Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Low powered blockchain consensus protocols based on consistent hash*

Lei YU[†‡1,2], Xiao-fang ZHAO[2], Yan JIN[2], Heng-yi CAI[1,2], Bo WEI[1,2], Bin HU[1,2]

[1]*University of Chinese Academy of Sciences, Beijing 100190, China*

[2]*Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China*

[†]E-mail: yulei@ncic.ac.cn

Received Feb. 23, 2018; Revision accepted May 18, 2018; Crosschecked Sept. 4, 2019

Abstract: Current blockchain consensus protocols have a triangle of contradictions in aspects of decentralization, security, and energy consumption, and cannot be synchronously optimized. We describe a design of two new blockchain consensus protocols, called "CHB-consensus" and "CHBD-consensus," based on a consistent hash algorithm. Honest miners can fairly gain the opportunity to create blocks. They do not consume any extra computational power resources when creating new blocks, and such blocks can obtain the whole blockchain network to confirm consensus with fairness. However, malicious miners have to pay massive computational power resources for attacking the new block creation privilege or double-spending. Blockchain networks formed by CHB-consensus and CHBD-consensus are based on the same security assumption as that in Bitcoin systems, so they save a huge amount of power without sacrificing decentralization or security. We analyze possible attacks and give a rigorous but adjustable validation strategy. CHB-consensus and CHBD-consensus introduce a certification authority (CA) system, which does not have special management or control rights over blockchain networks or data structures, but carries the risk of privacy breaches depending on credibility and reliability of the CA system. Here, we analyze the robustness and energy consumption of CHB-consensus and CHBD-consensus, and demonstrate their advantages through theoretical derivation.

Key words: Blockchain; Consensus protocol; Consistent hash; Low energy consumption; Decentralization
https://doi.org/10.1631/FITEE.1800119 **CLC number:** TP301

1 Introduction

1.1 Blockchain technology

The Bitcoin (Nakamoto, 2008) technology that emerged after 2008 has been followed widely in recent years, and is considered a pioneering practical system for cryptographic digital currency. The underlying technology platform for digital currency is blockchain. The core protocol can be summarized as a combination of the following technical terms: peer-to-peer (P2P) network, signature verification based on an asymmetric cryptographic algorithm, consensus of transaction information in the current period to the whole network, and blockchain data structures linked by a one-way hash function. These terms were described in detail in Nakamoto (2008). The essence of blockchain technology can be viewed as a distributed database that holds historical transactional data shared by all nodes through a distributed consensus protocol (Yuan and Wang, 2016). The core values of blockchain technology include decentralization, a distributed consensus protocol, digital signature, cryptography based on an asymmetric public key

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (Nos. 61672499, 61772502, and 61972382), the Key Special Project of Beijing Municipal Science & Technology Commission, China (No. Z181100003218018), the Natural Science Foundation of Inner Mongolia, Open Foundation of State Key Laboratory of Networking and Switching Technology of China (No. SKLNST-2016-2-09), the SV-ICT Blockchain & DAPP Joint Lab of China, and the ICT-SSC Blockchain Joint Lab of China

ORCID: Lei YU, http://orcid.org/0000-0003-4316-1763

mechanism, and a timestamp. P2P transactions are based on decentralized credit in distributed systems without nodes that trust each other. This provides a solution to the problems of high cost, credit monopoly, and reliability associated with dependency on a single point, which are prevalent in centralized institutions. The advent of the blockchain solves two major problems of digital currency: double-spending and the Byzantine generals problem (Lamport et al., 1982; Lamport, 1983; Reischuk, 1985; Fedotova and Veltri, 2006; Nelson, 2007; Fan et al., 2013). Blockchain technology in the financial, insurance, online payment, notarization, and other fields has broad application prospects.

Blockchain technology in practical applications can be divided into two basic types based on whether the transactional link information or the account change information is recorded in the blockchain. In this study, we use only the transaction blockchain protocol. Currently, transaction blockchain technology does not have an industry standard. The basic blockchain data structure is shown in Fig. 1.

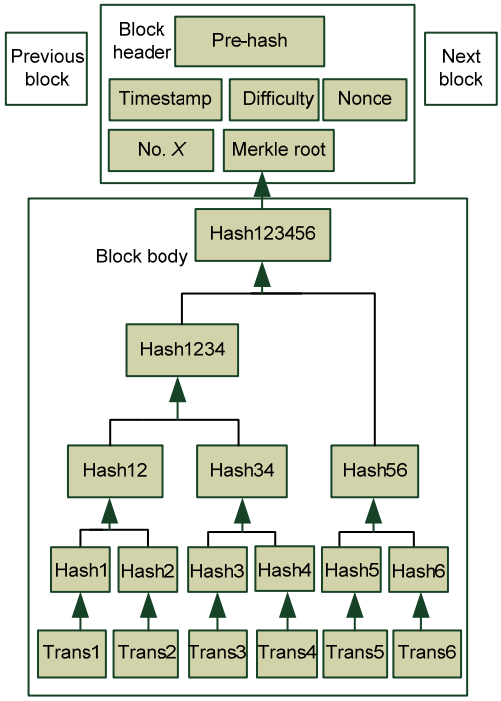

Fig. 1 Basic data structure of a blockchain

The blockchain data structure, the Merkle tree of transaction information, and the consensus mechanism ensure that historical transaction data is extremely difficult to tamper with. The transaction data in each block is the transaction information in the corresponding period. The logical link structure of transactions is shown in Fig. 2.

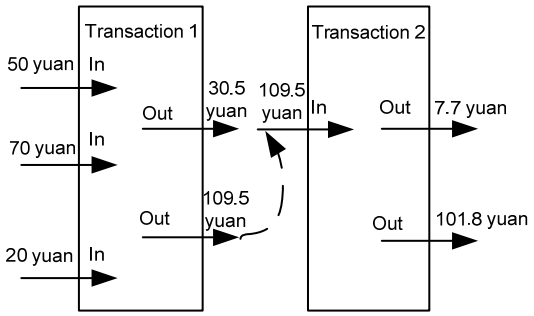

Fig. 2 Logical link structure of transactions in a blockchain

Taking Bitcoin as an example, starting from the creation block, the historical ledger data of the blockchain contains all chains consisting of the end-to-end transfer transactions of digital as-sets. The outputs of the previous transaction become the inputs of the current transaction. The current transaction outputs are taken as the next transaction inputs. The data structure of each transaction is as follows:

```
{
        hash:[ ]      // The hash value of the current
// transaction
        in:   // Source of income
        {
            prev_out:              // Pre-transaction
// information
            {
                hash:[ ] // The hash value of the
// pre-transaction
                n:[ ]      // Pre-transaction output
// index
            }
            scriptSig:  <sig>  <pubKey>
// Signature and public key of the owner of the
// pre-transaction
        }
        out:   // Where to spend
        {
            value:[ ] // Payment amount
            address:[ ] // Recipient address of hash
            scriptPubKey: // Output script content
```

{

Validate that the user's public key address matches the current public key address of hash (based on the irreversible hashing encryption algorithm, such as SHA256)

Validate that the user's digital signature matches the user's public key (based on the asymmetric cryptography algorithm, such as ECDSA)

}

}

}

End-to-end transaction data (including the digital signature of the sender) is validated by the content of script in transactions, and is recorded by blocks with different timestamps to form the data body of the interlinked blockchain. The nodes in the blockchain network compete for the accounting privilege of transactions through the consensus process to avoid the problem of double-spending and to prevent historical transaction data from being easily tampered with.

To satisfy decentralized requirements and reliability, blockchain technology is generally based on a P2P network. Each node in the network is connected and interacts with others in a flat topological network structure without any special centralized nodes. There is no hierarchical structure. Each node will assume the network routing, verify the transactions, block data, and propagation transaction data, and discover new nodes.

