SCIENCE CHINA
Information Sciences

• REVIEW •

February 2021, Vol. 64 121101:1–121101:15
https://doi.org/10.1007/s11432-019-2790-1

A survey of Blockchain consensus algorithms: mechanism, design and applications

Xiang FU[1,2*], Huaimin WANG[1,2*] & Peichang SHI[1,2]

[1] *College of Computer, National University of Defense Technology, Changsha 410073, China;*
[2] *Key Laboratory of Software Engineering for Complex Systems, National University of Defense Technology, Changsha 410073, China*

Received 3 October 2019/Revised 18 November 2019/Accepted 5 February 2020/Published online 18 November 2020

Abstract In 2008, Blockchain was introduced to the world as the underlying technology of the Bitcoin system. After more than a decade of development, various Blockchain systems have been proposed by both academia and industry. This paper focuses on the consensus algorithm, which is one of the core technologies of Blockchain. In this paper, we propose a unified consensus algorithm process model that is suitable for Blockchains based on both the chain and directed acyclic graph (DAG) structure. Subsequently, we analyze various mainstream Blockchain consensus algorithms and classify them according to their design in different phases of the process model. Additionally, we present an evaluation framework of Blockchain consensus algorithms and then discuss the security design principles that enable resistance from different attacks. Finally, we provide some suggestions for selecting consensus algorithms in different Blockchain application scenarios.

Keywords Blockchain, consensus algorithm, Byzantine fault-tolerant, process model, design principles

Citation Fu X, Wang H M, Shi P C, et al. A survey of Blockchain consensus algorithms: mechanism, design, and applications. Sci China Inf Sci, 2021, 64(2): 121101, https://doi.org/10.1007/s11432-019-2790-1

1 Background of Blockchain

In 2008, Blockchain was introduced to the world as the underlying technology of the Bitcoin system. In the founding paper on Bitcoin, "Bitcoin: a peer-to-peer electronic cash system" [1], Blockchain was described as a data structure using asymmetric encryption algorithms and hash functions to ensure that data tampering and forgery are impossible [2, 3]. The data-blocks in Blockchain represent transfer transactions between users in chronological order. In crypto-currencies such as Bitcoin and Ethereum [4], Blockchain is a decentralized distributed ledger enabling the free transfer of end-to-end digital assets without involving any central role or third party. Due to its basic characteristics such as decentralization, tamper-proof and traceability, Blockchain can also serve as a distributed network protocol enabling a trust relationship between different participants who do not know each other. After more than ten years of development, the Blockchain technology is not only used in the field of crypto-currency but also increasingly applied in other fields such as the Internet of Things, healthcare, and education [5–8]. Providing the variety of scenarios where the Blockchain technology can be applied, the current focus of research is on designing more efficient and secure Blockchain systems.

In the traditional chain-structured Blockchain, the adjacent blocks are connected through hash codes, which can not only serve as unique IDs but also prove the integrity of the blocks. An intuitive representation of the Blockchain with a chain structure is illustrated in Figure 1(a). At the same time, a new data structure for Blockchain has sprung up in recent years based on directed acyclic graphs (DAGs) [9]. As illustrated in Figure 1(b), blocks are no longer organized as a chain in the DAG-based Blockchain but as a directed acyclic graph, where the nodes represent data blocks, while the directed edges represent the validation relationship between parent-child data blocks. No matter which type of the data structure is

* Corresponding author (email: fuxiang13@nudt.edu.cn, whm_w@163.com)

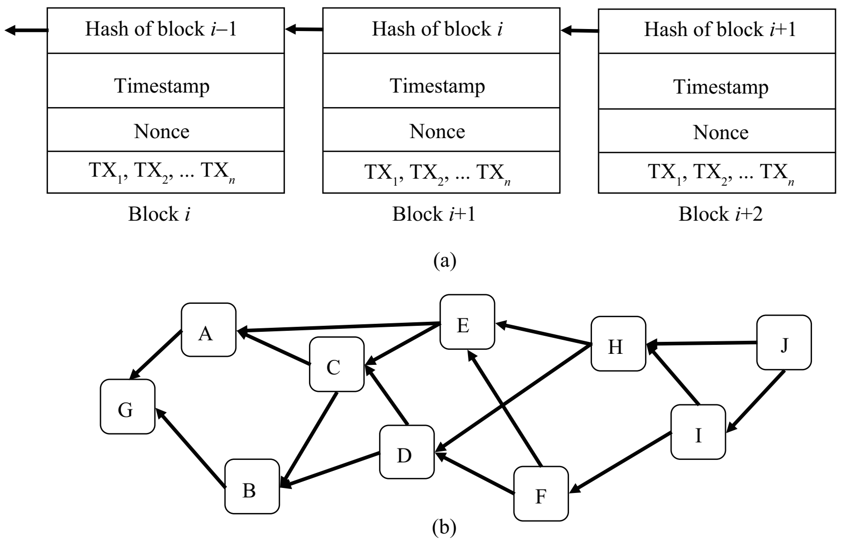

Figure 1 Chain (a) and DAG-based (b) data structures of Blockchain.

deployed in a Blockchain, the ultimate purpose of the structure is to ensure that all nodes reach consensus on the data without involving any central role or third parties. Generally, Blockchains can be divided into three categories according to the different access mechanisms of their nodes: public Blockchain, consortium Blockchain, and private Blockchain. In the public Blockchain, anyone can participate in the consensus process, and nodes can join or quit freely. The private Blockchain only allows managers to join the consensus process; hence, it can be regarded as a completely centralized data management system. The centralization degree of the consortium Blockchain is between those of the public and private Blockchains; in particular, a group of pre-selected users from several organizations would dominate the consensus process [10, 11]. From the perspective of consensus algorithms, we do not deliberately distinguish between the three types of Blockchains as the differences and connections among them can be naturally reflected in the consensus algorithm process model proposed in this paper.

2 Byzantine generals problem and Blockchain consensus algorithms

Consensus algorithms are derived from the famous Byzantine generals problem, which was first described by Lamport in his paper “The byzantine generals problem” in 1982 [12]. The Byzantine generals problem can be described as follows. Byzantine is the capital of the ancient eastern Roman Empire. There are several fiefs in Byzantine, each stationed by a general and his soldiers to resist foreign enemies. When facing enemies, each general can give two orders: attack or retreat. Only when all honest generals agree on an order to attack or retreat can they minimize casualties and win a war. However, Byzantine is so vast that these generals cannot discuss the order together because they have to guard in their fiefs separately. Therefore, the orders from generals are delivered via signalmen. The generals eventually make their final decisions (attack or retreat) by sending their orders to the other generals and collect the orders from the other generals. In this scenario, we assume that the signalmen are honest. However, some of these generals are traitors, who may send wrong orders or send different orders to different generals and ultimately undermine the overall decision of the honest generals. In summary, the Byzantine generals problem can be formulated as a problem of getting honest generals to reach a consensus in the presence of several traitors.

