CP-consensus: a Blockchain Protocol Based on Synchronous Timestamps of Compass Satellite

Lijing Zhou, Licheng Wang and Yiru Sun

***Abstract*—Bitcoin, the first decentralized cryptocurrency, achieves great success but also encounters many challenges. In this paper, we mainly focus on Bitcoin's five challenges: low network synchronization; poor throughput; high information propagation delay; vulnerabilities to fork-based attacks and consumption of a large amount of computational power to maintain the blockchain. To address these challenges, we present the CP-consensus, a blockchain protocol based on synchronous timestamps of the Compass satellite. Firstly, CP-consensus provides a quasi-synchronous network for nodes. Specifically, nodes synchronously begin or end in each phase. Secondly, the block propagation delay is significantly reduced by adopting cache-nodes. Moreover, the block verification delay is significantly reduced since it is limited only by the size of block-header. Thirdly, CP-consensus has a high throughput with a larger block size since that the block size does not influence the consistency of CP-consensus. Fourthly, CP-consensus resists fork-based attacks and consumes a small amount of computational power. Finally, parameters setting and the security of CP-consensus are discussed.**

***Index Terms*—Blockchain, consensus, throughput, fork, synchronous timestamps, Compass satellite.**

I. Introduction

BITCOIN, proposed in 2009 by Satoshi Nakamoto [1], is the first decentralized cryptocurrency which maintains a public transaction ledger, called blockchain, in a distributed manner without the central authority. The core technological innovation of Bitcoin is Nakamoto consensus which provides a high-probability guarantee that an adversary cannot alter a transaction once this transaction is sufficiently deep in the blockchain, assuming honest nodes control the majority of computational resources in the system. The Nakamoto blockchain works in a permissionless model, where any node can freely join and leave the protocol, and there is no a-priori knowledge of the set of consensus nodes. At the time of writing of this paper (August 2017), Bitcoin held a market price of 4,283 USD and a market capitalization of 68.8 billion USD [2]. Alternative cryptocurrencies called altercoins (e.g., Litecoin [3], Ripple [4] and Ethereum [5]) have achieved enormous success. Several consensuses to manage blockchain-based ledgers have been proposed: proof-of-work [6], proof-of-stake [7], [8], proof-of-space [9], proof-of-activity [10], proof-of-human-work [11], practical Byzatine fault-tolerance [12], or some combinations [13], [14], [15]. Especially, most existing cryptocurrencies, including Bitcoin, adopt proof-of-work. Despite Bitcoin's success, it suffers several technical problems.

- **Technical problem 1: Vast consumption of computational power.** In July 2017, the hash rate of the Bitcoin network is about 7×10^6 TH/s [2]. Roughly speaking, if all miners use the AntMiner-S9 [16], the whole Bitcoin network will consume about 6×10^9 kwh of electricity per year.
- **Technical problem 2: Concentration of computational power.** Due to the huge difficulty of block generation, it is almost impossible to generate a block with an ordinary computer. Thus, miners form the mining pool, managed by the pool controller, where all members work concurrently and get reward according to their amount of work. In July 2017, biggest four mining pools control more than 50% computational power [2]. Consequently, the Bitcoin's assumption that a majority of the computational power is honest becomes less credible.
- **Technical problem 3: Long propagation delay.** Blocks and transactions propagation delay in the Bitcoin network is measured by Decker et al. in [18] via establishing connections with each node. Furthermore, this problem is further studied by Pappalardo et al. in [17]. In particular, they show that 43% of transactions are not recorded in the blockchain after one hour from the first time they were seen in the blockchain and 20% of transactions are still unrecorded after 30 days. Moreover, blocks first announced by some nodes are disseminated consistently faster (or slower) than others. Additionally, blocks first propagated by the fastest nodes reach 50% of nodes in 2.3 seconds whereas blocks propagated by the slowest nodes reach 50% of nodes in more than 1,800 seconds.
- **Technical problem 4: Forks.** At P2P2013, Decker et al. study the information propagation delay in the Bitcoin network in [18] and show that the propagation delay in the network is the primary cause for blockchain forks. They also claim that Blockchain forks not only causes a prolonged inconsistency, but also weakens the systems defenses against attackers. By observing 1,000 blocks, they discover that there are 169 blockchain forks, resulting in an observed blockchain fork rate 1.69%. At CCS2016, Gervais et al. further investigate performance of Bitcoin blockchian in [19]. They observe that forks

L. Zhou was with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Bei Jing 100876, P.R. China. E-mail: (379739494@qq.com).

