Voting-Based Decentralized Consensus Design for Improving the Efficiency and Security of Consortium Blockchain

Gang Sun, *Member, IEEE*, Miao Dai, *Graduate Student Member, IEEE*, Jian Sun, *Member, IEEE*, and Hongfang Yu, *Member, IEEE*

***Abstract*—Since its emergence, blockchain technology has received great attention because of its advantages in terms of decentralization, transparency, traceability, and the ability to be tamper proof. These advantages help blockchain become a better option for fields, such as digital currency and information storage. Specifically, consortium blockchain is preferred by researchers because it provides a certain degree of access control and a supervisory mechanism. However, in real-world applications, blockchain platforms pervasively show bottlenecks, such as ultrahigh energy consumption, time inefficiency, low transaction throughput, vulnerability to targeted attacks, and poor fairness of user profits, which seriously influence the performance of this technology and thus hinder its development and adoption. In this article, we try to enhance the performance of blockchain platforms by optimizing the quality of its core module, known as the consensus algorithm. To do so, we introduce proof of assets and proof of reputation to design a voting-based decentralized consensus (VDC) algorithm for consortium blockchain. Combined with the verifiable random function (VRF), VDC realizes better fairness of user profits and time efficiency with acceptable energy consumption and without sacrificing security. The simulation results show that the proposed algorithm achieves a faster consensus process and better user fairness than existing algorithms while still maintaining a negligible energy cost and adequate security.**

***Index Terms*—Consensus algorithm, consortium blockchain, deposit, reputation, verifiable random function (VRF), voting.**

Manuscript received August 9, 2020; revised September 14, 2020 and September 23, 2020; accepted October 6, 2020. Date of publication October 9, 2020; date of current version April 7, 2021. This work was supported in part by the PCL Future Greater-Bay Area Network Facilities for Large-Scale Experiments and Applications under Grant PCL2018KP001. *(Corresponding author: Gang Sun.)*

Gang Sun is with the Key Laboratory of Optical Fiber Sensing and Communications (Ministry of Education) and the Agile and Intelligent Computing Key Laboratory of Sichuan Province, University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: gangsun@uestc.edu.cn).

Miao Dai and Jian Sun are with the Key Laboratory of Optical Fiber Sensing and Communications (Ministry of Education), University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: 1450775540@qq.com; sj@uestc.edu.cn).

Hongfang Yu is with the Key Laboratory of Optical Fiber Sensing and Communications (Ministry of Education), University of Electronic Science and Technology of China, Chengdu 611731, China, and also with Peng Cheng Laboratory, Shenzhen 518066, China (e-mail: yuhf@uestc.edu.cn).

Digital Object Identifier 10.1109/JIOT.2020.3029781

I. Introduction

In a traditional centralized structure, privileges are concentrated in the hands of individuals or the minority, who have strong control over the system and are responsible for maintaining its normal and stable operation. The main architecture of contemporary systems continues to be characterized by this mode because this kind of system can run efficiently with no disagreement, reasonably schedule system resources and make full use of the computational power of the system.

Regarding the development of new application requirements, the centralized mode is increasingly showing its innate drawbacks. There are no central nodes that possess infinite CPU power, and when the scale of the system approximates the CPU limit of the center, it can no longer run smoothly. Such scenarios are currently quite common in the era of big data. In addition, in a centralized system, information is recorded in the central database, and managers have absolute power to tamper with and leak data or to make unreasonable regulations, such as asking for a high transaction tax.

Since the emergence of blockchain technology, there is another option for addressing the problems above. Compared with traditional architecture, blockchain technology involves a new design pattern. The greatest difference between the two types of architecture is that a system using blockchain technology has no central institution: 1) every node in the system shares equal authority and 2) no subordinate relationship exists.

Moreover, blockchain promises better privacy than traditional architecture because nodes are identified by their addresses, not by personal identity. Blockchain also has a strong tamper-proof ability: every transaction contained by the body of a block will generate a hash value; then, a unique hash result will be calculated from these values and be written into the header of the block. In Bitcoin, it is the Merkle tree root [1]–[3], together with the time stamp, the current version information, the hash value of the previous block and so on. Thus, any modification, whether in the body of a block or the header, will lead to a different hash result that can easily be detected. Under such circumstances, if one wants to tamper with a block, he or she must recalculate all the hash values of the blocks from the current block to the end of the blockchain because the latter block contains the hash value of the former block, and such recalculation is believed to be impossible.

These characteristics of blockchain determine that it is adept at solving security [4]–[7] and privacy problems [8], [9]. The application of blockchain technology has extended to many fields, such as the Internet of Things (IoT) [10]–[14], the Internet of Vehicles (IoV) [15], [16], artificial intelligence (AI) [17]–[23], healthcare [24], [25], and supply chain management [26], [27].

The advantages of blockchain are apparent; at the same time, however, they also cause some problems for systems. In such a decentralized system, every participant maintains a replica of the blockchain account book and has equal privileges in accessing transaction records. Thus, it is natural to consider how to ensure the consistency of different account books and to determine who is responsible for generating new blocks or to judge whose block is legal and acceptable. These issues are closely related to the consensus algorithm.

As the most vital segment of blockchain technology, the consensus algorithm is responsible for determining who is to package new blocks and to implement consistency in the account book replicas. The performance of the consensus algorithm will directly affect the performance of the system, including the energy cost, time efficiency, and transaction throughput. For example, in Bitcoin, which employs Proof of Work (PoW) as its consensus algorithm, participants are asked to compete for the privilege of packing a block by solving difficult mathematical puzzles, which is called mining. From a safety and consistency perspective, PoW is believed to be impeccable because any malicious behavior that tries to reverse the existing chain will end up failing as long as the majority of the computational ability spent extending the longest chain is honest. However, PoW comes at the cost of extra CPU consumption, a higher time cost and a lower system throughput, which cannot cope with the Quality-of-Service (QoS) requirements of some scenarios, thus hindering the extensive application of blockchain technology. To mitigate these shortcomings, we propose a new consensus algorithm for consortium blockchain to achieve better performance in time delay, transaction throughput and fairness of user profits with the necessary security and a fairly low energy cost. The main contributions of our work are as follows.

1) We design a new consensus model where different system nodes are assigned different identities and operate different functions, handling distinct affairs during the consensus process. In our voting-based decentralized consensus (VDC) algorithm, each node can possess only one identity each moment but can transform into other kinds under some circumstances, which helps to restrict and decentralize the authority of nodes to avoid a monopoly or an overconcentration of power.
2) We design a simple and efficient method called *DrawLottery* to determine the ownership of the privilege of packing a block instead of mining or appointing a specific leader. When combined with the verifiable random function (VRF), it can achieve unpredictable and verifiable properties, which holds great significance for resilience against targeted attacks. In addition, the simplicity and efficiency of lottery operation allow VDC to frequently reallocate the privilege to other nodes to realize greatly balanced opportunities for participants to pack a block, i.e., better fairness of user profits.
3) Unlike the general consensus algorithms for consortium blockchain, which adopt only reputation or credit as incentives, we also take into account assets to further encourage nodes to behave honestly. Operations complying with the consensus regulations will receive corresponding rewards; in contrast, any behavior that violates the rules will be punished both reputationally and financially. This incentive mechanism helps raise the cost of malicious behaviors and makes VDC more secure.

The remainder of this article is organized as follows. Section II reviews related works. Section III introduces and describes some preliminaries and the system model. Section IV describes our VDC algorithm in detail. Then, we analyze the properties of VDC in Section V. Section VI shows and analyzes the simulation results. Section VII draws the final conclusions of this article.

II. Related Work

In this section, we describe some previous works about blockchain consensus algorithms.

A. Proof of Work

PoW was the earliest consensus algorithm employed by blockchain platforms. In 2008, Satoshi Nakamoto published a paper called "Bitcoin: A Peer to Peer Electronic Cash System" [28] that marked the birth of the first digital currency system, which adopted PoW as its consensus algorithm. The basic principle of this algorithm involves proving validity through the consumption of computational power: in addition to the fields mentioned in Section I, there is also a nonce value in the header of blocks generated using PoW. When packaging a new block, the packager must try different nonce values again and again until the hash value of the block header satisfies the difficult requirement, for example, with several leading zeros.

The security and reliability of PoW are grounded in Assumption 2 in Section III-B. Once a packager gives a block meets the requirement, this means that he or she has spent enough computational resources to obtain the result. Anyone who wants to modify the chain needs to rechoose the right nonce value for every block from the modification point to the latest block, and he or she must do it faster than the extending speed of the longest chain, which is believed to be impossible as long as the computing power controlled by malicious nodes is no more than that of honest nodes.

Although PoW is advantageous in providing security, the energy consumption induced by mining is nonnegligible. In addition, to reduce forking and inconsistency, the block interval is set to an average of 10 min to keep the probability that concurrent blocks exist at a low level. Moreover, to prevent double-spending attacks, transactions in a block need to wait for confirmation from six extra subsequent blocks to reach a final commit, which cannot meet the requirements of high-frequency trading scenarios. Moreover, as the opportunity to win a competition is closely related to the CPU power of nodes, those with low CPU capacity will find it more difficult to succeed and naturally harder to obtain revenue from blocks, i.e., unfairness of user profits exists.

To solve these problems, many researchers have attempted to improve the PoW algorithm. Eyal *et al.* [29] proposed a consensus algorithm called Bitcoin-NG. In this algorithm, blocks are divided into two types: 1) key blocks and 2) microblocks. Participants still use PoW to generate key blocks, the owner of the latest key block wins the right to collect transactions in the blockchain network and pack them into microblocks, and he or she can keep them until the next key block is announced by other nodes. The difficulty of mining remains unchanged, while the number of blocks that contain transactions is increased.

Despite the optimized transaction processing system (TPS) earned from the separation of key blocks and microblocks, Bitcoin-NG needs an extra method to prevent microblock forks because they are generated cheaply and quickly by the leader. Byzcoin [30] combines the idea of Bitcoin-NG and practical Byzantine fault tolerance (PBFT) to make further improvements. In this article, key blocks are still packed through mining, but they contain a parameter w as the window size. The nodes that generate the key blocks covered by this window are the group that carries out PBFT with regard to the microblocks announced by the current leader. Each time a new key block appears, the window will move forward one step, possibly producing a new group of nodes to carry out PBFT for the next leader.

Except for the classic chain structure, some blockchains explore the use of novel structures to improve the TPS. Li *et al.* [31] employed a directed acyclic graph (DAG) to enhance the ability of the blockchain system to handle concurrent blocks. Forks are no longer unacceptable here; thus, the criteria for selecting the longest chain as the main chain changes to extending the chain to the subblock with the largest subtree. Each block in the main chain starts a new epoch, and other blocks referenced by the main chain block are pulled to the same epoch, and the epoch of blocks is used to determine the final sequence of transactions.