The consensus algorithm in Blockchain is used to solve the problem of ensuring data consistency in the presence of several failure nodes in a distributed system. Failure nodes can be divided into two types: Crash fault nodes and Byzantine fault nodes. Crash fault nodes fail only by halting; that is, they can only stop working and have no other malicious behaviors present [13]. In this case, messages can only be delayed or lost. In contrast, Byzantine fault nodes behave arbitrarily. They can send wrong messages to other nodes or send different messages to different nodes to undermine the process of reaching a consensus. When a distributed system has only Crash fault nodes, the consensus issue is relatively simple, with many Crash fault tolerate algorithms being proposed, including the Paxos [14, 15] and

Raft [16] algorithms. However, different nodes in a typical Blockchain system often represent different individuals or consortiums, who may behave arbitrarily when there is no central role. Therefore, the Blockchain consensus algorithm should be able to tolerate Byzantine fault nodes.

When designing a consensus algorithm, the FLP theorem published in the paper "Impossibility of distributed consensus with one faulty process" by Fischer et al. in 1985 [17] should be considered. The name of this theorem is composed of the initials of the three authors' surnames. The FLP theorem states that no overall consensus can be reached in a distributed system following the asynchronous communication model as long as one process fails (unresponsive or suspended). The synchronous communication model uses a consistent clock frequency, allowing errors within a limited time, whereas there is no unified clock and nodes cannot use the timeout mechanism in the asynchronous communication model. In the latter model, node failures cannot be detected, and messages can be delayed without a limit, so the message receiver cannot know when the message will arrive. The majority of the current consensus algorithms are based on either the complete synchronous communication model or the weak synchronous communication model, where a timeout mechanism is set for the transmission of messages. Here, we assume that Blockchain consensus algorithms employ the weak synchronization communication model; that is, a message may be delayed but will eventually reach the receiver within a specific time limit, beyond which, the sending node will be considered as failed. Under this assumption, Blockchain consensus algorithms must guarantee the consistency and liveness properties.

Consistency. If a transaction is valid on an honest node, it is also valid on other honest nodes. Consistency ensures that the double spending attack [18] will never succeed in the Blockchain system if honest nodes are in the majority.

Liveness. All valid transactions sent by honest nodes must be eventually confirmed, which ensures the availability of the system.

Consistency and liveness are the fundamental issues to be considered when designing a Blockchain consensus algorithm. A Blockchain consensus algorithm is correct only when it meets these two requirements. On the premise of guaranteeing consistency and liveness, other aspects of Blockchain can be designed according to application requirements, such as improving throughput, increasing scalability, and reducing resource costs. Consensus algorithms have been studied for a long time in the field of traditional distributed systems. Some Blockchain consensus algorithms such as proof of work (PoW) [19], proof of stake (PoS) [20], and practical Byzantine fault tolerance (PBFT) [21] were deployed in the early Blockchain systems. A large number of new Blockchain consensus algorithms have emerged more recently. According to some surveys such as [5, 22–24], these new consensus algorithms can be broken down into three categories: (1) variants of the original consensus algorithms, e.g., Bitcoin-NG [25] and Algorand [26], which are the improvements of PoW and PBFT, respectively; (2) combinations of the original consensus algorithms, e.g., delegated BFT (DBFT) [27], which is the combination of PoS and PBFT; (3) DAG-based consensus algorithms, e.g., Byteball [28] and Hashgraph [29]. At the same time, many existing surveys of Blockchain consensus algorithms separate the chain and DAG structures or analyze the consensus algorithms only for the chain structure. According to our research and analysis, consensus algorithms for chain and DAG structures have a particular inner link. Therefore, we propose a unified Blockchain consensus algorithm process model in the next section.

3 Blockchain consensus algorithm process model

With the development of the Blockchain technology, a variety of Blockchain consensus algorithms have been proposed in both academia and industry. Nowadays, the data structures of Blockchains are no longer the traditional "block + chain". To meet the demands for high throughput, low latency, and low communication cost in some scenarios, Blockchains based on the DAG data structure have been proposed [9]. After analyzing various Blockchain consensus algorithms based on both the chain and DAG structures, we found that although they are different, they had internal connections. Based on this, we propose a unified Blockchain consensus algorithm process model. Different Blockchain consensus algorithms can be analyzed using this model by determining their process phases and understanding their core ideas. As illustrated in Figure 2, the proposed Blockchain consensus algorithm process model has three phases corresponding to the execution processes, from the generation to the final confirmation of a data unit (a transaction or a block). These phases are accountant selection, block addition, and transaction confirmation.

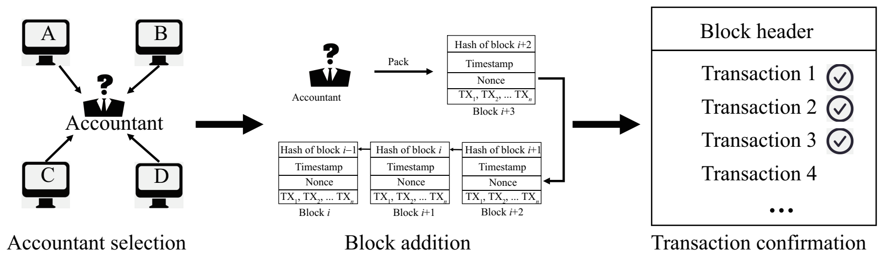

Figure 2 Blockchain consensus algorithm process model.

3.1 Accountant selection

Accountant selection is the first phase in the model. Accountants are responsible for generating blocks, which includes the collection and validation of transactions, packing transactions into a block, and sending the block to other nodes (in some Blockchains, a block may contain only a single transaction). In this phase, the input includes transactions and all nodes, while the output includes an accountant identifier and a new block. According to our analysis of existing Blockchains, the following three main scenarios can happen in this phase:

(1) Randomly choosing an accountant for every accounting round using PoW, PoS, or verifiable random functions (VRF) [30];

(2) Choosing an accountant for every accounting round in a predetermined order;

(3) Every node is an accountant, and a node is only responsible for the transactions it generates. This situation always occurs in DAG-based Blockchains, where there are no accountant rounds.

3.2 Block addition

Each node maintains a copy of the Blockchain (ledger) locally. After a node receives a block sent by an accountant, it first verifies the accountant and block. The validation of the accountant depends on the way of accountant selection, while the validation of the block includes the validation of the block header and transactions. The block header contains the hash value of other blocks; hence, the validation of the block header includes the validation of the data structure. If both the accountant and block are valid, then the block can be added to the Blockchain. It is vital to note that some consensus algorithms such as PBFT require votes from the majority of the nodes before adding a block to the Blockchain. These consensus algorithms involve high communication costs and require the nodes to be online. In this phase, the inputs are the current Blockchain and new blocks, while the output is a new Blockchain.

3.3 Transaction confirmation

After the first two phases, each node has a local copy of the Blockchain from its perspective. The transaction confirmation phase is responsible for confirming transactions according to the Blockchain each node holds. The following two scenarios are possible in this phase.

(1) The block addition phase requires real-time votes from the majority of nodes. In this situation, every block has obtained real-time votes from the majority of nodes before it is added to the Blockchain. Therefore, every block in the Blockchain is confirmed.

(2) Blocks are added to the Blockchain after the validation without real-time votes from the majority of nodes. In this situation, different nodes may have different Blockchain data due to the network delay or other reasons. Therefore, a block being on the Blockchain does not mean it has been confirmed. The confirmation of a transaction depends on the data structure of the Blockchain. For example, in the Bitcoin system, a block can be confirmed if there are at least six blocks after it.

In this phase, the input is the current Blockchain, while the output includes confirmed blocks or transactions.