L. Wang was with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Bei Jing 100876, P.R. China. E-mail: (wanglc2012@126.com).

Y. Sun was with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Bei Jing 100876, P.R. China.

Manuscript received April 19, 2005; revised August 26, 2015.

rate is 1.85% and that block propagation mechanism obviously influences the security of blockchian and the fork rate.

- **Technical problem 5: Attacks.** Blockchain is vulnerable to fork-based attacks as Selfish mining, Double spending and Whale attacks. A Selfish mining [20] attacker with significant mining resources would cause two negative effects. Firstly, it increases transaction approval times. Secondly, it makes Bitcoin more vulnerable to Double spending [21]. A successful Selfish mining attacker becomes more profitable than honest nodes, and can grow steadily. Therefore, it thus eventually drives other nodes out of system. Profits from Selfish mining increase as more computational power is held by the attacker making its attack increasingly effective until it eventually holds over 50% of the computational resources in the network. In this case, the attacker is able to collect all block rewards and arbitrarily modify the transaction history at any time. Liao et al. [24] propose a new attack, Whale attack, in which an attacker issues an off-blockchain whale transaction with an anomalously large transaction fee in order to convince miners to fork the current main blockchain.
- **Technical problem 6: Low throughput.** The throughput of Bitcoin is investigated in many studies [17], [18], [19], [30], [31]. Currently, the Bitcoin network can handle at most 7 transactions per second [2], whereas Visa handles on average from 2,000 to 7,000 [27]. This reveals a great inefficiency in the Bitcoin system.
- **Technical problem 7: Instability of blocks generation.** Pappalardo et al. [17] show that the minimum time to mine a valid block is about 2 minutes, while the maximum time is about 77minutes; the medium time for discovering a block is about 9 minutes and the 50% percentile is about 6 minutes.

In this paper, we present CP-consensus, a blockchain protocol based on synchronous timestamps (CP-timestamps) of Compass satellite. By utilizing CP-timestamps, time is divided into equi-long epochs and an epoch is divided into three phases—recording-phase, propagating-phase and mining-phase. Furthermore, CP-consensus has four kinds of nodes—CP-node, record-node, cache-node and trade-node. Properties of CP-consensus are as follows.

- **Optimizing the blockchain network topology and achieving the low block propagation delay.** In CP-consensus, block-body generation and PoW computation are separated. Specifically, cache-nodes store block-bodies before these block-bodies are used to compute the proof-of-work (PoW). Therefore, if some record-node finds a solution of PoW, it just needs to propagate its block-header to others. Meanwhile, because the block-header is very light, so the block-header can be propagated to a majority of record-nodes in a very short time. Therefore, the block creator and cache-nodes can almost synchronously broadcast the block. In this way, we optimize the blockchain network topology of propagating blocks and obviously reduce the block propagation delay.
- **Low block verification delay.** In most blockchains, the block verification must be performed after receiving the block. Because the block propagation delay is non-negligible and the block verification delay is almost equal to the block propagation delay, so the block verification delay is obvious. However, in CP-consensus, the block verification can be performed before receiving the block. Therefore, the block verification delay is limited only by the size of block-header. Because the size of block-header is about 1KB, it can be propagated to a majority of 6000 nodes in about 0.0163s. Therefore, the block verification delay of CP-consensus is very low.
- **Quasi-synchronous network.** The CP-node (Compass satellite) periodically transfers CP-timestamps to others on the ground and others can receive CP-timestamps almost synchronously with time receivers. By using CP-timestamps to strictly restrict behaviors of players, players can synchronously begin or end in each phase.
- **High throughput.** In CP-consensus, because the block verification delay is limited only by the size of block-header and network bandwidth, so CP-consensus can have an obviously higher throughput than Bitcoin via a larger block size. Furthermore, in recording-phase, a record-node does nothing but records transactions. In other words, it can use all its own computational power to process transactions in recording-phase. Therefore, the rate of processing transactions will be further increased.
- **Resistance to fork-based attacks.** In fork-based attacks (e.g., Selfish mining [20], Double spending [21], Whale transaction [24] attacks), attackers generate (or induce others to generate) forks to break the consistence of blockchain in order to obtain more benefits. To address this problem, CP-consensus requires that the block-body must be published before its creator begins to compute the PoW. Furthermore, by utilizing a random selection mechanism, only a random fraction of record-nodes can participate in the mining-phase to compute the PoW. Consequently, attackers cannot accomplish a fork-based attack with an overwhelming probability.
- **Small amount of computational power.** Essentially, computing PoW occurs only in the mining-phase, which is a fraction of the epoch. Moreover, no matter how many users join in the system, only average k miners, which can be maintained by periodically adjusting some difficulty value, can participate in each mining-phase. Besides, by setting the range of nonce, all miners have the same quota of hashing queries per mining-pase. Therefore, the upper bound of the whole system hash rate and the amount of consumed computational power can be calculated in advance.
- **Fixed block interval.** In CP-consensus, the block interval is fixed by CP-timestamps. For instance, BD-consensus can strictly produce one block per 10 minutes.

