\lambda\,(DDPoS)$ increasing with time. This is supposed to benefit from one of the innovations in this paper: we try to decentralize the right of generating blocks in the blockchain system to make more nodes get the opportunity to generate blocks.

From (a) to (f), we can see that, even if the nodes size changes from 500 to 5000, the total number of witness nodes that have generated blocks and the number (such as $\lambda\,(DDPoS)$, $\varphi\,(DDPoS)$, $\lambda\,(DPoS)$, and $\rho\,(DPoS)$) of witness nodes of different types in DPOS and DDPOS does not change much.

It can be obtained from Fig. 10 that while the *POB* of DDPoS increases with time, the DPoS remains nearly

TABLE 4. Fault-tolerant analysis of DDPoS.

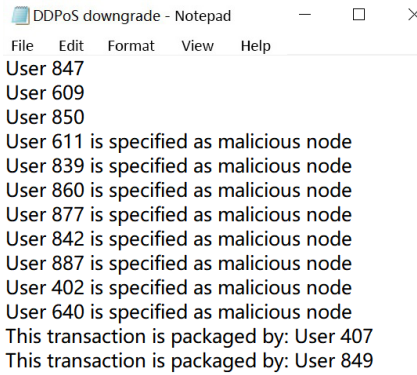| Time(min) Experimental result	10	20	30	40	50	60
Number of specified malicious nodes	8	8	8	10	10	12
Number of discovered malicious nodes	7	8	8	10	10	12

FIGURE 11. Specified malicious node.

unchanged. Taken together, as the size of the *N* increases in the same period of running time, the *POB* of the DPoS and DDPoS algorithms are decreasing. As mentioned in the previous content, while the size of *N* as $NUMBER_{N_{\Delta t}}$ becoms larger, the accounting efficiency *SPB* of DDPoS is decreasing, and the number equal to $\left(NUMBER_{N^{W}_{GOOD}}\right)_{\Delta t}$ of witness nodes which have generated blocks is slightly reducing, so the *POB* is inevitably decreasing. However, the DDPoS consensus algorithm always has a higher *POB* than DPoS in the same length of running time when more than 10 minutes.

3) PERFORMANCE OF FAULT-TOLERANT

The fault-tolerant performance of DDPoS consensus algorithm is analyzed to test whether the downgrade mechanism introduced in this paper can realize the downgrade of malicious nodes. In our simulation experiment, several malicious nodes are specified in the witness nodes set N^{W}, so that they can generate a wrong block and broadcast them to the other witness nodes for consensus. Then we observe the experimental results to check whether the system can find out the malicious nodes and perform the downgrade mechanism. We set the size of *N* to 1000, the malicious nodes number and running time are set as Table 4. We firstly simulate the DDPoS mechanism in 10min with specified 8 malicious nodes as shown in Fig. 11. Finally, we get the experimental result as shown in Fig. 12.

Next, we set the consensus time to 20, 30, 40, 50 and, 60 min, then count the number as shown in Table 4 of specified and discovered malicious nodes respectively.

It is found from Fig. 12, when running time is 10min, that in the simulation experiment only 7 malicious nodes have been discovered from the total of 8, shown in Table 4.
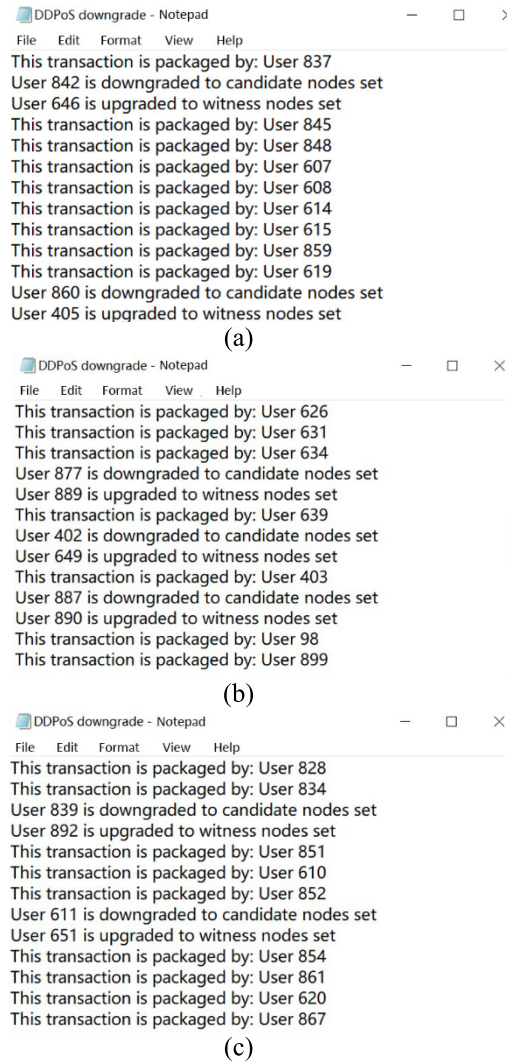

FIGURE 12. Malicious node downgrade result in 10min.