According to different application requirements, the designs of the three phases can be different, each with its advantages and disadvantages. The accountant selection phase in PoW ensures high scalability, while it has a low throughput and the risk of centralization. The block addition phase in PBFT requires real-time voting, which brings low transaction confirmation latency. However, it involves a high communication cost and has the risk of denial of service (DoS) attacks [31]. The accountant selection phase in Hashgraph makes everyone an accountant; hence, while the centralization risk is avoided, the double

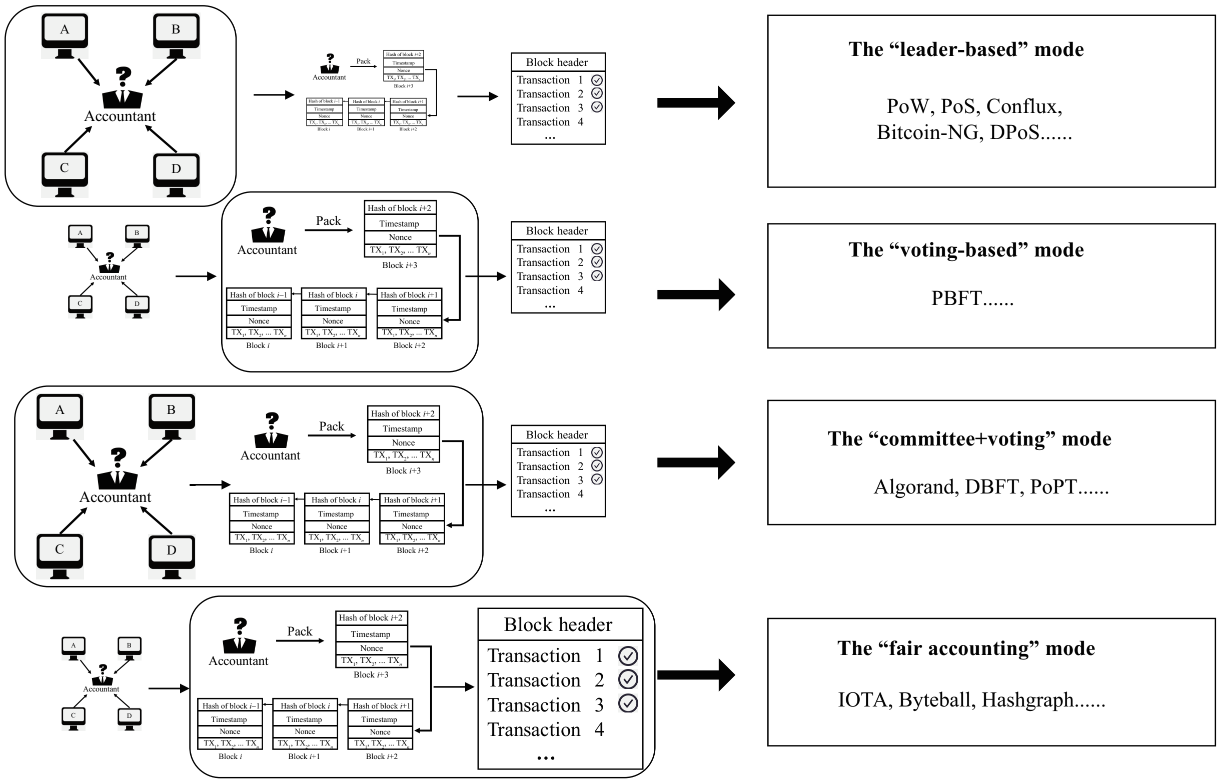

Figure 3 Classification of Blockchain consensus algorithms.

spending attack can be easily launched. Moreover, the liveness of Hashgraph cannot be guaranteed in a public environment [32].

4 Classification of Blockchain consensus algorithms

In this section, we analyze the consensus algorithms that are currently in the mainstream of industry and academia from the perspective of the Blockchain consensus algorithm process model proposed in the previous section. According to the different designs of each phase in the process model, these algorithms can be broken down into four modes as shown in Figure 3. (1) The "leader-based" mode, in which designers focus mainly on the "accountant selection" phase; (2) the "voting-based" mode in which the "block addition" phase is the most important; (3) the "committee + voting" mode in which the "accountant selection" and "block addition" phases are mainly designed; (4) the "fair accounting" mode in which the "block addition" and "transaction confirmation" phases are mainly designed.

4.1 "Leader-based" mode

The consensus algorithms of the leader-based mode mainly focus on the design of the accountant selection phase. The algorithms in this group include PoW, PoS, and their variants. The computing power of nodes determines the attribution of the accounting rights of PoW. In the Bitcoin system, which is the first notable application of PoW-based Blockchain, each participating node solves a complex but easily verified mathematical problem based on their respective computing powers in the accountant selection phase. This mathematical problem can be expressed as finding a suitable random number (Nonce) for the block header according to the current difficulty value so that the SHA256 [33] value of the block header is less than or equal to the target value. This process is also known as "mining" [34]. The node solving the problem is first selected as the accountant and may receive an accounting reward. The design of the block addition phase is relatively simple: after verifying that a block was created by the accountant, the block is immediately added to the current Blockchain, if it passes the verification. The transaction confirmation phase follows the longest chain principle; that is, when a fork occurs, the longer chain is considered to be the legal one. If a block has six or more successor blocks, the transactions it contains can be regarded as confirmed. PoW relies on the computing power to get accounting rights. Due to the randomness and openness of this competition, PoW has good scalability and decentralization. However, before the consensus is reached for each block, all nodes participate in the competition for the accounting right, which leads to a waste of the computing power.

The original PoW algorithm as utilized in Bitcoin has inspired many further algorithms such as Bitcoin-NG and conflux [35]. The purpose of Bitcoin-NG is to solve the problem of low throughput and high latency in Bitcoin. Bitcoin-NG reuses the Bitcoin PoW algorithm, so mining is still employed in the accountant selection phase with the exception of the following modification. Transactions are considered in the calculation of the hash value of a block in Bitcoin, whereas the mining process in Bitcoin-NG does not include packing transactions. The accountant selected by PoW is responsible for the block generation for the next period. As opposed to PoW that uses the traditional chain structure for Blockchain data, Conflux uses the DAG data structure. Mining is also used in the accountant selection phase of Conflux. In the block addition phase, a block in Conflux chooses two previous blocks to link to so that a DAG can be formed. Conflux supports forks, turning invalid bifurcation blocks in Bitcoin into valid blocks. In this way, Conflux increases throughput and reduces the computing power waste.

To solve the enormous computing power consumption problem of PoW, researchers proposed the PoS consensus algorithm. In PoS, nodes compete for the accounting rights through the tokens each node holds. Therefore, the core idea of PoS is that the node with a higher stake rather than more computing power has a higher probability of obtaining the accounting right. When we talk about the PoS algorithm, we do not refer to a specific algorithm, but the consensus algorithms that rely on the stake to compete for accounting rights. There are several implementations of PoS. In the simplest one, the node with the highest stake directly obtains the accounting rights in the accountant selection phase, or, based on PoW, the mining difficulty changes according to the proportion and time of the token each node holds, thus speeding up the search for Nonce. PoS primarily uses the proof of stake to replace the proof of hash-based procedure in PoW. Thus, the accounting rights are assigned to the nodes with the highest stake rather than the highest computing power. Although this method reduces the waste of computing power, it may have the risk of monopoly, which leads to the centralization trend of the system while allowing malicious attackers to have a clear target to attack, undermining the security of the system. To reduce the monopoly risk of PoS, researchers proposed the DPoS consensus algorithm [36]. In DPoS, the owner of the stake grants the rights to other nodes to vote for accountant candidates, with the top-ranked candidates taking turns in having the accounting rights. The next two phases are the same as in PoS. The PoS and DPoS algorithms are almost identical to PoW in their block addition and transaction confirmation phase. There are many other leader-based mode consensus algorithms such as proof of luck [37] and proof of burn [38]. Proof of luck uses the trusted execution environment (TEE) platform [39] to generate a trusted random number for choosing a leader, while in proof of burn, miners destroy (burn) coins by sending them to an address that does not spend. The miner who burns the largest amount of coins during a period is selected to be the leader to generate a new block.