Organization. In Sec.II, related work is introduced. In Sec.III, we show Bitcoin and Bitcoin's protocol. In Sec.IV, we introduce the model of CP-consensus. In Sec.V, an overview of CP-consensus is given. In Sec.VI, we propose details of

executing CP-consensus. In Sec.VII, we analyze the security of CP-consensus. Finally, a short conclusion is given in Sect. VIII.

II. Related work

In this section, we introduce related work to our protocol as follows

- **Time synchronization.** Tang et al. [36] investigate the time synchronization between stations based on the Compass satellite common-view. They show that, by using single IGSO Compass satellite, the precision of time synchronization between stations is 10 ns, and the precision is 3 ns by using single GEO Compass satellite common-view. Therefore, the time synchronization between stations with the Compass satellite can get high precision and stability. In our protocol, we use this method to execute the time synchronization. Specifically, the CP-node, which is the Compass satellite, periodically transfers synchronous timestamps (CP-timestamps) to other nodes on the ground. By using time receivers, other nodes can almost synchronously receive CP-timestamps.
- **Synchronous network.** At CCS2016, Gervais et al. [19] propose that Double spending and Selfish mining can be alleviated if all nodes in the blockchain system are tightly synchronized. Furthermore, at ECRYPT2015, Garay et al. [28] demonstrate that the Bitcoin blockchain protocol satisfies the consistency in a full synchronous network. In their analysis model [28], they assume that all parties in the network are able to synchronize in the course of a round; all parties have the same quota of hashing queries per round; the adversary controls less than 50% of the total computational power. In our protocol, by using CP-timestamps, players can synchronously begin or end in each phase; by significantly reducing the block verification delay, the network synchronization is further improved; by setting the range of nonce, all players have the same quota of hashing queries per round; we also assume that the adversary controls less than 50% of the total computational power. Therefore, our protocol is similar to the model in [28] and their results support our protocol to some extent.
- **Propagation delay.** Many studies [17], [18], [19], [30], [29], [31], [33] research the information propagation delay. Miller et al. [29] and Sallal et al. [30] show that the number of nodes and the network topology have a great impact on the propagation delay. Furthermore, Sallal et al. suggest that faster information propagation could be achieved by applying the clustering theory. In our protocol, cache-nodes and the block creator can almost synchronously broadcast a valid block. Therefore, the network topology is optimized and the block propagation delay is significantly reduced.
- **Block interval is divided into phases.** At NSDI2016, Eyal et al. propose Bitcoin-NG [32], in which time is divided into epochs and each epoch is divided into mining-phase and recording-phase. Specifically, in mining-phase, miners produce a key-block to select a leader via solving a cryptopuzzle. Then the leader generates several micro-blocks to record transactions. In our protocol, by utilizing CP-timestamps, time is divided into equi-long epochs and each epoch is divided into recording-phase, propagating-phase, and mining-phase.
- **Unforgeable timestamp.** At FC2014, Eyal et al. [20] change the Bitcoin protocol to raise the threshold of the minimum successful mining resource share to 25%. While, at FC2014, Heilman E. [33] raise this threshold from 25% to 32% under the assumption that they have unforgeable timestamps. In our protocol, CP-timestamps are unforgeable and we use them to resist fork-based attacks including Selfish mining attack.
- **Leader selection.** In each of epoch of Bitcoin-NG [32], nodes use the key block to select a leader. Besides, Micali et al. [34] propose Algorand, a blockchain protocol based on message-passing Byzantine agreement, which uses secret cryptographic sortition and secret credentials to secretly and randomly select the leaders (a subset of verifiers) in charge of generating the new block. In our protocol, we propose a new method via hash function to randomly and periodically select leaders.
- **Zeroblock.** Solat et al. [35] use the expected time and the dummy block, called Zeroblock, to prevent block withholding and Selfish mining. Specifically, the key contribution of their solution is that an honest node must generate or receive a block within an expected time calculated by honest nodes. Otherwise, it generates a Zeroblock. In our protocol, epochs (block intervals) are equi-long. Furthermore, if a block is generated in the n-th epoch, this block can be received by others only in the n-th epoch. Otherwise, others will reject this block. Moreover, when there is no node to generate a block within an epoch, we follow the Zeroblock that all honest nodes generate a Zeroblock.