B. *Proof of Stake*

The concept of Proof of Stake (PoS) was first proposed by [32]. In PoS, nodes have an attribute called coin age, which is the product of the coins they hold and their holding time. The rights of packing a block are no longer entirely dependent on CPU capacity; the higher nodes' coin ages are, the easier it is for them to obtain the opportunity to generate a block. Once a node succeeds in packing a block, his or her coin age will be reset and need to be reaccumulated. The security of PoS is grounded in Assumption 1 in Section III-B; nodes with high assets are unwilling to destroy the consensus process, as the profits of malicious attacks cannot cover their financial loss. PoS, to some extent, mitigates the problem of resource waste under the PoW mechanism; thus, some platforms that use PoW as their consensus algorithm show a wish to switch to PoS, e.g., Ethereum [33], [34].

Compared with PoW, PoS is more energy efficient, but it also has drawbacks. For example, the linear growth of coin age over time may encourage nodes to increase the holding time of their coins, which can decrease the coins actually invested in consensus, thus destabilizing the foundation of security. Reddcoin [35] manages to mitigate this problem by adjusting the growth curve of coin age, making it increase more and more slowly, thus pushing nodes to actively participate in consensus rather than hoarding.

Ouroboros [36] combines verifiable secret sharing (VSS) and coin tossing to achieve guaranteed output delivery (GOD) in PoS. Time is divided into slots, and several slots become epochs. At the beginning of each epoch, every node generates a genesis block, recording the identities of the nodes who will participate in subsequent consensus together with their corresponding stakes, and a consistent random seed obtained using the techniques mentioned above is used to determine the leader of each slot in this epoch. These genesis blocks are kept in their local memory and, except for the first block, will not be sent to the blockchain. After that, using the random seed and the index of the current slot as inputs of a function that chooses a stakeholder with probability proportional to his or her relative stakes, nodes can know who the leader is and wait for new blocks unless they are the leader; if a node is the leader, it is his or her turn to pack a new block.

Instead of selecting a leader, Algorand [37] directly grants different blocks different priorities. In its design, users are considered to be several subusers related to how many coins they have. With a random seed, every node computes a hash value and a corresponding proof using the VRF. The ratio of the hash result to the hash maximum determines whether he or she is eligible to pack a block or not and how many subusers are chosen. The proof can be used for other nodes to verify the correctness. Then, every user with at least one subuser selected is qualified to propose a new block and assigns its priority value as the highest result by hashing the hash output concatenated with the subuser's index. Only the proposal with the highest priority will be accepted for this round, and its proposer needs to generate a new random seed for the next round through the VRF, taking the current seed and round number as inputs.

Bitshare proposed a Delegated PoS (DPoS) [38]. Each miner can vote for a delegate in light of his or her stakes, and those who received the most votes obtain the right to pack blocks. DPoS serves as a combination of PoW and PoS, effectively reducing energy consumption and optimizing the balance of stakes.

There are also some works that use other attributes to take the place of stakes or work, such as proof of authority [39], proof of vote [40], proof of majority [41], and proof of activity [42].

C. Practical Byzantine Fault Tolerance

In this section, we introduce some algorithms that reach consensus through multiple communications with other nodes to exchange for better energy efficiency.

Some of these algorithms originate from Paxos [43], and they can quickly complete consensus with tolerance only against crash faults, which means that nodes will not send error or contradictory messages. Raft [44] is an example. It uses heartbeats for synchronous information. When initializing, nodes are set as followers. They wait for heartbeats from the leader. If they do not receive a heartbeat and one of them triggers his or her timer, then he or she will convert into a candidate and ask others for votes to become a leader. Provided that he or she receives enough votes, he or she will take the responsibility of a leader, regularly sending data and heartbeats to others; otherwise, he or she will try it later only if he or she receives no heartbeats and there is a leader election during that time.

However, it is unreasonable to assume that there are only crash faults in real systems. Therefore, researchers have begun to explore new ways to make them resilient against Byzantine faults. PBFT [45] is a classic algorithm that satisfies this demand, where having up to 1/3 Byzantine nodes is tolerable. The consensus is implemented in three phases, which are called *preprepare*, *prepare*, and *commit*. In *preprepare*, the leader announces a message to all other nodes and then enters the *prepare* phase: every node checks the message received from the leader and forwards it to others if the validation passes. Once a node receives prepares for the same message from more than 2/3 different nodes, he or she can ensure that the majority received the same *preprepare* from the leader; thus, he or she enters the commit phase and sends commit messages to others. When a node receives commits for the same message from more than 2/3 different nodes, he or she knows that the majority has agreed on that message, i.e., consensus has been reached.

PBFT accomplishes a meaningful improvement from crash resilience to Byzantine resilience, but it also makes it necessary to communicate several times between every two nodes, saving computational costs while sacrificing communicational complexity, which limits the speed of consensus when the number of participants increases. Wang *et al.* [46] introduced credit for nodes to assist in recommending a few delegates to launch PBFT rather than the whole set. Jalalzai and Busch [47] and Lei *et al.* [48] attempted to reduce the overall complexity by optimizing the communications that take place during the view change process.

Albatross [49] divides blocks into two categories: 1) macro blocks and 2) microblocks. The former category is in line with PBFT; it does not store any transactions but contains a group of active packagers and a random seed used to generate the next block. The latter is revealed by each slot owner, i.e., an active packager is selected based on tokens that he or she paid when becoming active. This kind of block contains transactions and is distributed to others by the slot owner without additional communications. Moreover, one macro block is followed by several microblocks; thus, it can reach a faster consensus than pure PBFT.

Du *et al.* [50] proposed an algorithm called mixed Byzantine fault tolerance (MBFT), which separates all participants into several consensus groups: a high consensus group (HCG) and low consensus groups (LCGs). Transactions are allocated to these LCGs, and the leaders in each LCG conduct a process similar to PBFT to generate microblocks. All LCGs do this in parallel and send these microblocks to the HCG. Then, the nodes in the HCG verify each microblock they received, and the leader packs these legal microblocks into one large block and announces it to other members in the HCG. If this large block obtains enough positive feedback from the HCG, i.e., over 2/3 agreement from members, it will be linked to the chain.

III. Preliminaries and the System Model

A. Concepts

First, we introduce the definitions of some concepts used in our algorithm as follows.

1) *Transaction Pool:* The transaction pool is used to preserve the transactions to be packaged. The transactions in the pool are simulated and executed; in other words, they have already been endorsed and are considered legal transactions.
2) *Deposit Pool:* The deposit pool is a place where various deposits are stored. Once nodes put their assets in this pool, these assets can no longer be casually used by their owners until some specific events occur.
3) *Genesis Block:* This block is the first valid block generated in the system.
4) *Secret Keys:* Every node in the system has a pair of asymmetric keys called the public key and the private key. Messages encrypted by the public key can be decrypted only by the corresponding private key and vice versa. The public key of any node is visible to all the participants in the system, but the private key is known only to its owner.
5) *Packaging Blocks:* The packagers in the system choose a number of transactions in its transaction pool, together with the hash value of the previous block, the current timestamp, the depth of this block and the root of the Merkle tree, to generate this block's hash value and to sign this block with the corresponding packager's private key. We also call this process the package of blocks, denoted as <*PACK_BLK, prehash, t, depth, merkle-root,* [*tx*]>, where [*tx*] stands for the transactions contained by this block.
6) *Signing a Block:* After the generation of a new block, its packager will sign its hash value with the private key to generate a signature of the block, serving as evidence of the block's ownership, and the signature can easily be verified by any other nodes using the packager's public key. We denote a signed block as <*PACK_BLK, prehash, t, depth, merkle-root,* [*tx*]>$_{SKi}$, where SK_i is the private key of node *i*.
7) *Reputation List:* In our system, every node maintains a reputation list of other nodes, and every node will automatically adjust the value of nodes based on their

behaviors. This list indicates how much the node trusts other nodes, and it holds great significance for our consensus algorithm.

8) *Node Lotteries:* Given the same value, different nodes can draw different lotteries through a series of computations. Lotteries are verifiable and comparable; thus, they are used to determine which node is the next to obtain the right to pack a block.
9) *Lottery Pool:* The lottery pool collects all the lotteries generated by nodes from a given input in each round; thus, by checking the lottery pool, one can learn the lottery situation of the whole system and then judge whether he or she has won in this round.

B. Premise Assumptions

In this section, we offer and explain some basic assumptions for blockchain networks.

Assumption 1 (Rational Participants): Both malicious and honest nodes are rational. The former will not launch an attack if doing so will damage their own benefits, and the latter will try to maximize their profits within the regulations.

Assumption 2 (Secure Hash Function): The process of the hash function is irreversible, which means that one can only compute the hash value from a message and cannot infer the original message from a given hash output. A repeated hash of the same input yields the same output, and the probability that different hash inputs generate the same hash output is negligible.

Assumption 3 (Unbreakable Secret Keys): We generate keys using the Rivest–Shamir–Adleman (RSA) algorithm, and we assume that the secret keys of nodes are sufficiently secure and that no one has sufficient power to crack the relevant private key through the public key in a limited period of time.

Assumption 4 (Unforgeable Information): In VDC, we assume that no malicious node can impersonate another node or forge any message in the name of others because it does not know their private keys.

Assumption 5 (Arbitrary Behavior of Malicious Nodes): We cannot predict the behaviors of malicious nodes; they may operate honestly and may try to break the consensus individually or collusively.

Assumption 6 (Honest Majority): In the consortium, the number of honest nodes always exceeds that of malicious nodes.

Assumption 7 (Guaranteed Message Delivery): Any message sent by a node will arrive at its destination within a finite time interval, and even malicious nodes cannot intercept the delivery process.

C. System Model

A consortium is an organization established and managed by several companies. In that organization, no one is willing to hand over the privileges of management to others because others are usually not completely trusted and because anyone who masters the power of dominating the system tends to earn higher profits. Thus, joint management of the consortium, in which all members supervise each other to avoid malicious behaviors, is an acceptable solution.

The participants of a consortium are abstracted as nodes. Some of them come from the same company or from different companies. Each node is marked with an ID when he or she registers with the consortium and has his or her unique secret key pairs. As participants in the consortium, all members carry an amount of assets and may experience four different roles in the process of consensus.

Ordinary Nodes: Upon joining the consortium, every node is labeled an ordinary node. Ordinary nodes do not have permission to generate or verify blocks, but they can see the procedure of consensus, forward new blocks and obtain voting information.

Candidates: These nodes still have no power to package or verify blocks, but they have the potential to be elected packagers. Any ordinary node can become a candidate as long as he or she pays a certain number of deposits, and these deposits are frozen and unavailable unless he or she resumes being an ordinary node.