4.2 Voting-based mode

In traditional distributed systems, the majority of consensus algorithms are based on voting, which is the most intuitive way of reaching a consensus, that is, getting approvals from the majority directly by voting. All traditional voting algorithms such as Paxos and Raft consider only the node's fail-stop error and cannot solve the Byzantine fault-tolerance problem, where nodes may have malicious behaviors. The most widely used consensus algorithm based on voting in Blockchain is the PBFT consensus algorithm, which serves as a basis for many other consensus algorithms designed in the "committee + voting" mode. PBFT was proposed by Castro and Liskov in 1999 [21]. Its accountant selection phase is straightforward, usually in the form of polling. PBFT requires that the maximum number of malicious nodes f does not exceed 1/3 of the total number of nodes n, that is, $n \geqslant 3f + 1$.

In the block addition phase, a block must pass the following three-stage commit process before it can be added to the Blockchain. In the first pre-prepare stage, the accountant sends a block to other nodes for them to verify it (we call the accountant as the primary and the other nodes as the replicas). In the second prepare stage, each replica sends the verification result to all other nodes, and all nodes re-confirm the block according to the message sent by each replica (if there are more than $2f + 1$ "YES" votes, the block is valid). In the third commit stage, each node sends the verification result of the prepare stage to all other nodes again, and each node makes a final confirmation of the block according to the received messages. If there are more than $2f+1$ "YES" votes, the block can be added to the Blockchain. In PBFT, the block addition phase contains the block confirmation phase providing that each block is confirmed by the majority of nodes in real-time. Therefore, transactions can be confirmed once the block is added to the Blockchain. It is apparent that in the three-stage commit process, every two nodes need to communicate with each other. Thus, the time complexity of PBFT is $O(n^2)$. Because of this high communication cost and the requirement of weak network synchronization, PBFT does not have good scalability. Simple BFT (SBFT) [40], which is the simplified version of PBFT, has a four-stage message pattern instead of the three-stage pattern employed in PBFT. In the fourth stage, the checkpoint messages are signed, and only $f+1$ signed checkpoint messages constitute the proof of an individual block (in a typical three-stage case, $2f+1$ signatures are required for the block proof). Hyperledger fabric [41] uses PBFT and SBFT as the consensus algorithms in its version 0.6 and 1.0, respectively. To improve scalability, VMware and IBM proposed Hotstuff [42] and MirBFT [43], respectively. Hotstuff is a leader-based Byzantine fault-tolerant replication protocol for the partially synchronous model. Messages are sent only between the leader and replicas, and the leader collects votes from replicas. The time complexity of Hotstuff is reduced to $O(n)$. Hotstuff serves as a basis for LibraBFT [44], which is the consensus protocol of Libra Blockchain [45] proposed by Facebook. MirBFT allows a set of leaders to propose request batches independently and in parallel while preventing request duplication performance attacks through a rotating assignment of a partitioned request hash space to leaders. The core consensus process of HoneyBadgerBFT [46] is the binary agreement (BA), which is a standard primitive that allows nodes to agree on the value of a single bit. Based on BA, the protocol from Ben-Or et al. [47] is used to agree on a set of values containing the entire proposals of at least $N-f$ nodes.

4.3 Committee + voting mode

The consensus algorithms of the committee + voting mode primarily focus on the accountant selection and block addition phases of the process model. The former phase is usually used to select the accountant and committee members (only the committee members can become a participant in the consensus process). The latter phase is designed for voting rules. This section introduces several popular consensus algorithms of the committee + voting mode, including DBFT, proof of previous transactions (PoPT) [48], and algorand. In DBFT, there are two types of nodes: the accounting node and ordinary nodes. The ordinary nodes vote to choose the accounting nodes based on the proportion of the stake they hold. The committee consists only of accounting nodes. In the accountant selection phase of DBFT, just like in PBFT, the accountant who generates the block is selected from the accounting nodes through polling. In the block addition phase, the committee runs the PBFT consensus. The PoPT consensus algorithm ranks each user in the accountant selection phase by the number of times the user becomes the accountant and fees the user pays in previous transactions. The top n users become members of the committee. Then, m ($m<n$) nodes are selected from the committee using a specific hash function, and they generate blocks in parallel. Thus, there are m blocks in one round. In the block addition phase, the committee runs PBFT consensus in parallel for every block generated.

Algorand uses the VRF to randomly determine the accountant and committee members for each block in the accountant selection phase. Then, the committee runs the BA* Byzantine agreement to reach a consensus for the block in the block addition phase. Similar to PBFT, transactions can be confirmed once the block is added to the Blockchain in the above three algorithms. The core idea of consensus algorithms of the committee + voting mode is to narrow the range of nodes participating in the consensus process, thereby reducing the communication complexity. Moreover, there is no sacrifice of the system's decentralization since all nodes have chances to become the committee members.

4.4 Fair accounting mode

The majority of the Blockchains with the consensus algorithms of the "fair accounting" mode use DAG-based data structures. In the accountant selection phase, each node is an accountant, which means that a node is only responsible for packing the transactions it generates. Therefore, every node has a fair chance to be an accountant. In the block addition phase, there are always some block-linking rules for blocks to choose their parent-blocks. A block always has more than one parent-block, and that is why the data structure is based on a DAG instead of a chain. Therefore, the block-linking rules determine the DAG-based data structure of the Blockchain. Finally, in the transaction confirmation phase, more complicated rules are used to determine whether a block has reached a consensus based on the location of

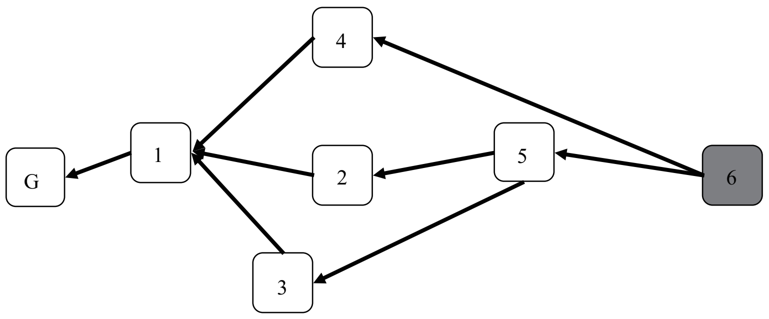

Figure 4 Tangle of IOTA.

the block in the DAG. In this subsection, we introduce three popular Blockchains of the fair accounting mode, including IOTA [49], Bytaball, and Hashgraph.