But if the running time exceeds 20 minutes, the DDPoS algorithm can find out and downgrade all of the malicious nodes, no matter how many malicious nodes are specified. As we know, the above mentioned DDPoS algorithm has an accounting efficiency of about 9s/block, so there are nearly 66 generated blocks in 10 minutes. Though we set the number of witness nodes to 101, not every witness node can generate a block in 10 minutes, so it is possible that one of the 8 malicious nodes specified with a sequence number greater than 66 in the witness nodes set has not been found. When the consensus time exceeds 20min, the sequence number of generated blocks is greater than 101, and each witness nodes has generated blocks, so all specified malicious nodes will be discovered. This also proves that the downgrade mechanism introduced in this paper can prevent the malicious behavior [45] of witness node in DPoS algorithm.

In a word, it is fully demonstrated that the improved DDPoS consensus algorithm has good performance in the above three aspects.

V. CONCLUSION AND FUTURE WORK

This paper presents an effective consensus algorithm named DDPoS for lower resource consumption, higher operating efficiency and stronger security of the blockchain. The downgrade mechanism we introduce is a good solution to the problem of witness's malicious behavior.

The innovations of this paper are as follows:

The first is the idea of combining the advantages of PoW and DPoS to improve the original DPoS algorithm. We use the PoW to select a set of nodes with sufficient computational power instead of stakes to participate in the election. Secondly, we stipulate that each node has only one vote for randomly voting, which improves the fairness and decentralizes the right of generating blocks to avoid the collusion attack and improving the node activity of the entire blockchain system. Finally, this paper adopts the downgrade mechanism to quickly downgrade the malicious nodes to maintain the good operation and security of the system.

Compared with the traditional consensus algorithm, our research has improved performance both in efficiency and security. But due to the difficulty in directly applying our consensus algorithm in the existing business environment such as Hyperledger and Ethereum, our experimental environment is not in the real situation. In ideal condition, if we can be authorized by big blockchain companies, our consensus algorithm would be tested in a real blockchain environment.

ACKNOWLEDGMENT

REFERENCES

[1] X. Shen, Q. Q. Pei, and X. F. Liu, "Survey of block chain," *Chin. J. Netw. Inf. Secur.*, vol. 2, no. 11, pp. 11–20, 2016.

[2] K. Salah, M. H. Ur Rehman, N. Nizamuddin, and A. Al-Fuqaha, "Blockchain for AI: Review and open research challenges," *IEEE Access*, vol. 7, pp. 10127–10149, 2019.

[3] H. R. Hasan and K. Salah, "Combating deepfake videos using blockchain and smart contracts," *IEEE Access*, vol. 7, pp. 41596–41606, 2019.

[4] M. A. Khan and K. Salah, "IoT security: Review, blockchain solutions, and open challenges," *Future Gener. Comput. Syst.*, vol. 82, pp. 395–411, May 2018. doi: 10.1016/j.future.2017.11.022.

[5] R. Almadhoun, M. Kadadha, M. Alhemeiri, M. Alshehhi, and K. Salah, "A user authentication scheme of IoT devices using blockchain-enabled fog nodes," in *Proc. IEEE/ACS 15th Int. Conf. Comput. Syst. Appl. (AICCSA)*, Oct. 2018, pp. 1–8.

[6] A. Suliman, Z. Husain, M. Abououf, M. Alblooshi, and K. Salah, "Monetization of IoT data using smart contracts," *IET Netw.*, vol. 8, no. 1, pp. 32–37, Jan. 2019.

[7] M. Turkanović, M. Hölbl, K. Košič, M. Hericko, and A. Kamišalić, "EduCTX: A blockchain-based higher education credit platform," *IEEE Access*, vol. 6, pp. 5112–5127, 2018.

[8] M. A. Rashid, K. Deo, D. Prasad, K. Singh, S. Chand, and M. Assaf. *TEduChain: A Platform for Crowdsourcing Tertiary Education Fund Using Blockchain Technology.* Accessed: Jan. 2019. [Online]. Available: https://arxiv.org/abs/1901.06327

[9] Z. Huang, X. X. Li, X. J. Lai, and K. F. Chen, "Blockchain technology and its applications," *J. Inf. Secur. Res.*, vol. 3, no. 3, pp. 237–245, 2017.

[10] G. Zyskind, O. Nathan, and A. S. Pentland, "Decentralizing privacy: Using blockchain to protect personal data," in *Proc. IEEE Secur. Privacy Workshops.* May 2015, pp. 180–184.