1.2 Consistent hash algorithm

Karger et al. (1997) proposed a consensus hash algorithm. The goal is to achieve a load or replica distribution balancing in a dynamically distributed system with dynamic adaptability and efficiency. The consistent hash algorithm is based on the hash algorithm; the original intention is to solve the "hot spot" in a distributed system. The consistent hash algorithm maps the entire hash space into a virtual ring. The entire hash space ranges from 0 to $2^{32}-1$, and is organized clockwise. 0 and 2^{32} are coincident. The schematic is shown in Fig. 3.

The classic problem solved by the consistent hash algorithm is load balancing; that is to say, a large amount of load data is evenly distributed to the server cluster consisting of nodes. First, all nodes compute

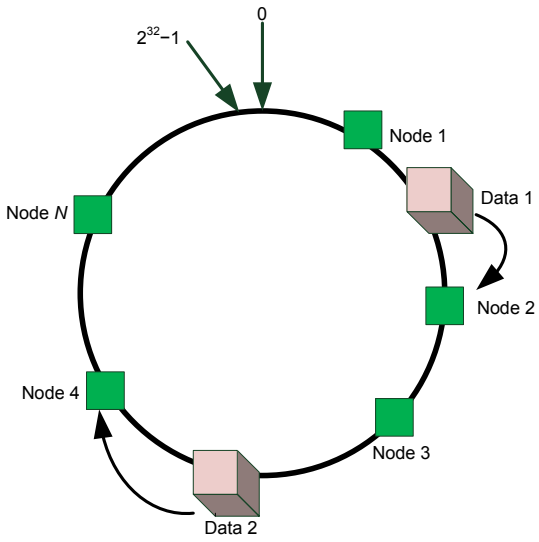

Fig. 3 Principle of the consistent hash algorithm

hash values, map each node value to a certain position on the hash ring, and compute the hash value $H(d)$ of some data or load. The $H(d)$ is mapped onto this hash ring, and the first node found clockwise from the value of $H(d)$ is the node processing data or load. This method is suitable for the scenes of dynamic entry and exit of a node in the distributed system. When a node comes in or out freely, only the neighbor node is affected. It is easy to see that due to the uncertainty of the node hash value, the node number, and the data or load, it is difficult to achieve complete balance among all nodes. An improved consistent hash algorithm is proposed, which generates a corresponding number of virtual nodes for each real node based on the processing power of each real node. The algorithm equilibrates the virtual node onto the hash ring and data or load mapping on the hash ring ID corresponding to a virtual node. The data or load is processed by the real node corresponding to the virtual node.

In this study, a blockchain consensus protocol based on a consistent hash algorithm, called "CHB-consensus," is designed for a blockchain of transactions.

2 Background

The aim of consensus in a blockchain is to achieve the unanimous confirmation of legitimate and

the ordered transactions by the nodes involved in the whole blockchain network. Historical transactions that have been confirmed online will not be tampered with (or it will be difficult to do so) by malicious nodes. The consensus protocol of a blockchain has always been a hot research topic.

The core idea of proof-of-work (PoW) comes from the research on preventing spam (Dwork and Naor, 1992). Back (2002) first proposed the HashCash concept and the workload proof method based on the hash function. The PoW consensus protocol designed by Nakamoto (2008) effectively avoids a Sybil attack (Douceur, 2002) that may occur when a non-trusted anonymous node enters or exits freely. It is one of the most widely used blockchain consensus protocols. Many blockchain systems (including Bitcoin) use PoW or improved PoW. The PoW protocol requires each node solve a computational problem, but easily verifies the SHA256 computational problem based on its own computational effort. Each node has to find a suitable random number of "Nonce." The input to the block header metadata and Nonce is an SHA256 hash value computed twice in succession, so that the result is less than the difficulty target set in the header of the block.

$$H\left(\text{nVersion, hashPreBlock, hashMerkleroot, nTimes, nBits, Nonce}\right) < \frac{\text{MaxTarget}}{\text{Diff}}, \tag{1}$$

where nVersion presents the current protocol version number, hashPreBlock the hash value of the previous block, hashMerkleroot the transaction Merkel root of current block, nTimes the timestamp of the current block, nBits the storage form of the difficulty value in the block header, Nonce a random number that meets the difficulty requirements, MaxTarget the maximum target value of the SHA256 calculation, and Diff the current difficulty value.

Parameters of the SHA256 hash function come from the block header metadata of the current block to be built. Due to the irreversibility of the (twice) SHA256 hash, the node has to pay enough computational power to perform the Nonce search to ensure that the result is as small as possible (as in Bitcoin mining). The PoW consensus calls for honesty nodes to create a new block (mining) that consumes enough computational power. A double-spending attack on a malicious node has to take 51% of the computational power of the entire network to obtain a sufficiently high probability of a successful attack. The advantage of PoW is that the algorithm is simple and the fault tolerance is up to 50%. The disadvantage is that the honesty nodes' competition mining wastes massive resources. According to reliable data, Bitcoin's mining power has surpassed the sum of the force of the world's top 500 computers. To ensure the stability of the Bitcoin system, the difficulty value is adjusted in cycles (every 2016 blocks) to ensure an average interval of 10 min between blocks. A reliable confirmation of a transaction usually requires linking six blocks, resulting in a confirmation of the transaction taking 60 min. Moreover, the big mine pool caused by the PoW mechanism has threatened the decentralization of the Bitcoin network.

The PoW protocol has been widely used in encrypted digital currency due to its security, decentralization, and simplicity. However, its high energy consumption has attracted much attention (O'Dwyer and Malone, 2014; Giungato et al., 2017; Mishra et al., 2017; Vranken, 2017). The PoW protocol requires all nodes who participate in the consensus process pay massive computational power to find the expected value of the hash puzzle to create a new block. This process, called "mining," makes the blockchain network consume a large amount of electrical energy in the period of creating blocks. Giungato et al. (2017) found that 40% of the total value of Bitcoin (using the PoW protocol) is equivalent to the electricity consumption. Furthermore, the cryptocurrency may have generated four gigatons of greenhouse gases, almost 13% of global releases per annum. The current estimated annual electricity consumption of Bitcoin mining is about 39.5 TW·h, which exceeds the annual consumption of the whole countries like Qatar and Bulgaria (https://digiconomist.net/bitcoin-energy-consumption). Bitcoin mining may consume the world's entire electrical energy by February 2020 due to its exponential growth (https://powercompare.co.uk/). An analysis in Mishra et al. (2017) using public miners' data concluded that if Bitcoin's transaction throughput reaches the level of the Visa and PayPal systems, the energy consumption will not be affordable. In summary, if blockchain applications use the PoW protocol, they are unlikely to be environmentally sustainable.

In the view of shortcomings of PoW, Bahri and Girdzijauskas (2018) explored an alternative mechanism for blockchain operation, which provides the same P2P transaction capabilities without having to consume such huge amounts of energy. The approach waives energy consumption by trust, and introduces the concept of proof-of-trust (PoT) blockchains. In a PoT blockchain, peer trust is evaluated in the network based on a trust graph that emerges in a decentralized fashion and that is encoded in, and managed by, the blockchain itself. Then this trust is used as a waiver for the difficulty of POW; that is, the more trust the miners prove in the network, the less work the miners do. Milutinovic et al. (2016) used hardware based on the trusted execution environments (TEEs) to implement a low energy consensus protocol called "proof of luck." The proof of luck blockchain uses TEE platform's random number generation to choose a consensus leader, which offers low-latency transaction validation, deterministic confirmation time, negligible energy consumption, and equitably distributed mining.