Packagers: Packagers are responsible for packaging new blocks, and they send the blocks generated to verifiers to perform verification. They are mainly elected by verifiers from candidates based on the reputation list. Only a packager who wins a lottery round, whom we call the winner, has the opportunity to pack a new block. We use N_p to denote the number of packagers.

Verifiers: Verifiers are mainly responsible for two jobs: 1) they elect packagers. After a specific number of legal blocks, they begin to select a new set of packagers from candidates based on their reputation lists of other nodes and 2) they vote on blocks. When receiving a new block from a packager, they begin to vote on that block to determine whether it is valid or invalid.

Verifiers are different from others in the consortium: they are nodes that possess high assets in their companies, which to some extent ensures that they are unwilling to cause damage to the consortium as long as they are rational. Similarly, we use N_v to denote the number of verifiers.

Messages sent by nodes in the consortium, especially packagers and verifiers, should be signed with the nodes' private keys. Signatures can be used to verify the source of a message to issue a reward or to accurately determine who is to blame when problems occur.

Although there are four roles in this model, nodes can hold only one identity at the same time. However, no one is able to possess an identity permanently; under some conditions, nodes can convert from one role to another: ordinary nodes can become candidates by simply paying deposits. Candidates can become packagers if they receive votes from verifiers during the process of packager election, or they themselves may choose to return to ordinary nodes and fetch the deposits they offered when they become candidates. Packagers can become verifiers when the new round of verifier election is executed, as long as they have more assets than some verifiers at that time, or they will be forced to return to being ordinary nodes if they are caught acting maliciously and be eliminated from the packager group. For verifiers, VDC sets a tenure for each group of

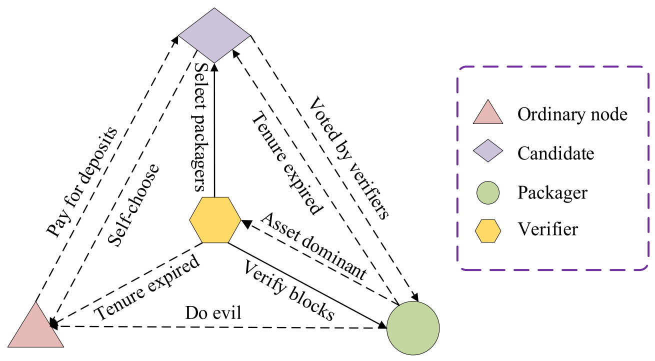


Fig. 1. Relationships among the four identities.

verifiers, that is, several legal blocks linked to the blockchain. When their tenure expires, all nodes in this group return to being ordinary nodes, and a new round of verifier election is conducted. This is also the time when some packagers with dominant assets can become verifiers. Fig. 1 shows the relationships among the four roles. The dotted lines indicate the transformation of identities, while the solid lines indicate voting behaviors.

IV. Algorithm Design

As introduced in Section II, many efforts have been made to design an excellent algorithm for blockchain platforms, but unfortunately, they all have their own weaknesses or are designed for specific scenarios. In PoW, billions of hash operations are executed per second just for a solution to a crypto puzzle, which is energy consuming and time inefficient with no other practical significance. Both PoS and PoW face the problem of privilege centralization. Byzantine fault tolerance (BFT) algorithms are usually restricted by the scale of participants, leading to low system capacity, and the strategy of appointing a leader is vulnerable to targeted attacks. Moreover, few of them are concerned about the fairness of user profits. In this section, we formulate in detail VDC, the algorithm we designed for consortium blockchain, which can achieve better fairness of user profits and time efficiency with acceptable energy consumption and without sacrificing security. First, we summarize the main parameters used in our algorithm in Table I.

A. Details of VDC

In VDC, time is divided into slots. Each slot has a packager to generate a block, and a certain number of nonempty slots compose an epoch. When initializing, every node that wants to join the consortium needs to send a register message in the form *<REGISTER, <id, assets, PK_i>$_{SKi}$, PKi>* to let others know his or her identity, how many coins he or she has and the correctness of his or her secret key pair. We use < > to denote a message and < >$_{SKi}$ to indicate the message signed by the private key of node *i*. If a node registers successfully, other nodes will set him or her an initial reputation value and add it to their reputation lists. Then, the detailed steps of VDC are as follows in Fig. 2.

TABLE I
Parameters Used in the VDC Algorithm

Symbols	Definitions
PK_i, SK_i	The public and private keys of node *i*.
N_v	The number of members in the group of verifiers.
N_p	The number of members in the group of packagers.
N_n	The number of nodes in the whole consortium.
N_a	The number of attackers in the whole consortium.
T_b	Length of an epoch and the tenure of verifiers.
D_c	The deposits paid to become a candidate.
D_b	The deposits paid by a verifier when voting for a block.
rpt_i	The absolute reputation of node *i*.
$rrpt_i$	The relative reputation of node *i*.
str1, str2	Constant string "*PACKAGER_ELECTION*" and "*LOTTERY_SEED*", respectively.
r1, r2	Segments of the random seed for drawing a lottery.
L_i, P_i	The lottery drawn by node *i* and its corresponding proof.
L_w, P_w	The lottery that wins in a round of lottery drawing and its corresponding proof.

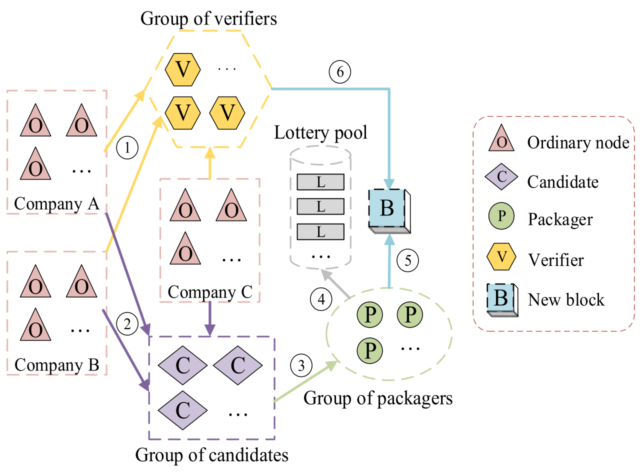


Fig. 2. Main steps of VDC.

Step 1 (Generation of Verifiers): At the beginning of each epoch, VDC selects N_v participants with dominant assets to form a group of verifiers, and these quotas are assigned to different organizations based on their relative assets. Assume that there are three companies A, B, and C, as shown in Fig. 2 and that their total assets are 10, 10, and 20, respectively. Then, their top 3, 3, and 6 nodes are selected as verifiers when N_v is set to 12. The tenure of these verifiers lasts until the end of this epoch.

Step 2 (Generation of Candidates): After the generation of verifiers, those that are not selected as a verifier can announce that they wish to become candidates using *<BE_CANDIDATE, id, D_c>$_{SKi}$*, where D_c is part of the assets they need to pay to make this announcement valid. All nodes that pay the required deposits form the group of candidates. Note that verifiers are not allowed to do this.

Procedure 1 *VoteForCandidate* (*Str1*, *Depth*, *SK_i*)

```
$c_i \leftarrow Encrypt_{SKi}(str1\|depth)$ // ciphertext
$loc_i \leftarrow hash(c_i)/2^{hashlen}$ // location
$j \leftarrow 1$ // index of candidates
while $loc_i \notin (\Sigma_{k=0}^{j-1}\text{rrpt}_k, \Sigma_{k=0}^{j}\text{rrpt}_k]$, do
    $j++$
return $j$
```

Step 3 (Election of Packagers): When the above two steps are finished, packagers are elected. Each member of the verifier group will encrypt a message that is the concatenation of a constant string *str1* = "*PACKAGER_ELECTION*" and the *depth* of the next block using his or her private key; others can validate its legitimacy using the corresponding public key. Then, the hash of the result dividing 2^{hashlen} will be used to decide which candidate to vote for. The pseudocode of this step is stated in Procedure 1.

In this pseudocode, $rrpt_j$ stands for the relative reputation of candidate j, where total is the sum of all candidates. We assume that j starts from 1 and that $rrpt_0$ is 0. After obtaining return value j, verifier i announces a message $<$*PACKAGER_ELECT, depth,* c_i, $j>_{SKi}$ to indicate that he or she decides to vote for candidate j.

Others can validate the message in three phases. First, they can check if the *depth* in this message is consistent with that of the block going to be packed; second, they can decrypt c_i with PK_i to see if the result equals $str1\|depth$; third, they can recalculate loc_i from c_i and check whether j is correct. If all these phases pass, this vote of verifier i is considered valid. Each verifier must vote for and can vote for only one candidate.

N_p packagers are chosen from the candidates based on the votes they gain, where N_p is a parameter that denotes the number of packagers; we simply set it equal to N_v. Therefore, we retain the attribute, coin age, to help the election of packagers when several verifiers vote for the same candidate, i.e., there are fewer than N_p candidates receiving a vote from verifiers.

Step 4 (Determination of the Slot Owner): Having a group of packagers, VDC needs to decide the slot owner for each slot in the current epoch, which can be done using a lottery drawing. Before that, we need to choose a random seed for the lottery drawing procedure, which consists of two parts, $r1$ and $r2$. $r1$ is the previous hash of the next block, which is empty for the genesis block, and $r2$ is computed collectively by all packagers.

For every packager k, he or she calculates ciphertext c_k using his or her private key SK_k : $c_k = \text{Encrypt}_{SKk}(\text{str2}\|\text{depth})$, where *str2* is another constant string "*LOTTERY_SEED*." Then, he or she announces it to both verifiers and packagers: $<$*ANNOUNCE_SEED*, *depth*, $c_k >_{SKk}$. Others can also check the correctness of c_k and store it locally if valid. When all these messages are received from packagers or time expires, they will compute $r2$ using the *XOR* operation of all the hash results of c_k. Finally, $r1\|r2$ is the random seed for the lottery drawing. The pseudocodes are shown in Procedure 2:

In Procedure 2, each packager k will use $r1$ and $r2$ as the random seed of the VRF to generate a lottery L_k and its corresponding proof P_k. Instead of revealing both of them, he or she just announces L_k to the verifiers and packagers: $<$*ANNOUNCE_LOT,* $L_k>$, and this message carries no signature information; thus, other people cannot know the ownership of this lottery at that time.