[11] J. Kang, Y. Rong, X. Huang, S. Maharjan, Y. Zhang, and E. Hossain, "Enabling localized peer-to-peer electricity trading among plug-in hybrid electric vehicles using consortium blockchains," *IEEE Trans. Ind. Informat.*, vol. 13, no. 6, pp. 3154–3164, Dec. 2017.

[12] Z. Meng, T. Morizumi, S. Miyata, and H. Kinoshita, "Design scheme of copyright management system based on digital watermarking and blockchain," in *Proc. IEEE 42nd Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, Jul. 2018, pp. 359–364.

[13] P. Vasin. (2014). *BlackCoin's Proof-of-Stake Protocol v2.* [Online]. Available: https://blackcoin. co/blackcoin-pos-protocol-v2-whitepaper. pdf

[14] T. Y. Song and Y. L. Zhao, "Comparison of blockchain consensus algorithm," *Comput. Appl. Softw.*, vol. 25, no. 8, pp. 1–8, 2018.

[15] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *Proc. Symp. Operating Syst. Design Implement.*, New Orleans, LA, USA, 1999, pp. 173–186. doi: 10.1145/571637.571640.

[16] P4Titan. (May 17, 2014). *Slimcoin: A Peer-to-Peer Crypto-Currency With Proof-of-Burn (Mining Without Powerful Hardware).* [Online]. Available: http://www.doc.ic.ac.uk/~ids/realdotdot/crypto_papers_etc_worth_reading/proof_of_burn/slimcoin_whitepaper.pdf

[17] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld, "Proof of activity: Extending bitcoin's proof of work via proof of stake," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 42, no. 3, pp. 34–37, 2014.

[18] M. Milutinovic, W. He, H. Wu, and M. Kanwal, "Proof of luck: An efficient blockchain consensus protocol," in *Proc. 1st Workshop Syst. Softw. Trusted Execution*, Dec. 2016, p. 2.

[19] D. Mazires. (Jul. 14, 2015). *The Stellar Consensus Protocol: A Federated Model for Internet-Level Consensus.* [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.696.93&amp=&rep=rep1&amp=&type=pdf

[20] Dantheman. (2016). *DPOS Consensus Algorithm-The Missing White Paper.* [Online]. Available: https://steemit.com/DPoS/@dantheman/DPoS-consensus-algorithm-this-missing-white-paper

[21] Y. Yuan and F.-Y. Wang, "Blockchain: The state of the art and future trends," *Acta Autom. Sinica*, vol. 42, no. 4, pp. 481–494, 2016.

[22] M. Walport. Distributed Ledger Technology: Beyond Block Chain. Government Office for Science. London, U.K. GS/16/1. Accessed: Jan. 19, 2016. [Online]. Available: https://www.gov.uk/government/publications/distributed-ledger-technology-blackett-review

[23] P. Zheng, Z. Zheng, X. Luo, X. Chen, and X. Liu, "A detailed and real-time performance monitoring framework for blockchain systems," in *Proc. 40th Int. Conf. Softw. Eng. Pract.*, Gothenburg, Sweden, May 2018, pp. 134–143.

[24] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2016, pp. 839–858.

[25] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the Internet of Things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

[26] I. Alqassem, I. Rahwan, and D. Svetinovic, "The anti-social system properties: Bitcoin network data analysis," *IEEE Trans. Syst., Man, Cybern., Syst.*, to be published.

[27] D. Basin, C. Cremers, T. H.-J. Kim, A. Perrig, R. Sasse, and P. Szalachowski, "Design, analysis, and implementation of ARPKI: An attack-resilient public-key infrastructure," *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 3, pp. 393–408, May/Jun. 2018.

[28] F. Schuh and S. Schießl. *The BitShares Blockchain.* Accessed: Nov. 29 2018. [Online]. Available: https://github.com/bitshares-foundation/bitshares.foundation/blob/master/download/articles/BitShares Blockchain.pdf

[29] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, Jul. 1982.

[30] M. Castro and B. Liskov, "Practical byzantine fault tolerance and proactive recovery," *ACM Trans. Comput. Syst.*, vol. 20, no. 4, pp. 398–461, Nov. 2002.

[31] D. Ongaro and J. K. Ousterhout, "In search of an understandable consensus algorithm," in *Proc. USENIX Annu. Tech. Conf.*, Mar. 2015, pp. 305–319.

[32] T. Nguyen and K. Kim, "A survey about consensus algorithms used in blockchain," *J. Inf. Process. Syst.*, vol. 14, no. 1, pp. 101–128, Feb. 2018.