The consensus protocol of proof-of-stake (PoS) has been proposed for the first time in the Bitcoin Talk Forum in 2011. The current interest of node could be taken as the voucher of the mining difficulty. Peercoin implements a PoS protocol, in which the block generation difficulty is inversely proportional to node's shareholding (King and Nadal, 2012), and age and currency constitute two proofs of equity. The core algorithm is expressed as

$$\mathrm{Hash}(T,c) \le d \cdot T.\mathrm{time} \cdot T.\mathrm{value}, \tag{2}$$

where T represents a transaction not yet used by the node, c the current status of the node, T.time the ownership duration of transaction T (e.g., the age of the currency), and T.value the currency value which is available for transaction T. The random search space for the PoS mechanism is limited, and a hashing attempt computation can be made every second. Inequality (2) shows that the difficulty of generating a block is inversely proportional to the currency age of a transaction owned by a node and to the currency value. The method of violent search by PoW is no longer used.

PoS consensus solves the PoW consensus energy consumption issues, and can shorten the interval between blocks. However, when the network is poorly synchronized with the PoS consensus, the creation cycle of each block will create multiple blocks, causing the blockchain to bifurcate. Malicious nodes can control network communication through their privilege to generate a block. They can send different blocks to different network partitions to form a double-spending attack. The PoS consensus cannot guarantee fairness in the initial formation of blockchain, as a handful of nodes with sufficient currency age and currency value are likely to generate blocks.

Delegated proof-of-stake (DPoS) (Asolo, 2018) consensus is based on the PoS; however, the privilege to create a block is specialized. First, a committee is selected based on the voting rights of each node, and each member of the committee creates a new block in turn. Members of the committee must guarantee that the statuses of 90% of all are online. The committee controls the privilege of new block creation, which can increase efficiency and achieve consistent confirmation in seconds. The disadvantages of the DPoS are obvious. When committee members become malicious nodes, resulting in a double-spending block, other nodes will be unable to do anything; DPoS is not completely decentralized.

The practical Byzantine fault tolerance (PBFT) (Castro and Liskov, 1999) consensus protocol is based on the messaging mode. When the number of malicious nodes F is smaller than $(n+1)/3$ (n represents the total number of nodes), the system can reach a consensus on a certain value. After three stages of network messaging within all nodes, the honest nodes can agree on a certain originating value. When PBFT is applied to a blockchain, only a primary node generates a block in each consensus period, avoiding bifurcation of the blockchain and the waste of massive computational power. It shortens the interval between blocks and the period of transaction confirmation. However, the PBFT consensus process cannot cope with a Sybil attack. Malicious nodes can generate multiple nodes, resulting in more than $(n+1)/3$ malicious nodes being across the network and compromising consistency and security. Since each block is generated by the primary node, the PBFT protocol is not suitable for the large-scale network nodes but suitable for the federation application of a blockchain.

3 Consensus mechanism based on a consistent hash algorithm

The security of CHB-consensus is based on the following assumptions:

Assumption 1 The asymmetric cryptographic algorithm is public. In the case of a known public key, it is not feasible to solve the private key through algorithm inversions, or randomly to search for possible private key space. It is not feasible to fake the digital signature.

Assumption 2 The hash digest algorithm cannot be inverted.

Assumption 3 The percentage of honest nodes in the blockchain network exceeds 50%. If the percentage of non-honest nodes is more than 50%, the blockchain system will be worthless.

Assumption 4 The consistent hash algorithm contains enough space to hold enough nodes on the hash ring. It ensures that any node's mappings on the hash ring do not overlap, and it is impossible to reverse.

The security of CHB-consensus is based on Assumptions 1–4: the first three are cornerstones of Bitcoin security assurance, and the fourth is the basic element of the consistent hash algorithm. Therefore, the premise of security of the CHB-consensus protocol is no harsh based on the basic cryptography-based security. All the algorithms mentioned in this study are based on Assumptions 1–4.

CHB-consensus does not make any hypotheses for certification authority (CA). CA does not introduce any additional security issues. Algorithms and protocols of CHB-consensus ensure that the CA does not have any special control or operational authority over the blockchain system. However, CA may introduce privacy leaks.

3.1 Data structure

The blockchain data structure of CHB-consensus follows the basic structure of a transaction blockchain (similar to the data structure and block structure of Bitcoin).

The blockchain data structure is shown in Fig. 4. The properties of each field are described in Tables 1 and 2.

Table 1 Block header

Name	Description
Pre-hash	Hash value of the previous block
No. *X*	The current block number
Timestamp	Block timestamp
Trans Merkle root	Merkle root of transactions
BaseCoinSig	Digital signature of token reward transactions
Cert-Merkle root	Merkel root of the digital certificate serial number

Table 2 Block body

Name	Description
Transaction-num	Number of transactions
Transactions	Transactions sorted by timestamp
Cert-num	Number of CA digital certificate serial numbers
Cert-serialNums	CA digital certificate serial numbers sorted by number
CoinBase-Trans	CoinBase transaction

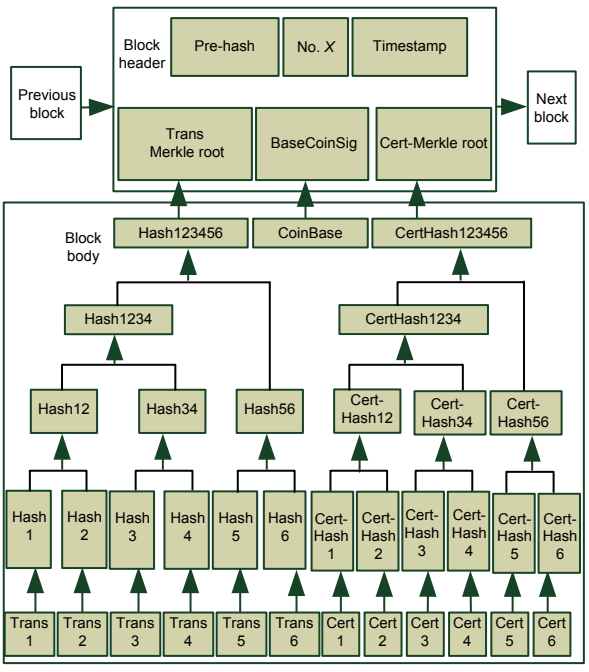

Fig. 4 Blockchain data structure of CHB-consensus

3.2 Network structure

The blockchain network formed by CHB-consensus follows the P2P network protocol; that is, each node receives and broadcasts transactions and blocks.

3.3 Public key infrastructure (PKI)/certification authority (CA) system

To ensure confidentiality, authenticity, integrity, and non-repudiation of information in the process of network information transmission, public key encryption mechanisms are commonly used. The public key needs to be transmitted over the Internet. Therefore, an important issue that needs to be solved is the public key trust problem. This is the role of the public key infrastructure (PKI). PKI is the foundation of current network security construction. A PKI system is composed of a CA, a key management center (KMC), registration agencies, directory services, security authentication application software, and certificate application services. Among these, CA plays a key role in the PKI system. To ensure the CA's credibility, it must be highly authoritative and impartial. The core function of the CA is to bind the user's public key and other users' identification information together through the registration process to form a digital certificate for the user (the user's private key is still in the user's local privacy management). As a CA, it must have the following basic functions:

1. Receive the certificate application and review the identity of the applicant;
2. Issue digital certificates;
3. Certificate management;
4. Query certificates and certificate status.

To avoid the influence of a Sybil attack on fairness, CHB-consensus requires that the node of the blockchain network should participate in the transaction with a unique public-private key of an asymmetric cryptographic algorithm and discard the random public-private key used in Bitcoin transactions. So, CHB-consensus introduces PKI/CA system management and digital certificate technology for a unique public-private key; that is, the node of the blockchain is first registered in the CA center to obtain a valid digital certificate (the digital certificate includes a public key, and the private key is stored by the participating nodes using local privacy management). The CA center provides the inquiry digital certificate function through a digital certificate serial number. CA can verify the legitimacy and correctness of digital certificates.