III. Bitcoin and Bitcoin's blockchain

Bitcoin [1] is a decentralized payment scheme in which every participant maintains its own local copy of the whole transaction history, chain of blocks called blockchain. Blockchain is maintained by anonymous participants called miners via executing a consensus scheme that extends the blockchain. Bitcoin consistency relies on the idea of computational puzzles—a.k.a. moderately proof-of-work put forth by Dwork and Naor [37]. In Bitcoin, payers broadcast transactions and miners collect transactions into their local blocks. A block contains two parts: block-body and block-header. Specifically, the block-body contains the transactions. The block-header contains the hash value of previous block, the current Unix time, target value, a nonce and a merkle root of transactions. In Bitcoin consensus, for a block to be valid, the cryptographic hash of its header must be smaller than a target value. If some miner finds a solution of the cryptographic puzzle, then he immediately broadcasted his block including the solution to others. After that, upon verifying the block, others will receive and add this block as a new one in its local blockchain and then continue the mining process on its updated blockchain. The creator of

the block is rewarded with bitcoins (coins in Bitcoin system) via the coinbase transaction which is the first transaction in the block-body. Consequently, bitcoins are created and distributed among miners. Moreover, this creator is also rewarded by transactions fees for all transactions included in the block.

Bitcoin's Assumption: Bitcoin assumes that a majority of computational power is controlled by honest players.

A. FORKS

If multiple miners create blocks with the same preceding block, the blockchain is forked into forks or branches. Other miners subsequently and randomly mine after them. In the Bitcoin protocol, this problem is resolved by the heaviest chain rule—the heaviest blockchain containing the most mining power is valid. All miners add blocks to the heaviest chain of which they know. The lighter fork will be ignored and called the orphan chain. The block included in the orphan chain is called the orphan block.

B. FORK-BASED ATTACKS

Currently, PoW-based blockchains are still vulnerable to various fork-based attacks (e.g., 51% attack, Double spending, Selfish mining and Whale attack).

51% attack. An attacker with more than 50% of the total computational power can control the whole system. Specifically, at any time and any position, this attacker can generate a longer fork to replace the original chain. Therefore, he can obtain all block rewards and arbitrarily modify the transaction history.

Double spending attack. A Double spending attacker creates two transactions T_A and T_B with the same coin. He sends T_A to a merchant and at the same time he generates a secrete fork including T_B. After required block confirmations (e.g., 6 block confirmations in Bitcoin), if the attacker receives goods from the merchant and his secrete fork is longer than the main chain, then he publishes the secrete fork. Therefore, others will accept the attacker's fork and T_B will become the valid one. Consequently, the attacker spends the same coin twice.

Selfish mining attack. Eyal and Sirer [20] proposed a strategy, called Selfish mining, and claimed that an alliance of dishonest miners with a proportion $\alpha < 1/2$ of the total computational power can achieve a proportion more than α of the mining revenue. Furthermore, at FC2016, Sapirshtein et al. [38] extend the Selfish mining attack and propose optimal strategies for Selfish mining attacks. The Selfish mining attack relies on block concealing and revealing only at some special time selected by attackers, called Selfish miners, with a non-shorter fork in order to achieve more revenue. By using Selfish mining, Selfish miners can achieve revenues larger than their fair share. Furthermore, Selfish mining attack can have serious consequences for PoW-based blockchain: rational miners will prefer to join the Selfish miners; the colluding group will increase its size until it becomes a majority, and if it happens, the system will thoroughly lose the decentralization; it will increase the risk of suffering the Double spending; the rate of producing blocks will reduce; the ratio of orphan blocks will increase; a vast of computational power is wasted due to the production of orphan blocks; it significantly impact the consistency of blockchain.