Procedure 2 *DrawLottery* (*Prehash, Str2, Depth, SK_k*)

```
for each packager $k'$, do
    $c_{k'} \leftarrow Encrypt_{SKk'}(str2\|depth)$
    $<ANNOUNCE\_SEED, depth, c_{k'}>_{SKk'}$
end for
$r1 \leftarrow prehash$
$r2 \leftarrow XOR_{k'=1}^{Np}\ hash(c_{k'})$
$< L_k, P_k > \leftarrow VRF_{SKk}(r1\|r2)$
return $<L_k, P_k>$
```

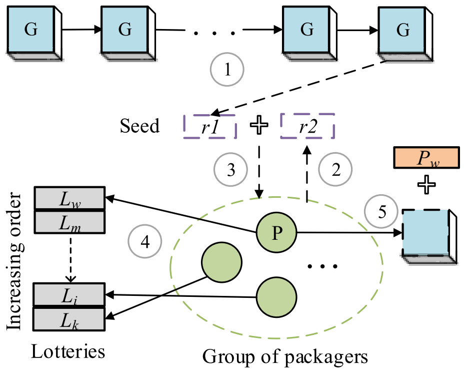


Fig. 3. Process of determining a slot owner.

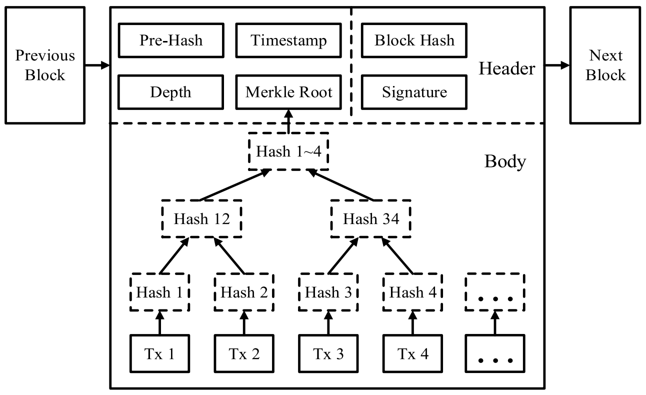


Fig. 4. Structure of a block.

After receiving all the lotteries belonging to this slot or time expires, the packagers and verifiers will store them in a vector and arrange them in increasing order. The smallest lottery wins this slot, and its producer will begin to pack a new block. Fig. 3 describes the process above.

Step 5 (Generation of New Blocks): After the process of drawing lottery, a specific packager k obtains the privilege of packing a new block with a common structure, as shown in Fig. 4. He or she immediately chooses several transactions from the transaction pool and packs them into a new block, $<$*PACK_BLK, prehash, t, depth, merkle-root,* [*tx*]$>$, together with the preblock's hash value, the timestamp, the root of the Merkle tree and other necessary messages. Of course, he or she needs to sign it with his or her private key SK_k. When all the steps above are completed, he or she sends a message to the verifiers and packagers, *<ANNOUNCE_BLK, id, <block>$_{SKk}$, P_k>$_{SKk}$*, including the new block that he or she generated and a proof to demonstrate his or her ownership of this slot.

Step 6 (Vote for New Blocks): Receiving an *<ANNOUNCE_BLK>* message from a packager, verifiers and other packagers treat it differently. A packager checks whether P_k contained in this message can validate the current smallest lottery $L_{\min}$; if it can, he or she simply does nothing and waits for decisions about this block from the verifiers; if it cannot, he or she ignores this message and waits for the real proof of $L_{\min}$ until a certain time expires, and then, he or she erases $L_{\min}$ from his or her local lottery pool. A verifier also first checks the proof; if he or she obtains a negative result, he or she addresses it as a verifier, but if the proof is proven to be correct, he or she begins to verify the block included in the message.

Verifier i extracts the block from *<ANNOUNCE_BLK>* if the proof attached here passes the examination. Then, he or she inspects whether the previous hash points to the latest block in the chain, whether the timestamp is correct, whether the Merkle root is correctly computed and whether the signature is valid, which means this block has not been modified at all. If no problems exist in these inspections, he or she will announce agreement on this block to every participant: *<APPROVE, <id, <block>$_{SKk}$, P_k>, D_b>$_{SKi}$*; otherwise, he or she will announce disagreement: *<REJECT, <id, <block>$_{SKk}$, P_k>, D_b>$_{SKi}$*. D_b is the deposit needed for a verifier to make this decision.

Notably, honest verifiers will vote for only one block received in each slot. Specifically, if an honest verifier receives two *<ANNOUNCE_BLK>* messages with the same *id and P_k* that are proven to belong to the real slot owner but different blocks are attached, such as *<block1>$_{SKk}$*, *<block2>$_{SKk}$*, he or she will vote for neither of them and then reveal these conflicts to others instead, reminding them that there may be double spending of the current slot owner. Malicious verifiers can treat it casually, generating *<APPROVE>* or *<REJECT>* messages for different *<ANNOUNCE_BLK>* and sending each node a specific message from them, or they can simply do nothing.

Step 7 (Judgment of New Blocks): After receiving *<APPROVE>* and *<REJECT>* messages from the verifiers, each node begins to check whether they are correctly signed by the verifiers, and only those that have been correctly signed will be kept locally. Then, they count the number of *<APPROVE>* and *<REJECT>* messages. If more than half of the verifiers agree on a block proposed by the right slot owner, nodes will link this block to their local blockchain; otherwise, if the majority of verifiers disagree with regard to this block, they will link a special empty block. This block contains no transactions, the timestamp field is copied from its previous block, the depth increases by 1, the Merkle root and signature remain *NULL*, and the hash information is computed. Although the empty block carries no signature information, each field can be inferred from the previous block, even when the previous block is also an empty block. As the blocks already in the chain are consistent, the empty block creates no inconsistency.

Similar to the last step, if honest nodes detect different blocks generated by the correct slot owner from either *<APPROVE>* or *<REJECT>* messages, they will accept none of these blocks as valid, even if someone receives agreement from more than half. Instead, they will choose to link an empty block, label the current slot owner as malicious and reveal the conflicts to warn others that double spending exists.

In the last case, if the *<APPROVE>* and *<REJECT>* messages that a verifier received are less than 50%, he or she will declare it and ask other verifiers for the messages that they received. If no one received enough judgments, the verifiers will also link an empty block.

B. Incentives and Punishments

In this section, we introduce the reward and punishment strategies used in VDC.

Different from some previously presented works that use only money or credit as the reward, we adopt both of them to motivate participants to behave honestly and impose penalties when they are caught damaging the consensus.

This usually takes place after step 7 is finished. If the new block proposed by the slot owner is ultimately judged to be invalid or he or she is found to be carrying out double spending, then other nodes will lower the reputation value of this packager by a predefined span, and the deposits that he or she paid when he or she became a candidate will be confiscated to reward other honest nodes except the verifiers because ordinary nodes did not pay these deposits when they became verifiers. Otherwise, if the block is linked to the chain, that packager will receive the tax offered by this block as financial profits, and his or her value on others' reputation lists will increase.

For a verifier, if his or her judgment about a new block is the opposite of that of the majority, i.e., over 50%, he or she is thought to have made a wrong decision, and the deposits that he or she paid for this decision will be punished to reward those who made accurate judgments. Additionally, his or her value on others' reputation lists will decrease. In contrast, when his or her judgment is consistent with that of the majority, his or her reputation will increase, and he or she will be able to fetch his or her deposits D_b, but no financial profits will be earned. Because verifying a block is an easy task, verifiers do it voluntarily for the sake of protecting their own interests, as they hold most of the assets in the consortium.

C. Conversion Between Identities

We briefly described the conversion among different identities in Section II; now, we state it more in depth in this section and provide pseudocodes formulating the full version of VDC with the incentive mechanism and identity switching integrated.

Identity switching often occurs at the beginning of each epoch. As shown in Fig. 5, each block is situated in a slot, and T_b nonempty slots compose an epoch. This parameter is

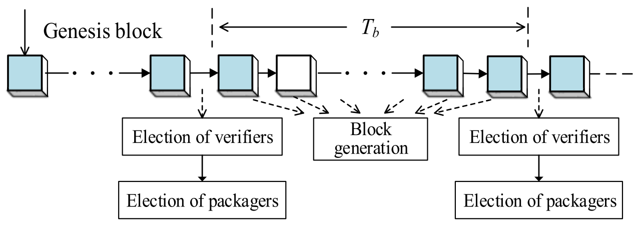


Fig. 5. Tenure of verifiers and packagers.

also used to denote the tenure of verifiers; in other words, the lifetime of a node's identity as a verifier equals an epoch.

When an epoch ends, verifiers reset to ordinary nodes, and a new round of verifier election is going to begin. However, before that, other nodes also need to reset to ordinary because after the consensus reached in the last epoch, some packagers in that epoch may have surpassed some verifiers in assets; thus, they should take the responsibility of being verifiers and leave the opportunity to become a packager to other participants. Then, a batch of ordinary nodes with dominant assets will form the next group of verifiers.

Those who did not become a verifier can choose to freeze D_c deposits to be a candidate or simply remain ordinary nodes, which means giving up the opportunity to be selected as a packager in the next epoch.

In the next phase, verifiers elect from candidates to form the group of packagers, following the regulations elaborated in the previous section.

In general, an honest packager can survive until the end of its epoch. However, if a malicious packager packs an invalid block and is caught in step 7, in addition to the financial and reputational punishments, his or her identity will no longer be admitted, and he or she will be removed from the group of packagers, leading to an identity conversion from packager to ordinary node. If all these packagers are malicious, they will be forced to return to being ordinary nodes and be removed from the group one by one, as long as they keep violating the principles. If the number of valid new blocks does not reach T_b when the size of the packager group equals zero, then the verifiers will re-elect a new group of packagers from the remaining candidates, triggering another identity conversion from candidate to packager. This is a special situation where the election of packagers does not occur closely behind the election of verifiers and candidates.

The division and conversion of identities grant VDC unique advantages in some application scenarios. For example, VDC can be applied to record charging transactions in the IoV. In this scenario, charging piles are the main participants in the consortium, which are responsible for collecting the transactions occurring on them and sending these transactions to the consortium for consensus. The practice of classifying nodes can effectively reduce the number of charging piles actually participating in the consensus process, accelerate it and enhance the system's capacity. Meanwhile, identity switching can also prevent some charging piles from monopolizing the privilege of consensus and obtaining rewards, leading to a much better fairness of user profits. The characteristics of VDC in decentralization and time efficiency make itself suitable for various specific fields, such as autodriving [51], crowd sensing [52], [53], and smart healthcare [54], where enormous perception data need to be handled in time.