CHB-consensus sacrifices some transaction privacies (relative to Bitcoin), and the CA through managing digital certificates becomes a "privacy single point" issue. However, this is exactly what some scenarios require.

CHB-consensus ensures that the CA management does not have any special authority of management and control over the formed blockchain network and data structure. The CA mechanism does not introduce any additional blockchain security issues.

3.4 Consensus implementation

The consensus protocol of competition mode provides stability and robustness of the blockchain technology system. The CHB-consensus protocol adopts the consensus of competition mode. However, instead of relying on computational power competition, the nodes compete in a gambling way for betting the privilege of creating a block at random and for tokens included in the block. Tokens can be obtained based on the pseudo-random component of the consistent hash algorithm.

Each participating node obtains a unique digital certificate by registering with CA, and completes the signature and verification of the transaction process using the public and private keys corresponding to the certificate.

3.4.1 Creating a new block

Assume that the total number of nodes in the blockchain network is n, the current blockchain height is h, and the number of to-be-built blocks is $h+1$. The CHB-consensus protocol provides that every new block is generated in every time period T. T is dynamically adjustable, depending on network bandwidth and the transaction volume in a unit of time. Specific steps are as follows:

Step 1: Within each new period T, each node broadcasts its own digital certificate serial number and collects the digital certificate serial numbers of other nodes in the network. Each node verifies the validity of the digital certificate serial number and forwards it. At the end of the period T, all the digital certificate serial numbers collected by nodes are sorted in numerical order to generate a digital certificate serial number set $\mathrm{Ce}(h+1)\{S_1, S_2, \ldots, S_n\}$. A Cert-Merkle tree is generated by the certificate set $\mathrm{Ce}(h+1)$ and is written into the header and body of the to-be-built block data structure.

Step 2: Within each new period T, each node can issue and broadcast a new transaction. Each node

receives new transactions on the network, verifies their correctness and legality, and forwards them. At the end of the period T, each node will sort all new transactions according to the hash digest of the transaction, to generate a transaction Merkle tree and write the header and body of the to-be-built block data structure.

Step 3: At the end of the period T, each node uses the digital certificate serial numbers recorded in the block of the N^{th} backward block before the current block numbered ($h-N$) generates the set Ce($h+1$)$\{S_1, S_2, \ldots, S_n\}$. Then, for each serial in the set Ce($h-n$), a consistent hash is computed and mapped onto the hash ring. Each S_i is regarded as a node in the consistent hash algorithm in Section 1, forming the mapping of Ce($h-n$) on the hash ring, denoted as $R(h-n)$ rings. Each blockchain node uses the Pre-hash, No. h+1, timestamp, and transaction Merkle root as inputs, to compute a consistent hash digest value HD(h+1), treat HD(h+1) as data in the consistent hash algorithm in Section 1, calculate the mapping position $L(h-N)$ of data on the ring $R(h-N)$, and find the processing node S_i corresponding to the position $L(h-N)$ in Section 4. The processing node S_i is the digital certificate serial number (for convenience, the above consensus hash calculation process is named AW(h+1)). Fig. 5 is a description of the above calculation process. Each node judges whether S_i, the digital certificate serial number, is registered by itself. If not, the node will cancel the competition to create current block h+1 and wait for block h+1 from other nodes. Then, the node will enter the next competition of the block h+2 creation process. Otherwise, this node will obtain the privilege of creating current block h+1 and obtain the token reward contained in the block (step 4).

Step 4: The node holding S_i enters the process of creating a new block h+1, and the output address of the token transaction points to the public key (or hash value of the public key) corresponding to the digital certificate S_i of the node. The value of the token number is expressed as

$$\mathrm{Re}(h+1) = B \cdot |\mathrm{Ce}(h+1)|, \tag{3}$$

where B represents a fixed constant, and |Ce(h+1)| represents the total number of digital certificate serials contained in the current block h+1. The current node writes the token reward transaction to the body of block h+1, performs the hash function on the token reward transaction to obtain Hm, digitally signs Hm with the private key corresponding to the digital certificate S_i to form an SIG_i(Hm), writes SIG_i(Hm) to the block h+1 header as "BaseCoinSig," and broadcasts the new block h+1.

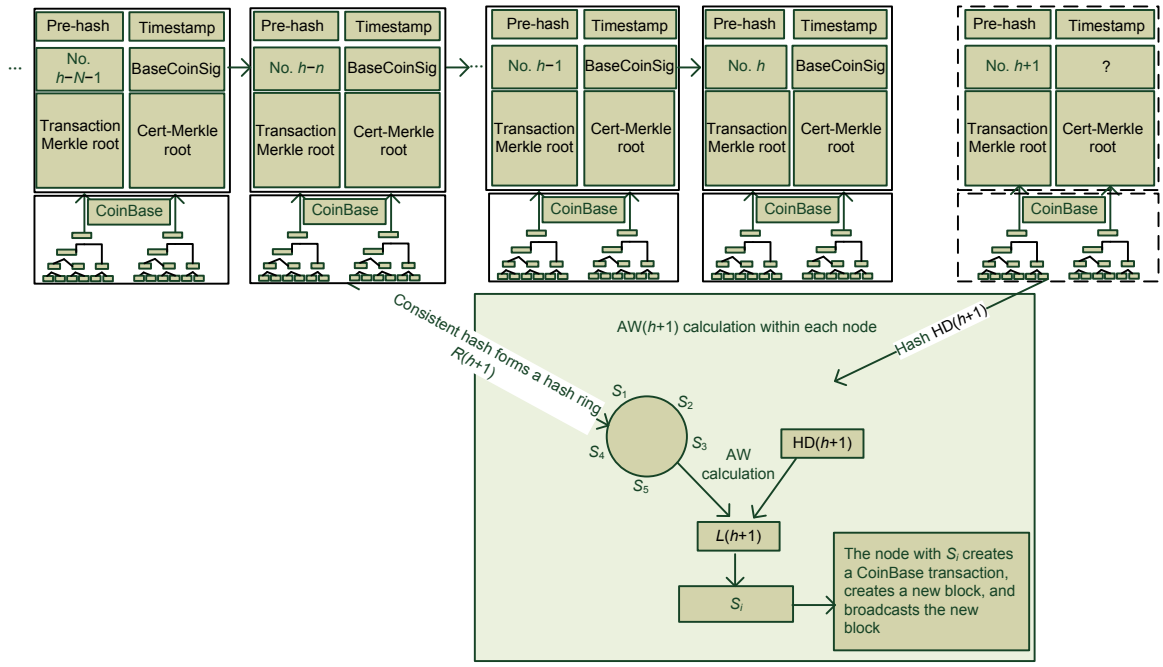

Fig. 5 AW calculation process when a block is created

Step 5: Other nodes receive the new block h+1 and perform the following validations:

Validation 1 The previous block referenced by the new block exists, and is valid.

Validation 2 The block contains the correctness of transactions.

Validation 3 Correctness and completeness of other data items exist in the block header and block body.

Validation 4 The output of the token reward transaction is pointing to an address which is different from that of the output token reward transaction contained in block h (to avoid aggression, two consecutive reward transactions sent to the same public key address are not allowed).

Validation 5 Recalculate HD(h+1), $R(h-N)$, and mapping position $L(h-N)$ on the corresponding hash ring, and validate the consistency with the calculation result of the current node in step 3.

Validation 6 Validate that the output of the token reward transaction in block h+1 is consistent with the public key address of S_i corresponding to the digital certificate serial corresponding to $L(h-N)$.

Validation 7 Validate that the digital certificate number S_j of this node is contained in the body of block h+1.

Validation 8 Validate the consistency of the digital signature BaseCoinSig in the header of block h+1 with the token reward transaction in the body of block h+1. When all the validations are passed, the node accepts the new block h+1 and links it to the end of the current blockchain.