Whale attack. Recently, Kevin et al. [24] provided an attack strategy, called Whale attack, in which an attacker issues whale transactions with an anomalously large transaction fee in an effort to convince miners to fork the current chain.

IV. MODEL OF CP-CONSENSUS

CP-consensus is comprised of a CP-node, which is the Compass satellite, and a set of general-nodes connected by a reliable peer-to-peer network. Each general-node can poll a random oracle [25] as a random bit source. Besides, there is no trusted public key infrastructure and each general-nodes can generate key-pairs by themselves. Moreover, by mortgaging a certain amount of coins, a general-node can become a record-node or cache-node.

Digital signature and hash function. We assume that digital signature $\mathsf{Sig}(\cdot)$ and hash function $\mathsf{H}(\cdot)$ used in CP-consensus are ideal such that no one can violate $\mathsf{Sig}(\cdot)$ and $\mathsf{H}(\cdot)$. CP-consensus, similar to Bitcoin, employs a cryptopuzzle system, defined by $\mathsf{H}(\cdot)$. The solution to this puzzle defined by the string m is a string x such that $\mathsf{H}(m|x)$ is smaller than some target value. Each node has a limited a mount of computational power measured by the number of computing hash function per second. A solution to the puzzle is called a proof-of-work (PoW) since it indicates the amount of work a node had to perform to find the solution.

Honest and malicious. A node is honest if it follows all protocol instructions and is perfectly capable of sending and receiving information. Furthermore, a node is malicious if it can deviate arbitrarily from protocol instructions. At any time t, a subset of cache-nodes (record-nodes) are malicious and controlled by a single adversary. Other nodes are honest. Moreover, we assume that more than 50% of cache-nodes are honest and more than 50% of computational power is controlled by honest record-nodes.

V. AN OVERVIEW OF CP-CONSENSUS

By mortgaging a certain amount of coins for a certain number of blocks, some general-nodes become record-nodes and some other general-nodes become cache-nodes. CP-node, which is the Compass satellite [36], periodically broadcasts synchronous timestamps (CP-timestamps) to cache-nodes and record-nodes. Furthermore, cache-nodes and record-nodes utilize time-receivers to synchronously receive CP-timestamps. Moreover, according to CP-timestamps, time is divided into equi-long epochs and each epoch is divided into recording-phase, propagating-phase and mining-phase. A overview of CP-consensus is described in Fig.1 and working processes of three phases are shown in Fig.2.

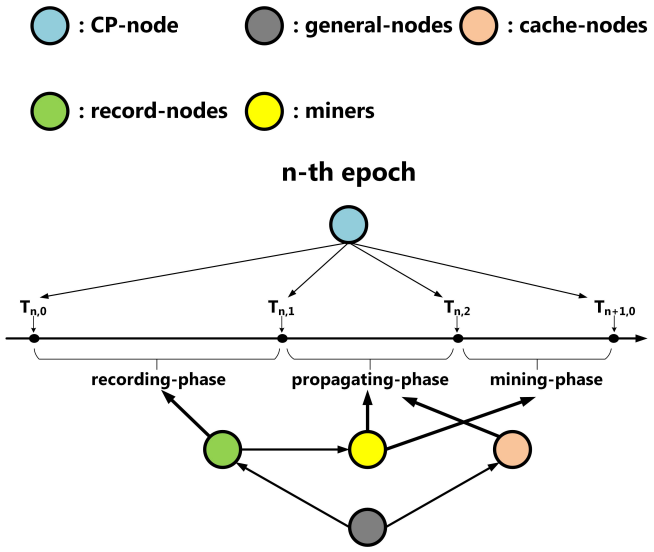

Fig. 1: An overview of CP-consensus: the n-th epoch is divided into recording-phase, propagating-phase and mining-phase by CP-timestamps, which are broadcasted by the CP-node; by mortgaging a certain amount of coins for a number of blocks, some general-nodes become record-nodes and some other general-nodes become cache-nodes; by performing the selection scheme, a subset of record-nodes become miners; record-nodes can only participate in recording-phase; miners should participate in propagating-phase and mining-phase; cache-nodes can participate in propagating-phase.

In recording-phase. Record-nodes record transactions inside their local blocks.

In propagating-phase. All record-nodes perform the selection mechanism to select a subset of record-nodes to become *miners* who can participate in the mining-phase and these miners immediately propagate their block-bodies to all cache-nodes. Consequently, cache-nodes store and open these block-bodies. At the end of propagating-phase, each of cache-node broadcast a *list* of hash values of all block-bodies received in this propagating-phase and a signature of this *list* to all record-nodes. This signature is used to promise that the cache-node has received these block-bodies and will not add, delete or modify any block-body.