As shown in Figure 4, the DAG-based data structure of IOTA is called Tangle. In Tangle, each transaction is represented by a vertex. We call a transaction that has no children a tip; in other words, a tip is not verified by any other transaction. In Figure 4, transaction T6 represents a tip. In the block addition phase, when a new vertex (transaction) is added to the Tangle, it needs to select two existing tips as parents and adds two edges pointed to the two parents from this new vertex. In the transaction confirmation phase, there are two types of confirmation; full and partial confirmation. A transaction that is directly or indirectly verified by all tips is ultimately fully confirmed. However, it is impossible for a node to have all tips due to the network delay, which means it is usually challenging to achieve a complete confirmation. Therefore, partial confirmation, which requires a certain proportion (for example, 80%) of the tips to be verified, is used instead of full confirmation. Byteball's block addition phase is similar to that of IOTA, with the only difference that all historical transactions from a generator should be verified by its new transaction. In the transaction confirmation phase, Byteball has an entirely different design compared to that of IOTA. Byteball introduces witnesses to achieve a consensus. A witness is a particular type of user who is elected by the community. Transactions directly or indirectly confirmed by more than half of the witnesses are considered to be confirmed transactions. Therefore, the maintenance of Byteball is in the hands of these witnesses, which leads to the weak decentralization of Byteball.

In Hashgraph, transactions are packed in events generated by nodes. When a node generates an event, it must set two parent-events: self-parent, which is the event generated by a node last time, and other-parent, which is generated by one of the node's neighbors. Hence, when receiving events in the block addition phase, a node must verify whether the events' parents are set according to the rule. Compared to IOTA and Byteball, the rules in the transaction confirmation phase of Hashgraph are much more complicated. The essence is that before an event can be confirmed, it must be verified multiple times by more than two-thirds of all nodes to ensure that the fork attack cannot succeed.

4.5 Comparison of different consensus algorithms

To sum up, the scalability of the leader-based mode is high, while there is a fairness problem, and usually, the efficiency is low. In the voting-based mode, transactions can be confirmed once they are added to the Blockchain, and the confirmation delay is low. However, there is a network synchronization is needed, communication overheads are considerable, and scalability is low. The committee + voting mode combines the advantages of the former two. However, when the number of committee members is large, there is a problem of large communication overhead. The fair accounting mode (mostly based on DAG data structures) has low resource consumption, its scalability and efficiency are high. However, there is a centralization risk, and the storage cost is always higher than that of the chain-structure Blockchains. It is mostly used in consortium Blockchains. A brief description of the three-phase process model of some mainstream consensus algorithms is shown in Table 1 [50–53]. The classification and comparison of some mainstream consensus algorithms are shown in Table 2. The data on throughput measured in transactions per second (TPS) were taken from other scientific papers, white papers, or technical reports provided by the corresponding teams or obtained from the existing operating systems.

Table 1 A brief description of the three phases of the mainstream consensus algorithms

Algorithms	Phases		
	Accountant selection	Block addition	Transaction confirmation
Pow	Mining (finding the Nonce that makes the hash value of the block header less than or equal to the target value)	Adding blocks to Blockchain directly after verify	Longest chain principle
Bitcoin-NG	Mining	Adding blocks to Blockchain directly after verification	Longest chain principle
Conflux	Mining	Adding blocks to Blockchain directly after verification	Choose main chain according to the data structure
PoS	Nodes with more stake have a higher probability of being accountants	Adding blocks to Blockchain directly after verification	Longest chain principle
PoA [50]	Combine PoW & PoS	Adding blocks to Blockchain directly after verification	Longest chain principle
PoB	Burning the most coins to be the accountant	Adding blocks to Blockchain directly after verification	Longest chain principle
Proof-of-Luck	Use TEEs to generate random numbers to choose the accountant	Adding blocks to Blockchain directly after verification	Longest chain principle
DPoS	Nodes vote for accountants by their stake	Adding blocks to Blockchain directly after verification	Longest chain principle
PBFT	Polling among all the nodes	Adding blocks to Blockchain after verification and three-phase commit	Blocks confirmed once the they are added to the Blockchain
Algorand	VRF	Adding blocks to Blockchain after verification and run BA* algorithm among the accountant and committee	Blocks confirmed once the they are added to the Blockchain
DBFT	Nodes vote for accountants by their stake	Same as PBFT	Blocks confirmed once the they are added to the Blockchain
Tendermint [51]	Polling among all the committee members	Simpler PBFT	Blocks confirmed once the they are added to the Blockchain
Casper FFG [52]	PoW for the accountant and PoS for the committee	Adding after verification for ordinary blocks, simpler PBFT for checkpoint blocks	Blocks before the checkpoint confirmed once the checkpoint block is added to the Blockchain
PoPT	Rank all the nodes according to previous transactions and use a hash function to select the accountants from the top n nodes	Run PBFT in parallel	Blocks confirmed once the they are added to the Blockchain
IOTA	Every node is the accountant for their own transactions	Adding blocks to Blockchain directly after verification	Confirmed by tips
Byteball	Every node is the accountant for their own transactions	Adding blocks to Blockchain directly after verification	Confirmed by witnesses
Hashgraph	Every node is the accountant for their own transactions	Adding blocks to Blockchain directly after verification	Confirmed after verified multiple times by more than 2/3 of all the nodes

5 Blockchain consensus algorithm design principles for security

Based on our analysis of the existing Blockchain consensus algorithms and surveys such as [22, 54, 55], the Blockchain consensus algorithms can be evaluated in the following three dimensions: effectiveness, security, and decentralization. Each of these dimensions has several evaluation indicators. As shown in Table 3, the effectiveness dimension covers all aspects of the Blockchain performance, including throughput (the number of transactions that the system can process in a unit of time), confirmation latency (the time required for a transaction to move from the generated stage to final confirmation stage), scalability (the number of nodes that can participate in the consensus process), and various resource costs such as CPU, network, memory, computing power, and storage. The decentralization dimension refers to the proportion of nodes participating in the consensus process, fairness of the accounting rights distribution,

Table 2 Classification and comparison of mainstream consensus algorithms

Mode	Advantages	Disadvantages	Consensus algorithms	TPS	Energy saving	Adversary model (%)
Leader-based	High scalability, Public Blockchain	Decentralization risk, Low efficiency, High resource cost	PoW (Bitcoin)$^{1)}$	7	No	50
			PoW (Ethereum)$^{2)}$	15	No	50
			PoW (Bitcoin cash)$^{2)}$	60	No	50
			Bitcoin-NG$^{1)}$	7	No	50
			Conflux$^{3)}$	6400	No	50
			PoS (Cardano)$^{2)}$	8	Partial	50
			PoS (Peercoin)$^{4)}$	7	Partial	50
			DPoS (EOS)$^{5)}$	359	Partial	50
Voting-based	Low transaction, Confirmation latency	Low scalability, High communication cost, Weak network synchronization	PBFT (Fabric 0.6)$^{6)}$	1000+	Yes	33
			MirBFT$^{7)}$	23k	Yes	33
			HoneyBadgerBFT$^{8)}$	10k+	Yes	33
Committee + voting	High scalability, Low transaction confirmation latency, Public Blockchain	Complicated design	Algorand$^{1)}$	90	Yes	33
			DBFT (NEO)$^{9)}$	1000+	Yes	33
			PoPT$^{10)}$	–	Yes	33
			Tendermint$^{10)}$	–	Yes	33
Fair accounting	High efficiency, High scalability	Decentralization risk, High storage cost	IOTA$^{11)}$	4	Partial	50
			Byteball$^{12)}$	4	Yes	50
			Hashgraph$^{10)}$	–	Yes	33