We now give a full version of VDC, and the pseudocodes are presented in Algorithm 1:

Algorithm 1 VDC
```
while transaction pool is not empty, do
  if an epoch expires
    reset identities
    generate the group of verifiers
    generate the group of candidates
    for each verifier $i$, do // elect packagers
      VoteForCandidate ($str1$, $depth$, $SK_i$)
    end for
  else if size(the group of packagers) = 0 //re-election
    for each verifier $i$, do // elect packagers
      VoteForCandidate ($str1$, $depth$, $SK_i$)
    end for
  end if
  for each packager $k$, do //determine the slot owner
    DrawLottery ($prehash$, $str2$, $depth$, $SK_k$)
    add $L_k$ to the lottery pool
  end for
  while size (lottery pool), do
    slot owner reveals a block and proof
    for each verifier and packager, do
      if proof is valid //get the right slot owner
        break while
      else if time expires //receive no valid proof
        remove the min $L$ from the lottery pool
      else
        ignore that block and proof
    end for
  end while
  if size(lottery pool) //the correct slot owner exists
    for each verifier, do
      vote for that new block
      announce vote info to every node
    end for
    for each node, check
      if valid <APPROVE> exceeds 50% && no double spend
        link the new block
      else
        link an empty block
        if valid <REJECT> exceeds 50% ||double spend
          remove the slot owner
        end if
      end if
      update reputation and asset info about nodes
    end for
  else //no legal slot owner appears
    for each node, do
      link an empty block
    end for
  end if
end while
```

V. Theoretical Analysis

In this section, we theoretically analyze the performance and security of VDC.

A. Performance

1) Fairness of User Profits: VDC is an algorithm that takes into account fairness. Here, fairness means the balance of chances or rights for users to pack new blocks and earn from them; thus, we specifically name this attribute the fairness of user profits

$$\Pr(i) = \left(1 - \left(1 - \frac{rpt_i}{\sum_{j=1}^{N_c} rpt_j}\right)^{N_v}\right) / N_p. \tag{1}$$

The above equation gives the lower bound of probability that candidate i becomes a slot owner. On the right, the former half indicates the probability that candidate i receives at least one vote from the verifiers, i.e., the probability he or she is elected as a packager. The latter half indicates that he or she wins a slot and obtains the right to pack a block because the lottery drawing process can be treated as roughly uniform, which is ensured by the nature of hash operation. The reason it is the lower bound is that those who receive no votes can also have the chance to become a packager as long as there are residual quotas, which can occur when one candidate receives more than one vote, as explained above.

VDC is advantageous because (1) has nothing to do with power and stakes, which are the main reasons the fairness of profits is not well guaranteed in PoW and PoS. Moreover, it is unnecessary for nodes to wait for several view changes to take over the right, as in PBFT, where each node may spend a long time holding the right, especially when faults rarely occur.

2) Time Efficiency: The time consumption of a consensus algorithm is often related to two points: 1) computational tasks and 2) communicational complexity.

We use a lottery drawing instead of mining to determine the "Leader," which is an absolutely distinct method. The greatest difference is that the method consumes constant number of hash operations to draw a lottery rather than continuing to try until the solution is found. Rational participants will not draw lots repeatedly in each slot because the random seed is a fixed value in each slot and the same input always generates the same output.

Regarding communicational complexity, the classification of identity helps decrease the number of nodes engaged in each step. Each verifier reveals vote information about the candidates in each epoch, inducing total $O(N_v N_n)$ complexity, and each valid block shares $O(N_v N_n / T_b)$. Each packager reveals the lottery seed and lottery to the verifiers and other packagers in each slot, inducing $O(N_p N_v)$. The slot owner proposes a block, inducing $O(N_n)$. The verifiers vote for a new block, inducing $O(N_v N_n)$. Therefore, the total complexity yielded in one slot or one block is $O(N_p N_v + N_v N_n)$, which is smaller than $O(N_n^2)$ in PBFT.

3) Energy Consumption: In blockchain, the main source of energy consumption is believed to be hash operations. Therefore, for those relying on mining, a large amount of CPU resources is spent by each miner on solving mathematical problems, while this consumption is not an issue in VDC.

Although VDC is much better in terms of energy consumption than algorithms that employ mining, it yields extra hash operations for the lottery drawing compared to PBFT: for each block, each packager computes a random seed for the VRF, inducing a small constant number of hash $O(1)$; then, the packager uses this seed to draw a lottery, inducing another constant $O(1)$. For each slot, the hashes induced by lottery drawings are $O(N_p)$. Thus, the total number of extra hash operations yielded by lottery drawings remains constant for each block and each packager only, which is considered negligible and will not impose much of a burden on participants.

B. Security

In this section, we illustrate the resistance of VDC to some prevalent attacks.

1) Double Spending: Double spending is a common type of malicious behavior, especially in algorithms that require no PoW. As in PoS, rational participants will choose to extend all branches simultaneously when they appear because each branch will provide them with an equal number of digital currency and almost nothing to lose, which is known as "nothing at stake."

Only packagers in VDC can have the opportunity to carry out double spending because no other identities can provide a legal lottery and its proof. While a malicious packager will not directly send an honest node to different legal blocks, this situation can be detected immediately, and consensus will switch to the next slot by linking an empty block.

If there is any method for a malicious packager to succeed in double spending, it must be collusion with malicious verifiers. We assume that there are N_v^a malicious verifiers and that the current slot owner is also malicious; they know each other's real identity and collude to try double spending. This type of packager packs two distinct valid blocks that conflict with each other, and then, he or she sends both to malicious verifiers while sending one to some honest verifiers N_v^1 and the other to the remaining verifiers N_v^2. The malicious verifiers know which block is sent to which verifiers, vote for the corresponding block and send *<APPROVE>* based on step 6. If N_v^a N_v^1 N_v^2 equally share N_v, then N_v^1 will see *<APPROVE>* messages from 2/3 for block 1, and N_v^2 will see *<APPROVE>* messages from 2/3 for block 2, which exceeds the threshold of 50%. However, double spending still will not succeed because N_v^1 also has 1/3 *<APPROVE>* messages for block 2 from *N2 V* and N_v^2 has 1/3 *<APPROVE>* messages for block 1 from N_v^1. Then, all honest verifiers will detect two different legal blocks from the same packager, they will know double spending happened, and they will label this packager malicious and link an empty block.

What if N_v^1 or N_v^2 is zero? This means that the packager generated two blocks but sent just one to all honest verifiers. Clearly, the verifiers will reach a consensus on this unique block. In another case, where the packager sends N_v^1 block 1 and sends N_v^2 nothing, if malicious verifiers announce *<APPROVE>* for block 2 to N_v^2, double spending is detected, and N_v^2 will notify others; otherwise, it becomes the situation described at the beginning of this paragraph.

In addition to double spending, if the slot owner is honest, malicious verifiers will collude to deliberately announce wrong votes to mislead honest nodes. After steps 6 and 7, each node should have N_v^a wrong votes and $N_v - N_v^a$ correct votes. To make a right decision, we need the following:

$$\begin{cases} N_v^a/N_v < 50\% \\ \left(N_v - N_v^a\right)/N_v > 50\%. \end{cases} \tag{2}$$

The above equation means that as long as the proportion of malicious verifiers does not exceed 50%, VDC can operate normally. Notice that the attacker ratio in the group of verifiers is no more than that in the whole consortium, as richer participants are less likely to damage the consensus. We make an extension such that VDC can tolerate up to 50% malicious nodes.

2) Selfish Packing: The concept of selfish packing originates from selfish mining, an attack method used in PoW to reverse a chain with less than 51% CPU power [55].

Because VDC does not employ mining to generate a block, we replace the name with selfish packing, which defines the behavior in which a slot owner packs different blocks and chooses the block that can bring him or her the best profits.

Ideally, when a packager wins a round of lottery drawing, he or she is going to pack and announce a new block before time expires, and the hash of this block will be used as a seed for the next round of lottery drawing, i.e., $r1$. In fact, no rational packager will simply pack a block and reveal it immediately; instead, he or she will try different blocks and calculate his or her next lottery in advance before his or her deadline arrives, revealing the block that leads to the minimum lottery, which can greatly increase his or her chance of being the slot owner continuously. Especially for those with a powerful CPU capacity, it is likely that they will dominate the block package procedure by carrying out selfish packing.

To prevent selfish packing, $r1$ can serve as only half of the random seed, and another half, $r2$, is determined by all packagers collectively and computed after the judgment of the current block. The combination of $r1$ and $r2$ makes carrying out selfish packing meaningless because $r2$ cannot be inferred from $r1$ and a small lottery from $r1$ does not mean a small lottery from $r1\|r2$. For this reason, we introduce $r2$ at the cost of inducing extra computational and communicational complexity.

3) Targeted Attacks: Targeted attacks are seldom discussed in consensus algorithms, but they exist, particularly in algorithms where the leader is appointed or can be inferred from the current context. PBFT and some instances of PoS are vulnerable to this attack.

These algorithms think that targeted attacks can be handled by changing to a different leader when the current leader is corrupted, but this holds only under the assumption that attackers cannot immediately corrupt a leader. If this condition is not satisfied, such algorithms cannot run normally. For example, if an attacker has the ability to immediately corrupt a node, he or she can instantly control the leader of PBFT after each view change is finished and cause another round of view change. This can be kept going if he or she wishes, and consensus will fall into a dead circle.

The same situation will not occur for VDC. The VRF in a lottery drawing can keep the real identity of the slot owner uncertain in the lottery revealing stage until the proof revealing stage, but when the attacker receives the correct proof, it is too late to corrupt this packager because the new block has already been announced by him or her and the next slot owner is probably not him or her. The only thing that this attacker can do is corrupt a packager randomly before the proof is revealed, with the probability shown in the following equation:

$$\Pr(\text{corrupt the right packager}) = 1/N_p. \tag{3}$$

However, even if the attacker immediately corrupts the first packager, this is still not enough. When the revelation time of the proof expires, nodes will remove the lottery from their lottery pools, wait for the proof and block corresponding to the second smallest lottery and so on. Thus, the probability that this attacker successfully corrupts this slot is stated in (4), and the probability that he or she successfully corrupts consecutive n slots is stated in (5)

$$\Pr(\text{corrupt the slot}) = 1/\left(N_p!\right). \tag{4}$$

$$\Pr(\text{corrupt } n \text{ slot}) = \left(1/\left(N_p!\right)\right)^n. \tag{5}$$

From the equations above, we conclude that VDC has a strong resistance to targeted attacks. In addition, it will become further strengthened as the number of packagers increases.

VI. Simulation Results

In this section, we conduct simulations to prove the correctness of our analysis in Section V. First, we observe the performance of VDC under different parameters. Then, we choose PBFT, a classic algorithm for consortium blockchain, and MBFT, a state-of-the-art algorithm, as comparison objects to demonstrate the advantages of our algorithm.

In the simulations below, the total number N_n is set to 50, and T_b is set to 10 legal blocks. Moreover, we monitor the average number and standard deviation (SD) of valid blocks packed by each node, together with the SD of node assets, to make an intuitive assessment of the fairness of user profits.