In mining-phase. Miners compute the PoW in this phase. When some miner finds a solution of the PoW, he immediately broadcasts the block-header (not the whole block) including the solution to other record-nodes and then other record-nodes, upon verifying the block-header, receive the corresponding block-body from some cache-node.

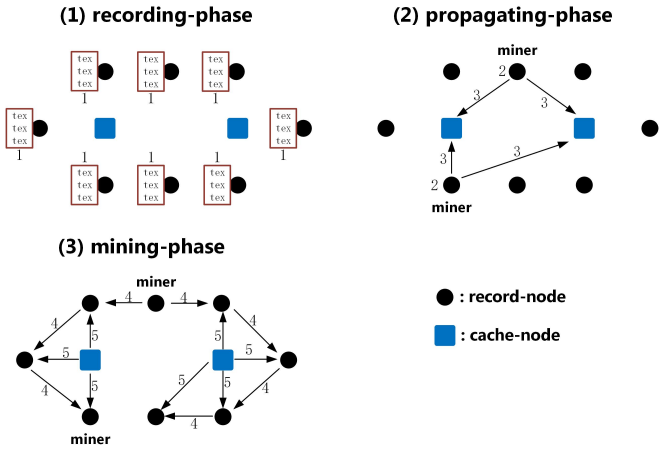

Fig. 2: Working process of recording-phase, propagating-phase and mining-phase. Step1: record-nodes record transactions in their block-bodies. Step2: By using selection mechanism, some record-nodes become miners. Step3: Miners propagate their block-bodies to cache-nodes. Step4: Some miner finds a solution of PoW and broadcast its block-header. Step5: Cache-nodes broadcast the block-body corresponding to the block-header.

VI. Achieving CP-consensus

A. RECORD-NODES AND CACHE-NODES

To become a record-node or cache-node, a general-node should mortgage a certain amount of coins as the guarantee deposit for a certain amount of blocks.

Specifically, to become a record-node, a general-node should generate a R-transaction to mortgage Q_R coins for $NR_1 + NR_2$ blocks. When the R-transaction is included in the blockchain and $NR_1 - 1$ blocks are produced after the block containing the R-transaction, this general-node become a record-node in the following NR_2 blocks and his address is published in the R-transaction. After these $NR_1 + NR_2$ blocks, the guarantee deposit will be returned to the record-node and the record-node will turn back into a general-node.

Similarly, to become a cache-node, a general-node should generate a C-transaction to mortgage Q_C coins for $NC_1 + NC_2$ blocks. When the C-transaction is included in the blockchain and $NC_1 - 1$ blocks are produced after the block containing the C-transaction, this general-node become a cache-node in the following NC_2 blocks and his address is published in the C-transaction. After these NC_1+NC_2 blocks, the guarantee deposit will be returned to the cache-node and the cache-node will turn back into a general-node.

B. BLOCK VERIFICATION

In this subsection, we redefine the validation of block-header and introduce the block verification.

At the end of propagating-phase, an honest cache-node should have verified validations of all block-bodies received in this phase and only stores valid block-bodies. Then, the cache-node generates and broadcasts a *list*, which is comprised by hash values of stored block-bodies, and a signature of the *list*.

According to the model of CP-consensus, we assume that a majority of cache-nodes are honest. Therefore, a hash value included in a majority of *lists* means that the corresponding block-body has been stored and verified by a majority of cache-nodes. Consequently, the corresponding block-body is credible. In this paper, we redefine the validation of block-header as follow.

Definition 1: A block-header is valid if the block-header satisfies Eq.(2) and the hash value, which is contained in this block-header, of the corresponding block-body is included in a majority of *lists*.

When a record-node receives a new block-header in the mining-phase, he should verify the block-header validation. If the block-header passes the verification, the record-node will consider that the corresponding block-body is valid and credible. After that, he will download the corresponding block-body from some cache-node or record-node. In this way, record-nodes can verify the block validation before receiving the whole block. While, in Bitcoin-like blockchain, the block verification must be performed after receiving the whole block. Therefore, the block verification delay of CP-consensus is very low and limited only by the size of block-header.