[33] D. Schwartz, N. Youngs, and A. Britto. (Apr. 10, 2018). *The Ripple Protocol Consensus Algorithm.* [Online]. Available: https://ripple.com/files/ripple_consensus_whitepaper.pdf

[34] J. Kwon. (2014). *Tendermint: Consensus Without Mining.* [Online]. Available: https://cdn.relayto.com/media/files/LPgoWO18TCeMIggJVakt_tendermint.pdf

[35] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proc. 26th Symp. Operating Syst. Princ.*, Shanghai, China, Oct. 2017, pp. 51–68.

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

[37] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song, "The honey badger of BFT protocols," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.* Xi'an, China, Oct. 2016, pp. 31–42.

[38] T. Duong, L. Fan, and H. S. Zhou. (Apr. 15, 2017). *2-hop blockchain: Combining Proof-of-Work and Proof-of-Stake Securely*. [Online]. Available: https://eprint.iacr.org/2016/716.pdf

[39] L. Ren. (Apr. 2014). *Proof of Stake Velocity: Building the Social Currency of the Digital Age*. [Online]. Available: https://www.reddcoin.com/papers/PoSV.pdf

[40] M. Ball, A. Rosen, and M. Sabin, and P. V. Vasudevan. (Feb. 27, 2017). *Proofs of Useful Work*. [Online]. Available: https://eprint.iacr.org/2017/203.pdf

[41] B. David, P. Gaži, A. Kiayias, and A. Russell, "Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.* 2018, pp. 66–98. [Online]. Available: https://eprint.iacr.org/2017/573.pdf

[42] Z. W. Zhang. *A Byzantine Fault Tolerant Algorithm for Blockchain*. Accessed: 2018. [Online]. Available: https://docs.neo.org/zh-cn/basic/consensus/whitepaper.html

[43] X. F. Liu, "Research on performance improvement of blockchain based on dynamic authorization of byzantine fault tolerance consensus algorithm," M.S. thesis, Dept. Comput. Sci. Technol., Zhejiang Univ., Hangzhou, China, 2017.

[44] Z. B. Zheng, S. A. Xie, H. N. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *Proc. IEEE Int. Congr. Big Data*, Jun. 2017, pp. 557–564. doi: 10.1109/BigDataCongress.2017.85.

[45] Stellabelle. (Sep. 29, 2017). *Explain Delegated Proof of Stake Like I'm 5*. https://hackernoon.com/explain-delegated-proof-of-stake-like-im-5888b2a74897d

FAN YANG received the Ph.D. degree in computer science and technology from the School of Computer Science, Wuhan University. She is currently an Associate Professor with the Zhongnan University of Economics and Law. She has long been engaged in teaching and research in the field of information security. Her research interests include the security mechanism under the wireless networks, the application of blockchain technology in the field of intellectual property, and e-commerce. She is also the Director of the Hubei Electronic Commerce Society.

WEI ZHOU received the B.S. degree in e-commerce technology from the School of Information and Safety Engineering, Zhongnan University of Economics and Law, in 2018, where he is currently pursuing the M.S. degree in computer science. He has long been engaged in the field of information security and computer science technology. His research interests include blockchain and intellectual property.
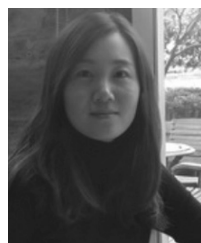
QINGQING WU received the B.S. degree in information management and information system from the Hubei University of Economics, in 2016. She is currently pursuing the M.S. degree in management of information safety with the Zhongnan University of Economics and Law. Her current research include information safety and consensus algorithm of blockchain.

RUI LONG received the B.S. and M.S. degrees in computer science technology from the School of Information and Safety Engineering, Zhongnan University of Economics and Law, in 2016 and 2019, respectively. He has long been engaged in the field of information security and computer science technology. His research interests include blockchain and intellectual property.

NEAL N. XIONG received the Ph.D. degrees from Wuhan University (about sensor system engineering) and the Japan Advanced Institute of Science and Technology (about dependable sensor networks). Before he attended Tianjin University, he was with Northeastern State University, Georgia State University, the Wentworth Institute of Technology, and Colorado Technical University (a Full Professor for five years) for ten years. He has published over 300 international journal papers and over 100 international conference papers. His research interests include cloud computing, security and dependability, parallel and distributed computing, networks, and optimization theory. He has been the General Chair, the Program Chair, the Publicity Chair, a PC Member, and an OC Member of over 100 international conferences and a Reviewer of about 100 international journals, including the IEEE JSAC, the IEEE SMC (Park: A/B/C), the IEEE TRANSACTIONS ON COMMUNICATIONS, the IEEE TRANSACTIONS ON MOBILE COMPUTING, and the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS.

MEIQI ZHOU is currently pursuing the B.S. degree in computer science and technology with the Zhongnan University of Economics and Law, Wuhan, China. Her current research interests include blockchain, cloud computing, and deep learning.

• • •