Step 6: All nodes accept the new block h+1. Start the next competition of the creation block process, and start again from step 1.

3.4.2 Formation of the initial chain

From the block creation process described in Section 3.4.1, it can be seen that the token transaction does not exist in blocks from 1 to N, so these blocks cannot contain any other transaction. That is to say, blocks numbered from 1 to N cannot implement the CHB-consensus protocol. These blocks can contain only the digital certificate serial set of all the nodes. Therefore, the blockchain network generating the blocks numbered from 1 to N must adopt other consensus protocols, such as the PoW consensus protocol. Of course, the consensus of offline is available.

The block numbered N+1 contains only one token reward transaction. The inputs of AW calculation are Pre-hash, No. N+1, timestamp, and transaction Merkle root, but the transaction data included in the transaction Merkle root is null. After that, starting with block N+1, block creation and validation processes are the same as those described in Section 3.4.1.

3.5 Robustness analysis

3.5.1 Attack on the privilege of creating a new block

According to the data structure and the way by which the Bitcoin system operates, the link of the transaction order and the forwarding process adopts the signature and validation of the public-private key pair corresponding to the digital certificate registered from the CA. Therefore, based on the security of the asymmetric cryptographic algorithm, a malicious node cannot misappropriate transactions (including the token reward transactions) belonging directly to others (based on Assumption 2) in step 3 in Section 3.4.1, because it cannot pass Validation 2 of other nodes.

In step 3, the set of digital certificates Ce($h-N$) used by each node executing AW calculation for the computing privilege of creating a new block is taken from a historical block that has been confirmed by the whole network through CHB-consensus. Therefore, no node can manipulate and forge the set Ce($h-N$), because it will cause other nodes to fail when perform the block Validation 5. Some malicious nodes forcibly modify the set Ce($h-N$) by modifying the block header of the block numbered $h-N$, which is equivalent to that of a double-spending attack, which will be discussed later. At the same time, a malicious node may attempt to modify its own transaction in the to-be-built block h+1 (based on Assumption 2, i.e., malicious nodes cannot tamper with the transaction signed by other digital certificates). For example, add a random value to the reserved field or make a continuous change between payment and change, to make the digital certificate serial number of the AW(h+1) calculation result point to itself. In this case, based on the characteristics of the consistent hash algorithm, malicious nodes need to deal with the computational difficulty, which is equivalent to the random event of violent collision probability of $1/n$ (n represents the number of network nodes), and it needs to be completed within limited time T. It is possible that a malicious node may succeed, but it would still

not be able to pass block Validation 5 of most honest nodes. Therefore, the winner (i.e., the corresponding digital certificate serial number) of the token reward transactions is random, and any node cannot manipulate the issuance of token rewards, which is a decentralized random event.

In step 3, malicious nodes cannot replace the legal creator of the current new block (with the privilege of creating a new block) to create and broadcast the new block because of Validation 8. The goal of Validation 8 is to prevent malicious nodes from randomly deleting other nodes' digital certificate serial numbers in the to-be-built block. If the malicious node does that, it will increase the probability of malicious nodes winning the token rewards in the future, and the tampered block will still be validated as legal by most honest nodes. However, a legal creator does not have such a motivation, because a legal creator prefers to include more digital certificate serial numbers in the to-be-built block, which can earn more token rewards through Eq. (3). The more the digital certificate serial numbers included in the to-be-built blocks, the more the nodes would validate that the to-be-built block is legal and can be accepted; that is, more nodes approve that the token reward transactions have been validated. This is more valuable for honest nodes than increasing the probability of obtaining future token rewards.

3.5.2 Double-spending attack

A malicious node can neither change a block in blockchain's history by changing another node's transaction, nor forge any transaction that does not belong to itself (based on Assumption 2). Malicious nodes can carry out only double-spending attacks, i.e., changing transactions that belong to themselves in historical blocks. Assume that the total height of the current blockchain is h, and that the malicious node attempts to tamper with the block numbered X. It changes (or eliminates) the transaction that has been recorded in the block numbered X, and then signs and forwards the tampered transaction, which results in a double-spending attack that will cause blocks X to h failure (they cannot pass Validation 3). Malicious nodes have to rebuild the blocks from X to h or beyond. Starting from block X, the malicious node attempts to manipulate AW(X) calculation. As a result of tampering the transactions in block X, the calculation result of AW(X) on the hash ring $R(X-N-1)$ of block numbered $X-N-1$ would be changed, and the token reward transaction of block X will be invalid (it cannot pass block Validation 6). For block X to pass block Validation 6 again, the malicious node can manipulate only the result of AW(X) calculation, so that the result points to its own digital certificate serial number. There are two possible options for a malicious node to achieve this goal:

1. Modify the digital certificate serial number set contained in the block numbered $X-N-1$, that is, the digital certificate set $\mathrm{Ce}(h-N-1)\{S_1, S_2, \ldots, S_n\}$ contained in block $X-N-1$. Based on the security of the CA, the digital certificate serial number cannot be forged. So, the malicious node can choose to remove one or some serial numbers in $\mathrm{Ce}(X-N-1)$ by a loop attempting to collide the correct result of AW(X) at random. This will invalidate blocks $X-N$ to h because of the change in block headers from blocks $X-N$ to h. Thus, the malicious node has to retry AW($X-N$) starting from block $X-N$, and if it still recursively chooses to modify the digital certificate number set $\mathrm{Ce}(X-2N-1)$ of block $X-2N-1$, it will cause the recursive invalidation of the entire blockchain. Malicious nodes need to pay massive computational power when rebuilding the entire blockchain to search for all possible digital certificate serial number combinations and for the possibility of tampering with the transaction belonging to the malicious node. The goal of a malicious node is to have all the token reward transactions of all blocks point to the digital certificate serial number belonging to itself. This reconstruction will make malicious nodes consume massive computational power. Even if a malicious node is lucky enough to rebuild the entire blockchain in a short enough time, the rebuilt blockchain cannot be accepted by the blockchain network because of Validation 4.

2. To avoid the situation described in the first goal, the malicious node can tamper only with the transaction information in block X belonging to its own signature forwarding. To obtain a sufficient random search space, the malicious node may tamper by continuously adjusting the ratio between output and the change of transaction or adding a random value in the reserved fields of the transaction, to make the AW(X) calculation be repeated multiple times. The goal is to make the result of $L(X-N-1)$ of AW(X)

calculation point to the digital certificate serial number belonging to itself, so that a malicious node can generate a legal block X. This will make the malicious node pay massive computational power. Based on the characteristics of the consistent hash algorithm, computational difficulty is equivalent to that of a random event with a collision probability of $1/n$ (n represents the number of network nodes). Assume that the malicious node can finish the calculation within effective time, that is, successfully forging the calculation of AW(X) of block X, and then block X+1 needs to do the same forgery. The tampered reconstruction from block X to the current block h has the possibility of success of a double-spending attack, but requires the malicious node pay massive computational power in a short enough time. To avoid this kind of extremity, Validation 4 is added when honest nodes receive a new block. Block Validation 4 ensures that block X+1 will not be validated by honest nodes, and that malicious block tampering cannot reach block X+1.

In summary, when $h-X$ is larger than two, any malicious node, even if paying massive computational power, will not succeed in a double-spending attack. That is, when the number of linked blocks after block X exceeds two, the system of blockchain with CHB-consensus can give confirmation of the validity of the transaction in block X, which is better than the case defined in the Bitcoin consensus that a transaction is deemed valid after confirmation from six blocks.