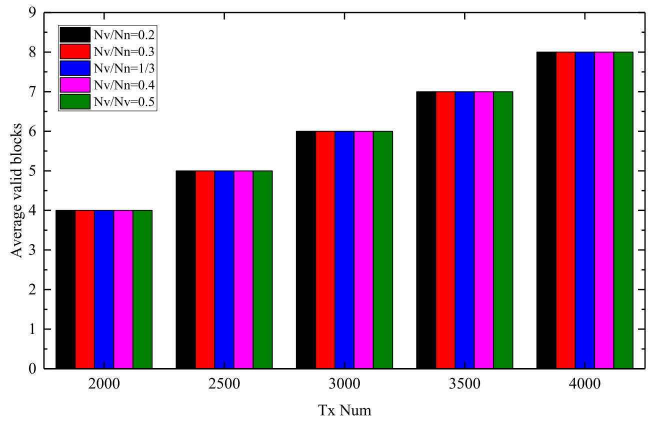
A. Influence of Parameters

From the equations given in Section V, we find that the parameters N_v and N_p have the greatest influence on the performance of VDC, and because in our design N_v equals N_p, we simply refer to the ratio of verifiers, i.e., N_v/N_n.

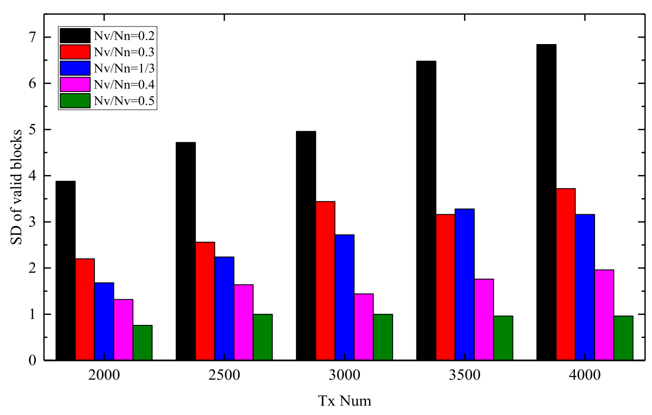
Fig. 6 shows the indicators we used to demonstrate the fairness of user profits. From Fig. 6(a), we conclude that the total number of blocks generated is consistent; it is a natural result because the size of a block and the transactions processed are the same. Fig. 6(b) indicates that these valid bocks are packed more evenly by all the participants as N_v increases, and Fig. 6(c) further proves this result. This conclusion is consistent with the theoretical analysis in Section V.

However, we need to emphasize that increasing N_v not only provides better fairness but also places a burden on energy consumption and time efficiency.

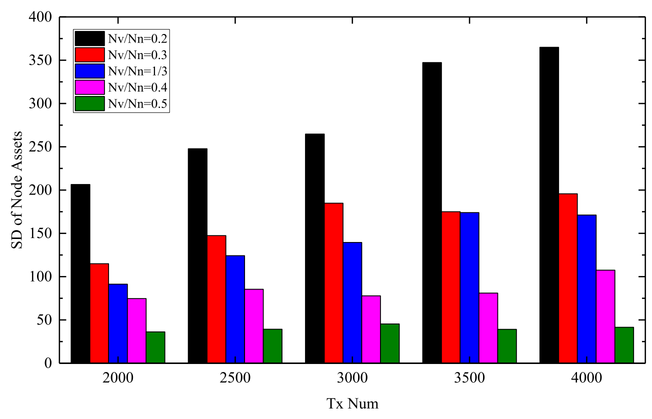
Fig. 7 shows the energy and time performance of VDC with different N_v/N_n ratios tested. The negative effects caused by the increase in N_v are obvious in this figure. The analysis



(a)

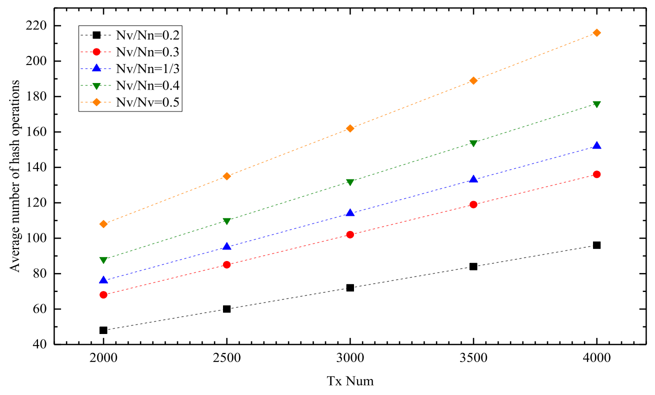


(b)

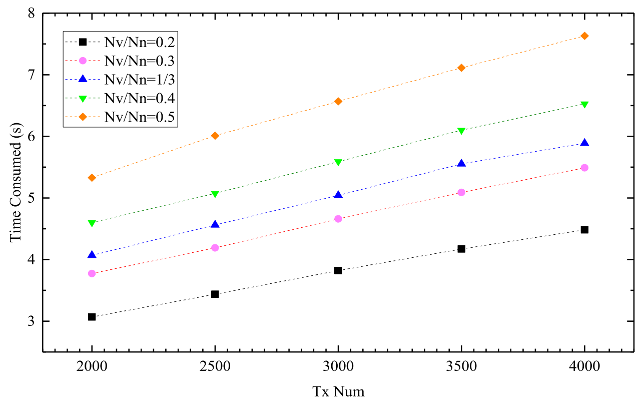


(c)

Fig. 6. Fairness under different N_v/N_n. (a) Average number of valid blocks packed by each node. (b) SD of the valid blocks of each node. (c) SD of node assets.



(a)



(b)

Fig. 7. Extra burden induced by increasing N_v. (a) Average number of hash operations that each node performs. (b) Total time spent for consensus on transactions.

in Section V also explains these phenomena: the hash operations yielded by lottery drawings relate to the number of N_p; when more packagers are elected, there will be more nodes to carry out a lottery drawing in each slot, leading to higher energy consumption. This reason can also be applied to the time cost because the total communicational complexity of VDC is $O(N_pN_v + N_vN_n)$. As N_v increases, more communications are needed to share information, which in turn results in a longer period of time to reach consensus.

Since the parameter N_v can both improve and worsen the performance of VDC, it should be adjusted based on the corresponding application requirements, whether fairness sensitive or energy and time sensitive. In the rest of this section, we adopt a compromise choice, i.e., set $N_v/N_n = 1/3$. Here, as N_n is 50, the probability of a successful targeted attack for a slot, i.e., the result of (4) with $N_p = 17$, is completely negligible.

B. Effect of the Incentive Mechanism

In this section, we simulate a scenario where malicious participants exist to see whether the incentive and punishment mechanisms work effectively.

We first set 30% of the participants as malicious, i.e., $N_a/N_v = 0.3$, and monitor the assets of each before and after the consensus of 3000 transactions using VDC; the results are shown in Fig. 8.

The figure above clearly shows that malicious nodes will have their assets damaged, and the more frequently they try, the more losses they will suffer. The reason is that each time they make a vital decision for consensus, the amounts of their deposits are combined and cannot be fetched when they are caught cheating.

Then, we try different N_a values and inspect the ratio of malicious nodes in the group of verifiers after each round of

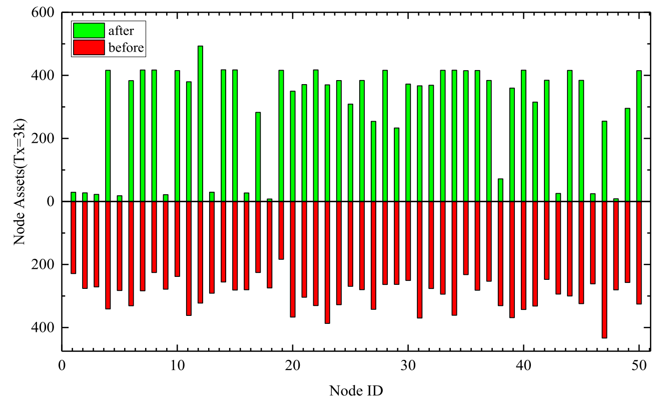


Fig. 8. Changes in assets when malicious nodes exist.

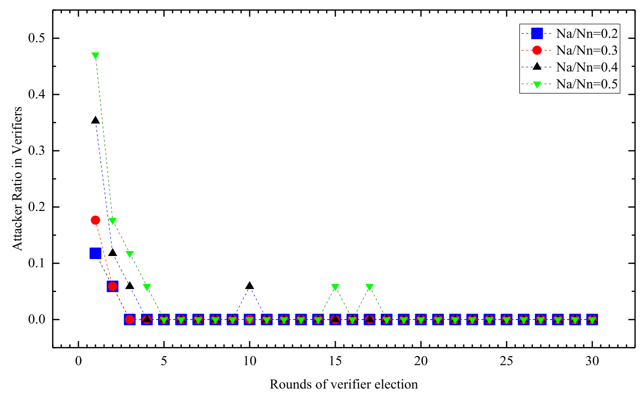
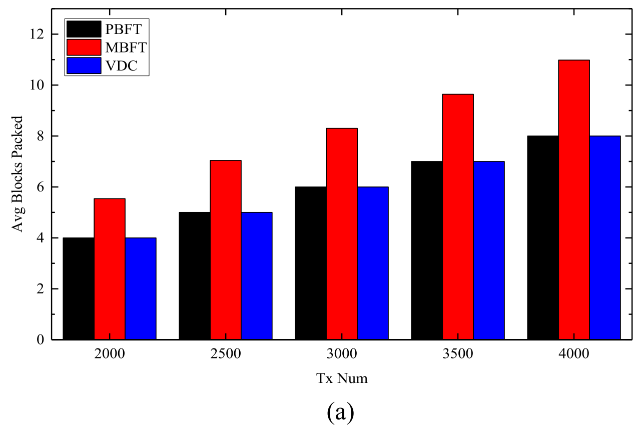


Fig. 9. Malicious ratio of verifiers as consensus continues processing.



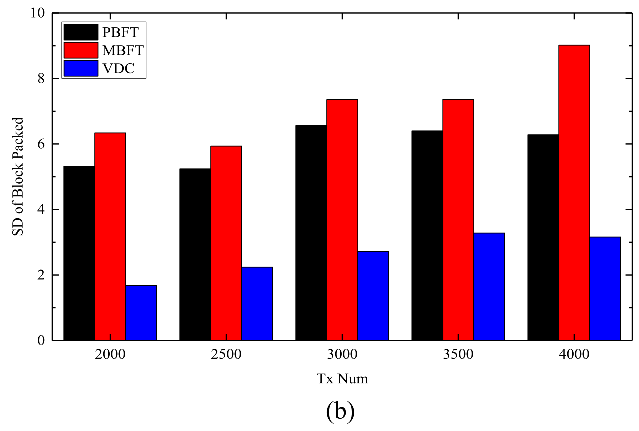


Fig. 10. Comparison of fairness in packing blocks. (a) Average number of blocks packed by each node. (b) Corresponding SD.

verifier election. Data are used to draw Fig. 9, where 0.2, 0.3, 0.4, and 0.5 are tested for the N_a/N_n ratio.