1) TPS data extracted from [22].
2) TPS data extracted from [23].
3) TPS data extracted from [35].
4) TPS data extracted from https://docs.peercoin.net/#/block-size-limit-and-block-time-spacing. Visited on May 21, 2019.
5) TPS data extracted from https://eosflare.io/. Visited on May 21, 2019.
6) TPS data extracted from [53].
7) TPS data extracted from [43].
8) TPS data extracted from [46].
9) TPS data extracted from https://docs.neo.org/docs/zh-cn/basic/whitepaper.html. Visited on May 21, 2019.
10) No running systems and different block sizes and intervals can result in different TPS.
11) TPS data extracted from https://thetangle.org. Visited on May 21, 2019.
12) TPS data extracted from https://explorer.obyte.org. Visited on May 21, 2019.

and node access mechanism. The security dimension refers to whether the Blockchain system guarantees the consistency and liveness, and resists various attacks, including double spending, DoS attacks, Sybil attacks [56], eclipse attacks [57], and selfish mining [58]. The design of consensus algorithms often requires achieving a trade-off among the different aspects of the three dimensions. At the same time, the security dimension cannot be sacrificed under normal circumstances. Therefore, the degree of decentralization may be sacrificed for a more optimized performance, whereas the effectiveness may suffer if guaranteeing a high decentralization is more important.

Designing a consensus algorithm means designing the three phases of the process model proposed in Section 3. The three phases are closely related. If the accountant selection phase is designed to be fair and the result is unpredictable, then the block addition phase can be simplified, and blocks can be added to the Blockchain directly after the verification without voting. If the block addition phase is designed with real-time voting, then blocks are confirmed once they are added to the Blockchain, which means the transaction confirmation phase can be ignored. Below, we discuss the principles of how each phase should be designed so that the consensus algorithm can resist various attacks, including double spending, DoS attacks, Sybil attacks, eclipse attacks, and selfish mining.

Double spending refers to the attacker's second purchase of a token in a transaction by manufacturing a fork when the transaction is not fully confirmed, which causes the transaction to be revoked. Double spending mainly destroys the consistency of the Blockchain system. It is ineffective for the consensus algorithms, in which a real-time voting process is designed in the block addition phase, as this phase guarantees that a fork cannot be added to the Blockchain. Therefore, double spending mainly targets the Blockchains, in which transactions are not immediately confirmed when they are added to the Blockchain. To prevent this type of attack, the transaction confirmation phase should be strictly designed to ensure that the possibility of revoking a transaction after it has been confirmed does not exist, or the probability is low only to the extent of a theoretical possibility. For example, six confirmation rules are designed in the transaction confirmation phase of Bitcoin; if more than six blocks are connected after a block, then the transactions in the block can be considered as the final confirmation. Byteball requires transactions

Table 3 Three evaluation dimensions of BLOCKCHAIN consensus algorithms

Evaluation dimensions	Evaluation indicators
Effectiveness	Throughput Confirmation latency Scalability CPU cost Network cost Computing power cost Memory cost Storage cost . . .
Decentralization	Proportion of nodes participating in consensus process Fairness of accounting rights distribution Node access mechanism . . .
Security	Consistency Liveness Double spending resistance DoS attacks resistance Sybil attacks resistance Eclipse attacks resistance Selfish mining resistance . . .

to be verified by more than half of the witnesses before they are finally confirmed.

DoS attacks. A DoS attack is a cyber-attack in which the attackers seek to make a machine or network resource unavailable to its intended users by temporarily or indefinitely disrupting services of a host connected to the Internet. DoS attacks make the system unable to guarantee liveness. In Blockchain systems, DoS attacks are typically addressed by reducing or avoiding the effects of single-point failures on the system. Among all nodes, only the accounting node's failure has a significant impact on the system. Therefore, the following two principles should be followed when designing the consensus algorithm: (1) make the attackers unable to predict which node is the accounting node and (2) design a highly efficient mechanism to deal with the failure of the accounting node. To address the first principle, the accountant selection phase should be designed and the result should be made unpredictable. PoW, PoS, or the VRF can be used to select the accountant. For the second principle, the selection of the accountant is usually known in advance to the whole network; hence, it is impossible to prevent the accountant from being targeted by attackers. In this case, we can only design a mechanism to minimize the impact. Therefore, it is necessary to design a voting process in the block addition phase to detect whether the accountant is invalid or not. For example, in PBFT, once the accountant has been detected to send a block over a period of time, he can be considered as invalid and then the accountant can be replaced by a view change.

Sybil attacks. In the Blockchain system, the Sybil attack means that the same user has multiple identities. If a voting mechanism is adopted, a malicious node may have multiple votes, so that the attacker may take control over the system. Sybil attacks mainly destroy the fairness of the Blockchain system and bring the risk of centralization. Therefore, if the way of selecting the accountant does not depend on the number of identities a user has, Sybil attacks can be prevented in the accountant selection phase. For example, in PoW, the accounting right is linked to the computing power; in PoS, the accounting right is linked to the stake of a user; in PoPT, the choice of the committee members is linked to the fees paid by the user and the number of times the user has been an accountant in the past. Another way to prevent Sybil attacks is to set an identity authentication mechanism for consortium Blockchains.

Eclipse attacks. To launch an Eclipse attack, the attacker always encroaches on the victim's routing table and then sets the neighbors of the victim as the identities of the attacker, thus separating the victim from the real Blockchain network and controlling the victim's external contacts. The Eclipse attack is based on the Sybil attack, which means the attacker must first set up enough Sybil nodes and declare them to be normal, and then use these Sybil nodes to communicate with the victim. The Eclipse attack cannot take place in a consortium Blockchain having an identity authentication mechanism; however, it is impossible to prevent an attacker from simultaneously controlling multiple nodes in a public Blockchain without an identity authentication mechanism. While it is possible to design a reasonable mechanism to prevent unfair accounting caused by Sybil attacks, it is impossible to prevent the network isolation

Table 4 Common Blockchain application scenarios and consensus algorithm recommendations

Application scenario	Characteristics	Blockchian type	Node size	Recommended consensus algorithms
Internal cooperation of a few companies	Node size fixed Low transaction frequency No light nodes	Consortium	<20	PBFT Hashgraph . . .
A few enterprises cooperate to provide external services	Node size fixed Medium transaction frequency Some light nodes	Consortium	<20	PBFT Byteball . . .
Dozens of enterprises cooperate to provide external services	Node size fixed High transaction frequency Many light nodes	Consortium	<100	Hashgraph PoPT Algorand Committee + voting
IoT application: smart city	Node size not fixed High transaction frequency Many light nodes	Public/consortium	<1000	PoS PoPT Algorand Committee + voting
Crypto-currency	Node size not fixed Nodes access or exit without limit Many light nodes	Public	1000+	PoW PoS Leader-based

caused by Eclipse attacks at the consensus algorithm level. Eclipse attacks should be dealt with at the network communication level (for example, periodically updating the routing tables).