3.6 Joining and leaving

If a new node chooses to join in the transaction process, it first needs to register in the CA center to obtain a valid digital certificate and the corresponding public-private key pair based on the asymmetric cryptographic algorithm. Then, the node can participate in the transaction process with digital certificate information. If the new node joins in only the transaction process, it does not need to download the blockchain data, but just queries the blockchain data from the network. Therefore, a new node joining in or leaving the transaction process does not require the blockchain network make any adjustments or perceptions. The node can also choose to terminate the transaction process.

If a new node joins in the betting process, which is controlled by CHB-consensus, it needs to broadcast its own serial number of the digital certificate in the current block creation period T. When other nodes in the network are building a new block, the sequence number is included in the current new block. After the current block is successfully created, the new node's digital certificate serial number is written into the latest block. After that, the new block is continuously created and confirmed. When the newly created block continues to the next $(N+1)^{\text{th}}$ block, the new node obtains the right to participate in the competition for token rewards in block N+1, and the new node gains a probability of about $1/n$ of obtaining the token bonus in the current block. Any node can drop its right to participate in the competition for token rewards to avoid broadcasting its own digital certificate serial numbers in any time.

4 Discussion

4.1 Fairness

For convenience, we omit the fairness problem existing in the consistent hash algorithm in the process of describing CHB-consensus. After all the digital certificate serial numbers $\mathrm{Ce}\{S_1, S_2, \ldots, S_n\}$ are mapped onto the consistent hash ring, the length of the hash interval obtained by each digital certificate serial number cannot be guaranteed to be exactly the same, but each hash interval length represents the node's winning probability. Therefore, CHB-consensus cannot guarantee that the length of each hash interval obtained by each digital certificate serial number be completely fair; however, continuous inequity between all nodes can be avoided by introducing certain and changing disturbances. For example, a new set $\mathrm{Ce}\{S_1+T, S_2+T, \ldots, S_n+T\}$ is formed by adding each digital certificate serial number to the timestamp from the current block header. All nodes' mapping states by each AW calculation will undergo unpredictable random changes, avoiding the continuous winning probability imbalance. Obviously, such a certain and changing disturbance does not affect the correctness and verifiability of CHB-consensus.

4.2 Security

4.2.1 Security and side effects of Validation 5

Validation 5 in Section 3.4.1 is an overly stringent validation which may prevent the new block from being successfully created.

When the overall network environment is poor and the volume of transactions is large, the network cannot guarantee that each node receives exactly the same transactions within the period T, but the transactions received by each node are valid. When this happens, AW calculation results by any one node are directed to an opponent node, with the result that no node can create the current new block. There are two solutions to this problem:

1. CHB-consensus gives up Validation 5. When the blockchain network is large enough and the honest nodes account for more than 50% of nodes, the resulting risk is controllable. In the absence of Validation 5, malicious nodes need massive computational power when attacking the privilege of creating new blocks. Also, the larger the number of network nodes in the blockchain, the more the computational power needed by the malicious nodes. However, honest nodes generating new blocks do not need to pay extra computational power. In this case, a new block from honest nodes will be broadcast more rapidly within the network and will more likely be accepted by most nodes in the blockchain network. This shows that the success attack rate of malicious nodes would still be very low.

Without Validation 5, in one period T, multiple nodes may generate multiple legitimate new blocks, causing a temporary branching of the blockchain. Since each branch produces a new block permanently in each period T, unlike in the PoW consensus, a branch cannot win by competing for length. Therefore, in the absence of Validation 5, additional Validation 9 is required in step 5 in Section 3.4.1 for combining multiple branches:

Validation 9 When a node receives each new block from the P2P network, it will determine whether a fork occurs or not. If a fork occurs, the node downloads multiple branches, and the following calculations are made for each branch by the node:

$$\mathrm{BranchWeight}=\sum_{i=h-b}^{i=h+1}\left[\frac{\mathrm{NumT}_i}{1+\mathrm{ownT}_i}\cdot \mathrm{Re}(i)\right], \quad (4)$$

where b represents the number of branched blocks, i the current block number, NumT_i the number of transactions contained in the current block, ownT_i the number of transactions from token reward winner signature forwarding in the current block, and $\mathrm{Re}(i)$ the number of tokens contained in the token reward transactions in the current block. The value of BranchWeight for each branch is compared and the branch with the largest BranchWeight value is the winner. If more than one branch has the largest value, the node randomly selects one of them. Multiple branches are temporarily tolerated, and the node will continue to compare by Eq. (4) when the next new block arrives, until there is only one branch.

2. Reserve Validation 5. This is feasible in a small-scale network with few nodes. By adjusting the creation period T and the parameters of the network, the probability of inconsistency of the transactions received by each node in each period T can be reduced. If an anomaly causes a new block creator to be uncertain, the period T can be extended, and then each node randomly broadcasts or forwards some transactions; therefore, each node will eventually obtain consistent transactions. This is feasible in some small-scale network scenes; however, when the number of blockchain nodes exceeds a certain threshold, this method will result in an increased network communication load.

4.2.2 Security of Validation 4

Validation 4 avoids violent attacks by a high-performance high-power node. However, it is impossible to avoid the accomplices of multiple high-performance high-power nodes. Therefore, based on the trust environment and network scale, Validation 4 may be properly adjusted to validate that the address to which the output of the token reward transactions of the current new block points is not the same as any address to which such transactions are pointing in the latest M blocks. M represents the number of accomplice nodes that may exist in the network with high computational power.

4.2.3 Parameter settings

Parameter N for the AW calculation can be empirically set, considering the security environment of the blockchain network. It is acceptable to set a value between two and six.

4.3 Changes in block data

Compared with the general blockchain structure, the blockchain data structure adds a set of digital certificate serial numbers. Assume that each serial

number is composed of eight bytes, the number of participating nodes is 100 000 in the blockchain network. The amount of additional data per block is less than 1 MB compared with the general blockchain structure, so the resulting incremental block data from CHB-consensus is controllable.

4.4 Privacy

Each node is required to use the CA-registered digital certificate to complete the signature forwarding the transaction process. CHB-consensus recommends a two-level digital certificate mechanism, namely, an identity registration digital certificate and a transaction digital certificate. The transaction digital certificate is generated by the identity registration digital certificate, and does not contain the user's private information. All nodes should use the transaction digital certificate and transaction digital certificate serial number to complete the process of CHB-consensus in the blockchain network. The CA center manages the relationship between the identity registration digital certificate and the transaction digital certificate. The privacy protection of transactional behavior in blockchain depends on the credibility of the CA. However, the CA poses no special threat to the security of blockchain data structures or transactions in historical blocks.

5 Improvement

From the foregoing description, we know that under the guarantee of a CHB-consensus protocol, a malicious node faces a hashing challenge to a block forgery attack (double-spending attack). The difficulty of a hashing challenge is that the violence collision probability is $1/n$ random events (n represents the number of network nodes), which will make the malicious node pay massive computational power. However, such a difficulty value is much lower than the difficulty value of the current PoW consensus protocol adopted by public blockchains (such as Bitcoin). Moreover, the computational power attacking difficulty value of CHB-consensus cannot be flexibly adjusted. The relatively fixed difficulty value is an opportunity for a large computational power node to enter the network to perform a double-spending attack. Therefore, in this section, we propose an improved CHB-consensus protocol, called "CHBD-consensus," which adopts a two-phase proof to provide stronger security guarantees than CHB-consensus. At the same time, CHBD-consensus has minimal impact on energy consumption.

5.1 CHBD-consensus

The CHBD-consensus protocol is divided into two phases. In the first phase, a new block creator is selected using the same method as CHB-consensus in Section 3.4.1. Afterwards, entering the second phase, the creator of the new block is required to provide PoW by solving a hash puzzle with difficulty.

The method of PoW in the second phase of CHBD-consensus is borrowed from Bitcoin. The difficulty value (Diff) is given in the block header. An appropriate random value is violently searched in the Nonce field of the block header by the creator, so that the target hash value, result of the double SHA256 operation of the block header, satisfies the requirement of the Diff. The field meaning of the block header of CHBD-consensus is shown in Table 3.