The result is in line with our analysis: as long as the malicious ratio in the whole consortium does not exceed 50%, VDC can guarantee that the ratio of verifiers stays under 50% and decreases to a lower level when consensus continues processing. The reason is that malicious verifiers will suffer from asset damage, lose their advantages in verifier election and gradually be replaced by other honest nodes.

There is a detail in Fig. 9; the higher the N_a/N_n is, the more rounds VDC needs to purify the verifiers. The reason is that there is little difference in the speed of asset accumulation for honest nodes under distinct values of N_a/N_n, and each honest node can take the place of only one malicious node, but there are more malicious nodes with high assets to overcome. Thus, more rounds are needed. However, this does not violate the fact that VDC can tolerate malicious participants amounting to 50%.

C. *Comparison With PBFT and MBFT*

Now, we give the last part of the simulations and compare VDC with PBFT and MBFT in terms of the fairness of user profits, time efficiency and energy consumption.

1) Fairness of User Profits: Considering that the number of large blocks in MBFT plus microblocks will exceed that in VDC and PBFT when they deal with the same quantity of transactions, this may lead to inaccuracy if we use only average blocks and the corresponding SD to illustrate fairness. Thus, for the sake of rigor, we also assume that leaders in PBFT and MBFT will earn taxes from blocks, even though they do not originally have financial rewards. In addition, in MBFT, the taxes in a microblock are shared equally by the leader in its LCG and the HCG because they both make contributions to consensus on this microblock. We set the number of LCGs to 3 in the simulations as follows.

Fig. 10 compares the fairness of profits in terms of packing blocks. VDC is undoubtedly superior to PBFT, but the same conclusion cannot be drawn between VDC and MBFT for the reason stated above. Therefore, Fig. 11 provides a supplementary comparison in terms of assets.

Combining the two figures, we see that MBFT is fairer than PBFT, as the leaders in each LCG can earn assets simultaneously. However, within each LCG, the identity of leader, is held by one node for a long time, rather than frequently switching as in VDC. Therefore, neither PBFT nor MBFT can match VDC in terms of the fairness of user profits.

2) Time Efficiency: In Fig. 12, we compare the consensus time after processing quantities of transactions using the three algorithms.

It is easy to understand that MBFT and VDC are superior to PBFT because the communicational complexity of PBFT is up to $O(N_n^2)$, while VDC and MBFT have reduced it to a smaller value. However, is MBFT really superior to VDC

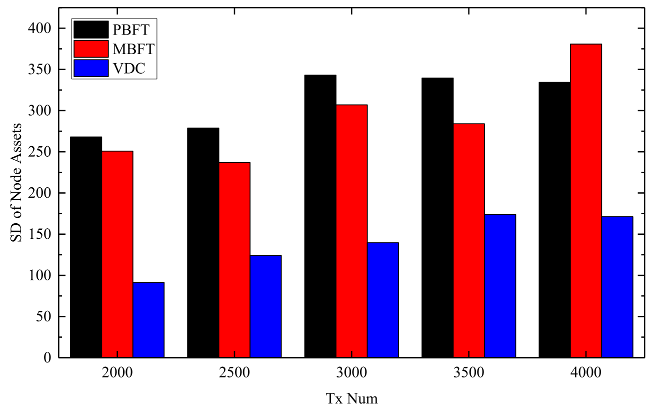


Fig. 11. Comparison of the fairness of assets.

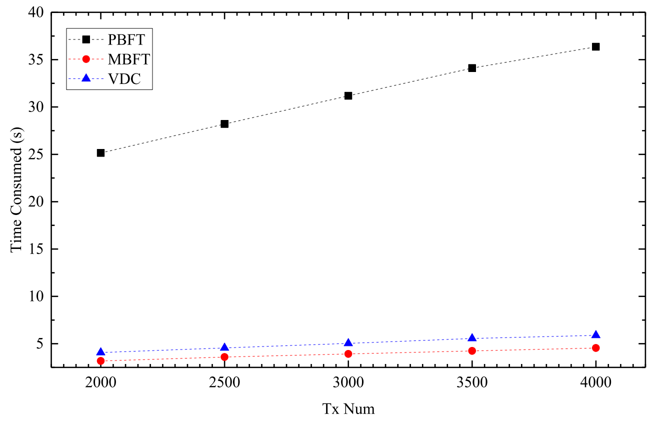


Fig. 12. Comparison of the time consumed.

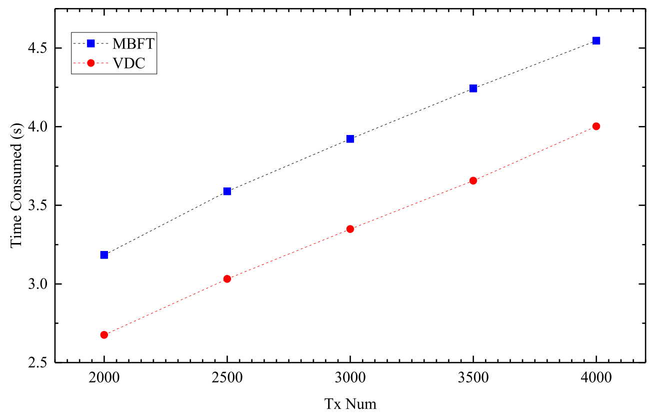


Fig. 13. Time cost when VDC and MBFT have the same size of blocks.

in terms of time efficiency? The number of LCGs can greatly influence the time consumption of this algorithm; the more LCGs there are, the larger a large block will be. Here, as we set 3 LCGs, a large block in MBFT is 3 times as large as that in VDC. What if we increase the blocks in VDC to the same size as the large blocks in MBFT? The results are shown in Fig. 13.

Fig. 13 proves that under the same size of chain blocks, VDC is more time efficient than MBFT. Although the parallel consensus in LCGs improves the performance of MBFT compared to PBFT, the leader in the HCG must spend

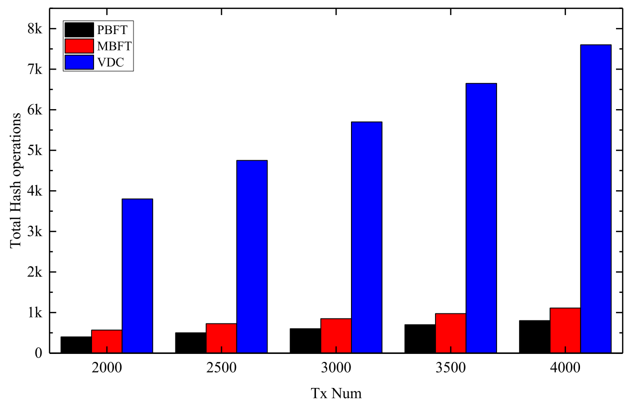


Fig. 14. Total hash times of each algorithm.

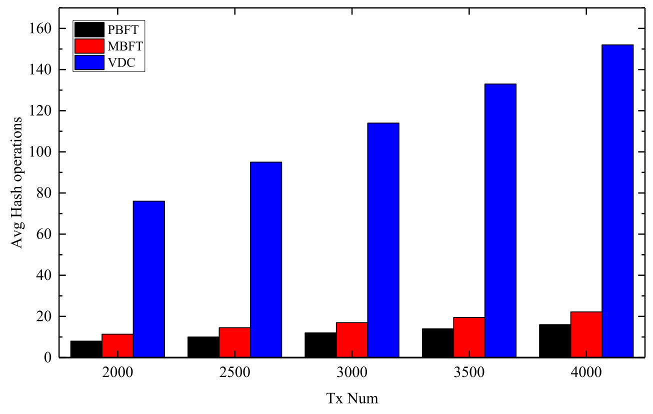


Fig. 15. Average hash times of each algorithm.

more effort verifying whether the microblocks received are valid individually and collectively. Clearly, these verifications take time.

3) Energy Consumption: As stated above, hash operations are considered to be the main source of energy consumption. Thus, we compare this property of the three algorithms by counting how many hash operations are performed.

We display the total number of hash operations that occurred in the entire consortium in Fig. 14, where MBFT exceeds PBFT and VDC exceeds MBFT. Regarding the former result, MBFT needs to generate additional large blocks, yielding more hash operations. The latter result is due to the lottery drawing mechanism.

Although Fig. 14 shows that of the three algorithms, VDC performs the most hash operations, the number of hash operations is far less than that needed by PoW to reach consensus, where PH/s represents 10^{15} hashes per second and is used as the unit of measurement [56]. Moreover, Fig. 15 shows the average hash times performed by these algorithms; no more than 200 hash operations are needed for each participant to reach consensus on 4000 transactions using VDC. Even a smart phone can easily achieve such a result, which, we think, can be considered energy efficient.

VII. Conclusion

This article focused on designing a consensus algorithm for consortium blockchain to realize better fairness, time efficiency, and lower energy consumption without sacrificing security. We divide the nodes into four categories, restricting their authority to avoid monopolies and lowering the complexity of communications. Then, based on the VRF, we propose an operation called lottery drawing, which can efficiently determine who is the slot owner and effectively protect him or her from targeted attacks; moreover, the frequent switching of slot owners means that our algorithm has a better fairness of user profits. In addition, we take both assets and reputation as incentives and punishments to encourage participants to behave honestly, and those who act maliciously will see their interests damaged both financially and reputationally.

First, we describe the main steps of our VDC algorithm in detail. Then, we theoretically analyze its performance in terms of the properties mentioned above and prove its performance through simulations. We test different parameters to investigate their influence on the performance of VDC and choose a moderate parameter. Then, we compare VDC to two algorithms, PBFT and MBFT, and the simulation results show that our design is superior in terms of the fairness of user profits (the SD of node assets decreases by 50%–60% and 40%–55%, compared to PBFT and MBFT, respectively), time efficiency (approximately 80% faster than PBFT and 15% faster than MBFT), and resilience against targeted attacks [cutting the probability of success from almost 100% to $1/(N_p!)$, i.e., (4)], with an acceptable extra cost in terms of energy consumption and without sacrificing the threshold of fault tolerance.

In the future, we will try to enhance VDC to make it adaptable to highly complex scenarios, such as those in the public chain environment, where Assumption 7, i.e., guaranteed message delivery, is no longer ensured.

References

[1] H. Massias, X. S. Avila, and J.-J. Quisquater, "Design of a secure timestamping service with minimal trust requirement," in *Proc. 20th Symp. Inf. Theory Benelux*, 1999, pp. 1–8.