Selfish mining mainly aims at the PoW-based consensus algorithms. A selfish miner can continue mining next blocks after successfully mining one of them, maintaining in this way the leadership without publishing and distributing the mined blocks to the rest of the network. When the rest of the network is about to catch up with the selfish miner, the latter can release the solved blocks into the network and claim block rewards for them. Using timestamps can be a solution for this attack: if a miner releases a long list of blocks with recorded timestamps in one shot, then the rest of the network can consider not accepting these blocks.

6 Suggestions for selecting consensus algorithms

Different application scenarios require different types of Blockchains. For example, public Blockchains are more suitable for open environments, whereas consortium Blockchains are more appropriate for enterprise cooperations. After analyzing the most common Blockchain application scenarios, we divided them into five categories with different node sizes, transaction frequencies, and Blockchain types. Table 4 provides suggestions for selecting consensus algorithms for each one of these application categories.

(1) When the application scenario is the cooperation between a small number of companies, a Blockchain can be used to record business transactions among them. The total number of nodes is usually less than 20, thus high scalability of the Blockchain is not required. Moreover, there are no light nodes and the number of transactions is not huge. At the same time, the synchronization requirements of the system can be easily achieved due to the small number of full nodes. Therefore, PBFT-based or Hashgraph-based consensus algorithms could be good choices in this case.

(2) When multiple companies cooperate in providing services to their customers, a Blockchain is usually used to record the business transactions between the customers and the companies. In this scenario, there is a big number of light nodes controlled by customers, while the total number of full nodes is still less than 20. Therefore, the scalability requirement of the Blockchain is low, and the synchronization of the system is easy to achieve. However, there may be a high number of transactions resulting in the demand for a high transaction throughput and a low confirmation delay. PBFT-based and Byteball consensus algorithms are recommended in this scenario. When using Byteball, companies always play the role of witnesses.

(3) When dozens of enterprises cooperate and provide services to their customers, the total number of full Blockchain nodes controlled by these enterprises is usually more than 20 and less than 100, in addition to a big number of light nodes representing customers. For example, in Jointcloud computing [59], there are many cloud service providers and customers. In this case, it is not very easy to ensure the synchronization of so many full nodes. Hence, the PBFT algorithm is not suitable. However, a Hashgraph-

based consensus algorithm would be a suitable choice in this scenario because the communication model of Hashgraph is based on the gossip protocol, which does not require the system synchronization. Consensus algorithms based on the committee + voting mode such as PoPT and Algorand can also be used. The number of committee members should not exceed 20; otherwise, the throughput may decrease due to a high network overhead required in the voting phase.

(4) When a Blockchain is used for information sharing and device collaboration among IoT devices or for the scenario similar to those described in [60], the number of full nodes is not fixed. Consensus algorithms based on the committee + vote mode can still be useful. Furthermore, PoS-based consensus algorithms can also be used if there is no need for high throughput.

(5) For the most widely deployed crypto-currency applications such as Bitcoin, Peercoin, and the digital currency proposed in [61], the total number of full nodes is enormous, with nodes changing dynamically. The public Blockchains deployed in these applications are decentralized. To ensure the security in public Blockchains, their effectiveness would be inevitably sacrificed. In this scenario, consensus algorithms based on PoW or PoS are better choices. Furthermore, it is not recommended to use algorithms based on the committee + voting mode because the identity of each node is hidden in public Blockchains. Moreover, managing caseswhen the committee members are dishonest is challenging.

7 Conclusion and future work

The consensus algorithm is one of the essential Blockchain technologies and determines the performance of all aspects of a Blockchain system. In this paper, we proposed a consensus algorithm process model allowing to describe both the chain- and DAG-based Blockchain consensus algorithms. Furthermore, we reclassified the existing mainstream Blockchain consensus algorithms based on the proposed process model and compared them. We also presented the evaluation framework of Blockchain consensus algorithms and discussed the security design principles for resisting the most common attacks, including double spending, DoS attacks, Sybil attacks, and eclipse attacks. Finally, we analyzed the characteristics of Blockchain systems in different scenarios from the perspective of Blockchain applications and provided suggestions for selecting consensus algorithms for each scenario. Currently, the introduction of TEE [39] into Blockchains to ensure their high performance and security is increasingly favored by researchers. If a weakly centralized TEE assistant is allowed, the performance of Blockchains can be improved at a low resource cost. For example, in MinBFT [62], the TEE provides a tamperproof trusted counter service guaranteeing that a malicious replica would not be able to make different correct replicas execute different operations as their i-th operation. Introducing TEE technology into Blockchain is one of the aspects that we intend to focus on in future research.

Acknowledgements This work was supported by National Key R&D Program of China (Grant No. 2016YFB1000100), National Natural Science Foundation of China (Grant No. 61772030), and GF Innovative Research Program.

References

1 Nakamoto S. Bitcoin: a peer-to-peer electronic cash system. 2008. http://bitcoin.org/bitcoin.pdf

2 Zheng Z B, Xie S A, Dai H N, et al. Blockchain challenges and opportunities: a survey. Int J Web Grid Serv, 2018, 14: 352

3 Yuan Y, Wang F Y. Blockchain: the state of the art and future trends. Acta Autom Sin, 2016, 42: 481–494

4 Wood G. Ethereum: a secure decentralised generalised transaction ledger. 2014. http://gavwood.com/Paper.pdf

5 Yuan Y, Wang F Y. Blockchain and cryptocurrencies: model, techniques, and applications. IEEE Trans Syst Man Cybern Syst, 2018, 48: 1421–1428

6 Christidis K, Devetsikiotis M. Blockchains and smart contracts for the internet of things. IEEE Access, 2016, 4: 2292–2303

7 Sharma P K, Chen M Y, Park J H. A software defined fog node based distributed blockchain cloud architecture for IoT. IEEE Access, 2018, 6: 115–124

8 Chen Q, Bridges R A. Automated behavioral analysis of malware: a case study of wannacry ransomware. In: Proceedings of the 16th IEEE International Conference on Machine Learning and Applications (ICMLA), 2017. 454–460

9 Bencic F M, Zarko I P. Distributed ledger technology: blockchain compared to directed acyclic graph. In: Proceedings of the 38th International Conference on Distributed Computing Systems (ICDCS), 2018. 1569–1570

10 Chen Z D, Dong A Q, Sun H, et al. Research on private blockchain based on crowdfunding. J Inf Secur Res, 2017, 3: 227–236

11 Pongnumkul S, Siripanpornchana C, Thajchayapong S. Performance analysis of private blockchain platforms in varying workloads. In: Proceedings of the 26th International Conference on Computer Communication and Networks (ICCCN), 2017

12 Lamport L, Shostak R, Pease M. The Byzantine generals problem. ACM Trans Program Lang Syst, 1982, 4: 382–401

13 Perry K J, Toueg S. Distributed agreement in the presence of processor and communication faults. IEEE Trans Softw Eng, 1986, 12: 477–482

14 Lamport L. The part-time parliament. ACM Trans Comput Syst, 1998, 16: 133–169

15 Lamport L. Paxos made simple. ACM Sigact News, 2001, 32: 18–25

16 Ongaro D, Ousterhout J. In search of an understandable consensus algorithm. In: Proceedings of USENIX Annual Technical Conference, 2014. 305–319

17 Fischer M J, Lynch N A, Paterson M S. Impossibility of distributed consensus with one faulty process. J ACM, 1985, 32: 374–382

18 Karame G, Androulaki E, Capkun S. Two bitcoins at the price of one? Double-spending attacks on fast payments in bitcoin. 2012. https://eprint.iacr.org/2012/248.pdf