Table 3 Block header of the CHBD-consensus

Name	Description
Pre-hash	Hash value of the previous block
No. X	The current block number
Timestamp	Block timestamp
Trans Merkle root	Merkle root of transactions
Cert-Merkle root	Merkle root of the digital certificate serial number
BaseCoinSig	Digital signature of the token reward transaction
Diff	Difficulty value
Nonce	Random number that meets the difficulty value

The inputs of the hash puzzle in the second phase of CHBD-consensus are Pre-hash, No. X, timestamp, trans Merkle root, Cert-Merkle root, BaseCoinSig, and Nonce. Assume that the difficulty value is $d_{\text{CHBD-second}}$, and that the new block creator searches for the appropriate Nonce, making the target value of the double SHA256 of the new block satisfy the following expression:

$$H\big(\text{Pre-hash}, \text{No. } X, \text{timestamp}, \text{trans Merkle root}, \text{Cert-Merkle root}, \text{BaseCoinSig}, \text{Nonce}\big) < \frac{\text{MaxTarget}}{d_{\text{CHBD-second}}}. \tag{5}$$

The basic process of the CHBD-consensus protocol is shown in Fig. 6. In the CHBD-consensus protocol, each node performs multiple validations after receiving a new block. In addition to the nine validations inherited from the CHB-consensus protocol, it also requires Validation 10:

Validation 10 Validate that the random number Nonce found by the creator makes the double SHA256 target hash value of the block header satisfy the difficulty value requirement.

5.2 Attack difficulty and energy consumption of CHBD-consensus

A double-spending attack of malicious nodes requires the results of the two hash puzzles in CHBD-consensus. The collision probability of the target value of the first phase hash puzzle is $1/n$. It can be considered that the difficulty value of the first phase hash puzzle is n. Assume that the overall attack difficulty value of CHBD-consensus is set to D_{CHBD}:

$$D_{\text{CHBD}} = n \cdot d_{\text{CHBD-second}}, \tag{6}$$

$$d_{\text{CHBD-second}} = \frac{1}{n} D_{\text{CHBD}}. \tag{7}$$

According to Eq. (6), we can flexibly adjust the attack difficulty of the CHBD-consensus protocol, and even set the attack difficulty value to be the same as that in the PoW consensus protocol of the public blockchain.

From Eq. (7), it can be seen that if the overall attack difficulty D_{CHBD} of the CHBD-consensus protocol is set to be the same as that of the PoW consensus protocol, the difficulty value of the hash puzzle in the second phase is equivalent to only $1/n$ of the PoW protocol.

Importantly, the CHBD-consensus protocol requires only a unique creator to complete the second-phase hash puzzle. Other honest nodes need to wait only for validation of the new block. This is obviously better than the PoW consensus protocol in which all

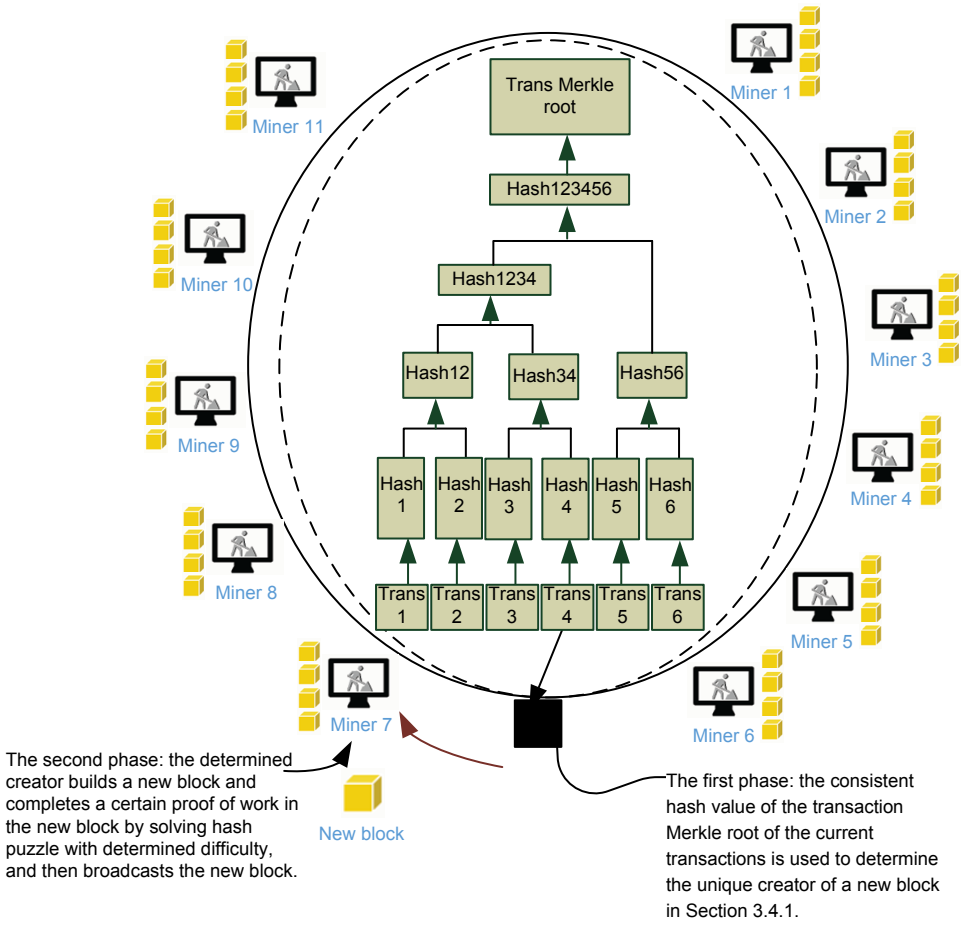

Fig. 6 Basic process of CHBD-consensus

nodes have to pay as much as possible computational power to compete for the privilege of creating blocks. Therefore, when the number of nodes of the CHBD-consensus protocol is large enough, its energy consumption is still far lower than that of the PoW protocol under the premise of the same attack difficulty.

5.3 Block interval oscillation of CHBD-consensus

In CHBD-consensus, PoW in the second phase needs to be completed separately by the creator. Because miners have different computational power resources, this will cause a significant oscillation in the block interval. A mining pool mechanism is one of the solutions. It concentrates on the computational power of a large number of nodes, contributes to the PoW of the second phase of a certain creator, and obtains the corresponding token revenue. In this way, the block interval would be significantly reduced. This is beyond the scope of this study and will be addressed in our future work.

6 Analysis of energy consumption

The main energy consumption of the PoW consensus protocol comes from the double SHA256 calculation of all miners for solving the hash puzzle. In contrast, energy consumption due to message passing in the network, block validation, and data storage can be ignored. The consensus protocols proposed in this study, i.e., CHB-consensus and CHBD-consensus, have the same message and space complexity as PoW, so in our analysis, we focus on only the energy consumption improvement that double SHA256 calculation allows.

In the PoW protocol, let D_{PoW} denote the difficulty value and V the target value. Then we can have

$$D_{\mathrm{PoW}}=V_{\max}/V, \tag{8}$$

where $V_{\max}$ denotes the largest possible value of the target. The hash function SHA256 for Bitcoin has been chosen, so that it behaves approximately as a uniformly random value between 0 and $2^{256}-1$. Thus, for any given Nonce, the probability p when it satisfies difficulty is given by

$$p=\frac{V}{2^{256}}=\frac{V_{\max}}{D_{\mathrm{PoW}}\cdot 2^{256}}=\frac{1}{D_{\mathrm{PoW}}}. \tag{9}$$

The number of search Nonce trials in the SHA256 until a block is successfully completed will be geometrically distributed; therefore, the expected number of hashes needed to find a block is D. To simplify the analysis, we assume that there are n consensus protocol participating nodes in the blockchain network, and that each node has the same computational power. In this case, suppose the number of hash trials of each node in a unit time is R. For each node, the expected time to find a block is expressed as

$$E_{\mathrm{PoW}}(t)=\frac{D_{\mathrm{PoW}}}{R}. \tag{10}$$

In the PoW protocol, n nodes perform a hash trial independently, so the expected time to find a block in the whole blockchain network is

$$E_{\text{w-PoW}}(t)=\frac{E_{\mathrm{PoW}}(t)}{n}=\frac{D_{\mathrm{PoW}}}{nR}. \tag{11}$$

The energy consumption in a unit time of each node is proportional to R, with a scaling factor a.