[2] S. Haber and W.-S. Stornetta, "Secure names for bit-strings," in *Proc. 4th ACM Conf. Comput. Commun. Security*, 1997, pp. 28–35.

[3] R.-C. Merkle, "Protocols for public key cryptosystems," in *Proc. IEEE Symp. Security Privacy*, 1980, pp. 122–134.

[4] X. Huang, C. Xu, P. Wang, and H. Liu, "LNSC: A security model for electric vehicle and charging pile management based on blockchain ecosystem," *IEEE Access*, vol. 6, pp. 13565–13574, 2018.

[5] G. Pulkkis, J. Karlsson, and M. Westerlund, "Blockchain-based security solutions for IoT systems," in *Internet of Things A to Z: Technologies and Applications*, Hoboken, NJ, USA: Wiley-IEEE Press, 2018.

[6] Z. Li, J. Kang, R. Yu, D. Ye, Q. Deng, and Y. Zhang, "Consortium blockchain for secure energy trading in industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 14, no. 8, pp. 3690–3700, Aug. 2018.

[7] O. Alphand *et al.*, "IoTChain: A blockchain security architecture for the Internet of Things," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2018, pp. 1–6.

[8] J. Wang, M. Li, Y. He, H. Li, K. Xiao, and C. Wang, "A blockchain based privacy-preserving incentive mechanism in crowdsensing applications," *IEEE Access*, vol. 6, pp. 17545–17556, 2018.

[9] T.-T. Kuo and L. Ohno-Machado, "Modelchain: Decentralized privacy-preserving healthcare predictive modeling framework on private blockchain networks," 2018. [Online]. Available: arXiv:1802.01746.

[10] A. Dorri, S.-S. Kanhere, R. Jurdak, and P. Gauravaram, "Blockchain for IoT security and privacy: The case study of a smart home," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops*, 2017, pp. 618–623.

[11] O. Novo, "Blockchain meets IoT: An architecture for scalable access management in IoT," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1184–1195, Apr. 2018.

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

[13] P.-K. Sharma, S. Singh, Y.-S. Jeong, and J. H. Park, "DistBlockNet: A distributed blockchains-based secure SDN architecture for IoT networks," *IEEE Commun. Mag.*, vol. 55, no. 9, pp. 78–85, Sep. 2017.

[14] P.-K. Sharma, M.-Y. Chen, and J.-H. Park, "A software defined fog node based distributed blockchain cloud architecture for IoT," *IEEE Access*, vol. 6, pp. 115–124, 2017.

[15] G. Sun, M. Dai, F. Zhang, H. Yu, X. Du, and M. Guizani, "Blockchain-enhanced high-confidence energy sharing in Internet of electric vehicles," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 7868–7882, Sep. 2020.

[16] T. Jiang, H. Fang, and H. Wang, "Blockchain-based Internet of vehicles: Distributed network architecture and performance analysis," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4640–4649, Jun. 2019.

[17] H. Kim, J. Park, M. Bennis, and S.-L. Ki. "Blockchained on-device federated learning," *IEEE Commun. Lett.*, vol. 24, no. 6, pp. 1279–1283, Jun. 2020.

[18] L. Lyu *et al.*, "Towards fair and decentralized privacy-preserving deep learning with blockchain," 2019. [Online]. Available: arXiv:1906.01167.

[19] A. Baldominos and Y. Saez, "Coin.AI: A proof-of-useful-work scheme for blockchain-based distributed deep learning," 2019. [Online]. Available: arXiv:1903.09800.

[20] A. Goel, A. Agarwal, M. Vatsa, R. Singh, and N. Ratha, "DeepRing: Protecting deep neural network with blockchain," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2019, pp. 2821–2828.

[21] 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.

[22] T.-N. Dinh and M.-T. Thai, "AI and blockchain: A disruptive integration," *Computer*, vol. 51, no. 9, pp. 48–53, Sep. 2018.

[23] C. Pop, T. Cioara, M. Antal, I. Anghel, I. Salomie, and M. Bertoncini, "Blockchain based decentralized management of demand response programs in smart energy grids," *Sensors*, vol. 18, no. 1, p. 162, 2018.

[24] S. Tanwar, K. Parekh, and R. Evans, "Blockchain-based electronic healthcare record system for healthcare 4.0 applications," *J. Inf. Security Appl.*, vol. 50, Feb. 2020, Art. no. 102407.

[25] K.-N. Griggs, O. Ossipova, C. P. Kohlios, A. N. Baccarini, E. A. Howson, and T. Hayajneh, "Healthcare blockchain system using smart contracts for secure automated remote patient monitoring," *J. Med. Syst.*, vol. 42, no. 7, p. 130, 2018.

[26] H. Min, "Blockchain technology for enhancing supply chain resilience," *Bus. Horizons*, vol. 62, no. 1, pp. 35–45, 2019.

[27] K. Francisco and D. Swanson, "The supply chain has no clothes: Technology adoption of blockchain for supply chain transparency," *Logistics*, vol. 2, no. 1, p. 2, 2018.

[28] S. Nakamoto, *Bitcoin: A Peer-to-Peer Electronic Cash System*, Bitcoin, 2008, pp. 1–9. [Online]. Available: https://git.dhimmel.com/bitcoin-whitepaper/

[29] I. Eyal, A.-E. Gencer, E.-G. Sirer, and R. van Renesse, "Bitcoin-NG: A scalable blockchain protocol," in *Proc. 13th USENIX Symp. Netw. Syst. Design Implementation*, 2016, pp. 45–59.

[30] E.-K. Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford, "Enhancing bitcoin security and performance with strong consistency via collective signing," in *Proc. 25th USENIX Security Symp.*, 2016, pp. 279–296.

[31] C. Li, P. Li, D. Zhou, W. Xu, F. Long, and A. Yao "Scaling Nakamoto consensus to thousands of transactions per second," 2018. [Online]. Available: arXiv:1805.03870.

[32] S. King and S. Nadal. (Aug. 19, 2012). *PPCoin: Peer-to-Peer Crypto-Currency With Proof-of-Stake*. [Online]. Available: https://www.chainwhy.com/upload/default/20180619/126a057fef926dc286accb372da46955.pdf

[33] V. Buterin, "A next-generation smart contract and decentralized application platform," White Paper, 2014. [Online]. Available: https://ethereum.org/en/whitepaper/

[34] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," 2014. [Online]. Available: arXiv:1011.1669v3.

[35] L. Ren, "Proof of stake velocity: Building the social currency of the digital age," White Paper, 2014. [Online]. Available: https://www.cryptoground.com/storage/files/1528454215-cannacoin.pdf

[36] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Proc. Annu. Int. Cryptol. Conf.*, 2017, pp. 357–388.

[37] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," *Proc. 26th Symp. Oper. Syst. Principles*, 2017, pp. 51–68.

[38] D. Larimer, "Delegated proof-of-stake (DPOS)," Bitshare White Paper, 2014. [Online]. Available: https://how.bitshares.works/en/master/technology/dpos.html#

[39] S. De Angelis, L. Aniello, R. Baldoni, F. Lombardi, A. Margheri, and V. Sassone, "PBFT vs proof-of-authority: Applying the cap theorem to permissioned blockchain," in *Proc. Italian Conf. Cyber Security*, 2018, pp. 1–11.

[40] K. Li, H. Li, H. Hou, K. Li, and Y. Chen, "Proof of vote: A high-performance consensus protocol based on vote mechanism & consortium blockchain," in *Proc. IEEE 3rd Int. Conf. Data Sci. Syst. (HPCC/SmartCity/DSS)*, 2017, pp. 466–473.

[41] J.-T. Kim, J. Jin, and K. Kim, "A study on an energy-effective and secure consensus algorithm for private blockchain systems (PoM: Proof of majority)," in *Proc. Int. Conf. Inf. Commun. Technol. Convergence*, 2018, pp. 932–935.

[42] 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.

[43] L. Lamport, C. Lee, A. Mizrahi, and M. Rosenfeld, "Paxos made simple," *ACM SIGACT News*, vol. 32, no. 4, pp. 51–58, 2001.

[44] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *Proc. USENIX Annu. Tech. Conf.*, 2014, pp. 305–320.

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

[46] Y. Wang *et al.*, "Study of blockchains's consensus mechanism based on credit," *IEEE Access*, vol. 7, pp. 10224–10231, 2019.

[47] M.-M. Jalalzai and C. Busch, "Window based BFT blockchain consensus," *Proc. IEEE Int. Conf. Internet Things (iThings) IEEE Green Comput. Commun. (GreenCom) IEEE Cyber, Phys. Soc. Comput. (CPSCom) IEEE Smart Data (SmartData)*, 2018, pp. 971–979.

[48] K. Lei, Q. Zhang, L. Xu, and Z. Qi, "Reputation-based Byzantine fault-tolerance for consortium blockchain," in *Proc. IEEE 24th Int. Conf. Parallel Distrib. Syst.*, 2018, pp. 604–611.

[49] B. França, M. Wissfeld, P. Berrang, and P. von Styp-Rekowsky, "Albatross: An optimistic consensus algorithm," 2019. [Online]. Available: arXiv:1903.01589.

[50] M. Du, Q. Chen, and X. Ma, "MBFT: A new consensus algorithm for consortium blockchain," *IEEE Access*, vol. 8, pp. 87665–87675, 2020.

[51] H. Lu, Q. Liu, D. Tian, Y. Li, H. Kim, and S. Serikawa, "The cognitive Internet of vehicles for autonomous driving," *IEEE Netw.*, vol. 33, no. 3, pp. 65–73, May/Jun. 2019.

[52] G. Sun, L. Song, H. Yu, Xi. Du, and M. Guizani, "A two-tier collection and processing scheme for fog-based mobile crowd sensing in the Internet of vehicles," *IEEE Internet Things J.*, early access, Aug. 12, 2020, doi: 10.1109/JIOT.2020.3015967.

[53] G. Sun, S. Sun, H. Yu, and M. Guizani "Towards incentivizing fog-based privacy-preserving mobile crowdsensing in the Internet of vehicles," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4128–4142, May 2020.

[54] J. Yang, W. Xiao, H. Lu, and A. Barnawi, "Wireless high-frequency NLOS monitoring system for heart disease combined with hospital and home," *Future Gener. Comput. Syst.*, vol. 110, pp. 772–780, Sep. 2020.

[55] I. Eyal and E.-G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," in *Proc. Int. Conf. Financ. Cryptogr. Data Security*, 2014, pp. 436–454.

[56] M. Swan, "Blockchain temporality: Smart contract time specifiability with blocktime," in *Proc. Int. Symp. Rules Rule Markup Lang. Semantic Web*, 2016, pp. 184–196.