19 Karame G. On the security and scalability of bitcoin's blockchain. In: Proceedings of ACM Sigsac Conference on Computer and Communications Security, 2016. 1861–1862

20 King S, Nadal S. Ppcoin: peer-to-peer crypto-currency with proof-of-stake. 2012. https://decred.org/research/king2012.pdf

21 Castro M, Liskov B. Practical byzantine fault tolerance. In: Proceedings of Symposium on Operating Systems Design and Implementation, 1999. 173–186

22 Bano S, Sonnino A, Al-Bassam M, et al. Consensus in the age of blockchains. 2017. ArXiv:1711.03936

23 Bach L, Mihaljevic B, Zagar M. Comparative analysis of blockchain consensus algorithms. In: Proceedings of the 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2018. 1545–1550

24 Wang W, Hoang D T, Hu P, et al. A survey on consensus mechanisms and mining strategy management in blockchain networks. IEEE Access, 2019, 7: 22328–22370

25 Eyal I, Gencer A E, Sirer E G, et al. Bitcoin-ng: a scalable blockchain protocol. In: Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI16), 2016. 45–59

26 Gilad Y, Hemo R, Micali S, et al. Algorand: scaling byzantine agreements for cryptocurrencies. In: Proceedings of the 26th Symposium on Operating Systems Principles, 2017. 51–68

27 Zhang Z W. A byzantine fault-tolerant algorithm for blockchains. 2016. https://docs.neo.org/en-us/basic/consensus/whitepaper.html

28 Churyumov A. Byteball: a decentralized system for storage and transfer of value. 2016. https://byteball.org/Byteball.pdf

29 Baird L. The Swirlds Hashgraph Consensus Algorithm: Fair, Fast, Byzantine Fault Tolerance. Swirlds Tech Reports SWIRLDS-TR-2016-01. 2016

30 Micali S, Rabin M, Vadhan S. Verifiable random functions. In: Proceedings of the 40th Annual Symposium on Foundations of Computer Science, 1999. 120–130

31 Li X Q, Jiang P, Chen T, et al. A survey on the security of blockchain systems. Future Generation Comput Syst, 2020, 107: 841–853

32 Fu X, Wang H M, Shi P C, et al. Jointgraph: a DAG-based efficient consensus algorithm for consortium blockchains. Softw-Pract Exper, 2019, 42: 2748

33 Rachmawati D, Tarigan J T, Ginting A B C. A comparative study of message digest 5(MD5) and SHA256 algorithm. J Phys-Conf Ser, 2018, 978: 012116

34 O'Dwyer K J, Malone D. Bitcoin mining and its energy footprint. In: Proceedings of the 25th IET Irish Signals & Systems Conference, 2014

35 Li C X, Li P L, Xu W, et al. Scaling nakamoto consensus to thousands of transactions per second. 2018. ArXiv:1805.03870

36 Larimer D. Delegated Proof-of-Stake (DPOS). Bitshare whitepaper, 2014

37 Milutinovic M, He W, Wu H, et al. Proof of luck: an efficient blockchain consensus protocol. In: proceedings of the 1st Workshop on System Software for Trusted Execution, 2016

38 Karantias K, Kiayias A, Zindros D. Proof-of-burn. In: Proceedings of International Conference on Financial Cryptography and Data Security, 2020. 523–540

39 Sabt M, Achemlal M, Bouabdallah A. Trusted execution environment: what it is, and what it is not. In: Proceedings of the 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, 2015

40 Fabric H. Simple BFT. 2018. https://jira.hyperledger.org/browse/FAB-378

41 Androulaki E, Barger A, Bortnikov V, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In: Proceedings of European Conference on Computer Systems, 2018

42 Yin M F, Malkhi D, Reiter M K, et al. Hotstuff: BFT consensus with linearity and responsiveness. In: Proceedings of ACM Symposium on Principles of Distributed Computing, 2019

43 Stathakopoulou C, David T, Vukolić M. Mir-BFT: high-throughput BFT for blockchains. 2019. ArXiv:1906.05552

44 Baudet M, Ching A, Chursin A, et al. State machine replication in the libra blockchain. 2019. https://developers.libra.org/docs/assets/papers/libra-consensus-state-machine-replication-in-the-libra-blockchain/2019-09-19.pdf

45 Amsden Z, Arora R, Bano S, et al. The libra blockchain. 2019. https://developers.libra.org/docs/the-libra-blockchain-paper

46 Miller A, Xia Y, Croman K, et al. The honey badger of BFT protocols. In: Proceedings of ACM SIGSAC Conference on

Computer and Communications Security, 2016. 31–42

47 Ben-Or M, Kelmer B, Rabin T. Asynchronous secure computations with optimal resilience. In: Proceedings of the 13th Annual ACM Symposium on Principles of distributed computing, 1994. 183–192

48 Fu X, Wang H M, Shi P C. Proof of previous transactions (PoPT): an efficient approach to consensus for JCLedger. IEEE Trans Syst Man Cybern Syst, 2019. doi: 10.1109/tsmc.2019.2913007

49 Popov S. The tangle. 2018. http://www.descryptions.com/Iota.pdf

50 Bentov I, Lee C, Mizrahi A, et al. Proof of activity: extending bitcoin's proof of work via proof of stake. 2014. https://eprint.iacr.org/2014/452.pdf

51 Buchman E. Tendermint: Byzantine fault tolerance in the age of blockchains. Dissertation for Ph.D. Degree. Guelph: University of Guelph, 2016

52 Buterin V, Reijsbergen D, Leonardos S, et al. Incentives in Ethereum's hybrid casper protocol. 2019. ArXiv:1903.04205

53 Dinh T T A, Wang J, Chen G, et al. Blockbench: a framework for analyzing private blockchains. In: Proceedings of ACM International Conference on Management of Data, 2017. 1085–1100

54 Nguyen G, Kim K. A survey about consensus algorithms used in blockchain. J Inf Process Syst, 2018, 14: 101–128

55 Chalaemwongwan N, Kurutach W. State of the art and challenges facing consensus protocols on blockchain. In: Proceedings of International Conference on Information Networking (ICOIN), 2018. 957–962

56 Douceur J R. The sybil attack. In: Proceedings of International Workshop on Peer to Peer Systems, 2002

57 Singh A, Ngan T W, Druschel P, et al. Eclipse attacks on overlay networks: threats and defenses. In: Proceedings of IEEE International Conference Computer and Communications, 2006

58 Kroll J A, Davey I C, Felten E W. The economics of bitcoin mining, or bitcoin in the presence of adversaries. In: Proceedings of the 12th Workshop on the Economics of Information Securit, 2013

59 Wang H M, Shi P C, Zhang Y M. Jointcloud: a cross-cloud cooperation architecture for integrated internet service customization. In: Proceedings of IEEE International Conference on Distributed Computing Systems, 2017. 1846–1855

60 Liang J, Han W L, Guo Z Q, et al. DESC: enabling secure data exchange based on smart contracts. Sci China Inf Sci, 2018, 61: 049102

61 Wu Y B, Fan H N, Wang X Y, et al. A regulated digital currency. Sci China Inf Sci, 2019, 62: 032109

62 Veronese G S, Correia M, Bessani A N, et al. Efficient Byzantine fault-tolerance. IEEE Trans Comput, 2013, 62: 16–30