In the PoW protocol, the total energy consumption of the entire blockchain network during a creation period is

$$C_{\mathrm{PoW}}=naRE_{\text{w-PoW}}(t)=aD_{\mathrm{PoW}}. \tag{12}$$

In the CHBD-consensus protocol, under the same attack difficulty as under the PoW protocol ($D_{\mathrm{PoW}}=D_{\mathrm{CHBD}}$), the difficulty value of the hash puzzle in the second phase is $d_{\text{CHBD-second}}=D_{\mathrm{CHBD}}/n=D_{\mathrm{PoW}}/n$. For each node, the expected time to find a block is

$$E_{\mathrm{CHBD}}(t)=\frac{d_{\text{CHBD-second}}}{R}=\frac{D_{\mathrm{CHBD}}}{nR}=\frac{D_{\mathrm{PoW}}}{nR}. \tag{13}$$

In the CHBD-consensus protocol, there is only one node to perform the second-phase PoW in each creation period; thus, the expected time to find a block in the whole blockchain network is

$$E_{\text{w-CHBD}}(t)=E_{\mathrm{CHBD}}(t)=\frac{D_{\mathrm{PoW}}}{nR}. \tag{14}$$

The total energy consumption of the entire blockchain network during a creation period is

$$C_{\text{CHBD}} = aRE_{\text{w-CHBD}}(t) = \frac{aD_{\text{PoW}}}{n}. \quad (15)$$

Through the above analysis, it can be seen that, under the same attack difficulty and in the same hardware and network environment, the expected time to create a new block using the PoW protocol or the CHBD-consensus protocol is the same. The ratio Rt of the energy consumption of the CHBD-consensus protocol to the energy consumption of the PoW protocol is

$$\text{Rt} = \frac{C_{\text{CHBD}}}{C_{\text{PoW}}} = \frac{1}{n}. \quad (16)$$

The CHB-consensus protocol does not consume any energy for hash calculation in the whole network. Therefore, there is no comparative analysis for the CHB-consensus protocol.

7 Conclusions

Current blockchain consensus protocols cannot be optimized synchronously in terms of decentralization, security, and energy consumption. However, these three aspects are equally important. In this study, two new consensus protocols based on a consistent hash algorithm, CHB-consensus and CHBD-consensus, have been proposed. CHB-consensus and CHBD-consensus still use the unforgeability of hash computational power to reach a consensus across the blockchain network. They force the attack behavior of malicious nodes to pay massive computational power, while an honest node's creation block process does not require additional computational power. While saving energy consumption, CHB-consensus and CHBD-consensus do not sacrifice security or decentralization. We have analyzed possible attacks in detail and gave a rigorous but adjustable validation strategy. Finally, we have analyzed the issues of fairness, security, efficiency, privacy, and energy consumption, proving the advantages of CHB-consensus and CHBD-consensus. In the same hardware environment and with the same security guarantee, compared with PoW, CHB-consensus no longer consumes computational power. CHBD-consensus power consumption is $1/n$ times that of PoW. The existence of CA creates a risk of privacy leakage; however, the level of risk depends on the reliability and credibility of the CA system.

Compliance with ethics guidelines

Lei YU, Xiao-fang ZHAO, Yan JIN, Heng-yi CAI, Bo WEI, and Bin HU declare that they have no conflict of interest.

References

Asolo B, 2018. Delegated proof-of-stake (DPoS) explained. https://www.mycryptopedia.com/delegated-proof-stake-dpos-explained/

Back A, 2002. Hashcash—a denial of service counter-measure. http://www.hashcash.org/papers/hashcash.pdf

Bahri L, Girdzijauskas S, 2018. When trust saves energy: a reference framework for proof of trust (PoT) blockchains. Companion Proc Web Conf, p.1165-1169. https://doi.org/10.1145/3184558.3191553

Castro M, Liskov B, 1999. Practical Byzantine fault tolerance. Proc 3rd Symp on Operating Systems Design and Implementation, p.173-186.

Douceur JR, 2002. The Sybil attack. In: Druschel P, Kaashoek F, Rowstron A (Eds.), Peer-to-Peer Systems. Springer Berlin Heidelberg, p.251-260.

Dwork C, Naor M, 1992. Pricing via processing or combatting junk mail. Proc 12th Annual Int Cryptology Conf on Advances in Cryptology, p.139-147. https://doi.org/10.1007/3-540-48071-4_10

Fan J, Yi LT, Shu JW, 2013. Research on the technologies of Byzantine system. *J Softw*, 24(6):1346-1360 (in Chinese).

Fedotova N, Veltri L, 2006. Byzantine generals problem in the light of P2P computing. Proc 3rd Annual Int Conf on Mobile and Ubiquitous Systems: Networking & Services, p.1-5. https://doi.org/10.1109/MOBIQ.2006.340426

Giungato P, Rana R, Tarabella A, et al., 2017. Current trends in sustainability of Bitcoins and related blockchain technology. *Sustainability*, 9(12), Article 2214. https://doi.org/10.3390/su9122214

Karger D, Lehman E, Leighton T, et al., 1997. Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the World Wide Web. Proc 29th Annual ACM Symp on Theory of Computing, p.654-663. https://doi.org/10.1145/258533.258660

King S, Nadal S, 2012. PPCoin: peer-to-peer crypto-currency with proof-of-stake. https://peercoin.net/assets/paper/peercoin-paper.pdf

Lamport L, 1983. The weak Byzantine generals problem. *J ACM*, 30(3):668-676. https://doi.org/10.1145/2402.322398

Lamport L, Shostak R, Pease M, 1982. The Byzantine generals problem. *ACM Trans Programm Lang Syst*, 4(3):382-401.

Milutinovic M, He W, Wu H, et al., 2016. Proof of luck: an efficient blockchain consensus protocol. Proc 1st Workshop on System Software for Trusted Execution, p.1-6.

https://doi.org/10.1145/3007788.3007790

Mishra SP, Jacob V, Radhakrishnan S, 2017. Energy consumption—Bitcoin's Achilles heel. https://ssrn.com/abstract=3076734

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

Nelson M, 2007. The Byzantine generals problem. http://www.drdobbs.com/cpp/the-byzantine-generals-problem/206904396

O'Dwyer KJ, Malone D, 2014. Bitcoin mining and its energy footprint. Proc 25th IET Irish Signals & Systems Conf and China-Ireland Int Conf on Information and Communications Technologies, p.280-285. https://doi.org/10.1049/cp.2014.0699

Reischuk R, 1985. A new solution for the Byzantine generals problem. *Inform Contr*, 64(1-3):23-42. https://doi.org/10.1016/S0019-9958(85)80042-5

Vranken H, 2017. Sustainability of Bitcoin and blockchains. *Curr Opin Environ Sustain*, 28:1-9. https://doi.org/10.1016/j.cosust.2017.04.011

Yuan Y, Wang FY, 2016. Blockchain: the state of the art and future trends. *Acta Autom Sin*, 42(4):481-494 (in Chinese). https://doi.org/10.16383/j.aas.2016.c160158