In Bitcoin, the median block propagation delay is about 8.7s and the average size of blocks is 534.8KB. While in CP-consensus, if the propagation rate and the size of network are same as Bitcoin, then the median block-header propagation delay is about $\frac{8.7}{534.8}$ ≈0.0163s since the size of block-header is 1KB. Therefore, the median block verification delay of CP-consensus is about 0.0163s.

In a special case, a block-header passes the verification and record-nodes download the corresponding block-body from cache-nodes, but the block-body is invalid for some reasons (e.g., some included transaction is invalid). Honest record-nodes will discard this block and then verify the other block-header, which is the newest one after the previous block-header, until get a valid block. If this wrong action happens for a cache-node, this cache-node will be severely punished. *Incentive and punishment for cache-nodes* will be introduced in Sec.VI-E.

C. RECORDING-PHASE, PROPAGATING-PHASE AND MINING-PHASE

The CP-node periodically generates and broadcasts CP-timestamps $T_{n,0}$, $T_{n,1}$, $T_{n,2}$, $T_{n+1,0}$, $T_{n+1,1}$, $T_{n+1,2}\cdots$ to cache-nodes and record-nodes. In particular, the n-th epoch is from $T_{n,0}$ to $T_{n+1,0}$; the recording-phase is from $T_{n,0}$ to $T_{n,1}$; the propagating-phase is from $T_{n,1}$ to $T_{n,2}$; the mining-phase is from $T_{n,2}$ to $T_{n+1,0}$. According to [36], the precision of time synchronization between stations on the ground is 10 ns by using single IGSO Compass satellite. Therefore, cache-nodes and record-nodes can use time receivers to receive CP-timestamps almost synchronously. Moreover, a block contains *block-header* and *block-body* as described in Fig.3. Furthermore, in CP-consensus, block-body and block-header are redefined as follows.

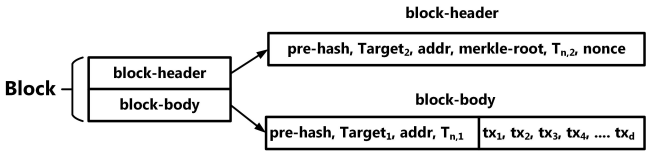

Fig. 3: The structure of block.

Block-header={*pre-hash, Target*$_2$*, addr, merkle-root,* $T_{n,2}$, *nonce*}

Block-body= {*pre-hash, Target*$_1$*, addr,* $T_{n,1}$, *Tx*}

In above definitions, *Tx* denotes all transactions included in this block-body. *pre-hash* denotes the hash value of the previous block. *merkle-root* denotes the hash value of *Tx*. *addr* denotes a record-node's address. $Target_1$ denotes the target value used in Eq.(1) to select miners. $Target_2$ denotes the target value used in Eq.(2) to select the creator of a block. *nonce* denotes a random number. Miners vary the *nonce* to solve the Eq.(2).

We take the n-th epoch for example. Cache-nodes and record-nodes work as follows.

- In the recording-phase, from $T_{n,0}$ to $T_{n,1}$, record-nodes record transactions in their local block-body.
- In the propagating-phase, from $T_{n,1}$ to $T_{n,2}$, at the time of receiving $T_{n,1}$, all record-nodes compute the hash value with the hash function $\mathsf{H}(\cdot)$ as

$$\mathsf{H}(\textit{pre-hash}||Target_1||addr||T_{n,1}) < Target_1. \quad (1)$$

  A part of record-nodes become *miners* if their *addr*s satisfy Eq.(1). Consequently, miners immediately propagate their block-bodies to all cache-nodes. Furthermore, the validation of block-body means that all included transactions, *pre-hash*, *Target*$_1$, *addr* and $T_{n,1}$ are correct. Honest cache-nodes should verify all received block-bodies and only store valid block-bodies. Additionally, that a cache-node publishes a block-body means that he has verified this block-body and the block-body is valid. Otherwise, the cache-node rejects this block-body. At the end of propagating-phase, an honest cache-node should immediately generate and broadcast a *list* of hash values of block-bodies received in this propagating-phase and a *signature* σ as follows.

$$\mathsf{H}(\textit{block-body}) = \mathsf{H}(\textit{pre-hash}||\textit{Target}_1||\textit{addr}||T_{n,1}),$$

$$list = \{\mathsf{H}(block-body_1)||...||\mathsf{H}(block-body_d)\},$$

$$\sigma = \mathsf{Sig}_{SK}\{T_{n,1}||list\}.$$

  List and σ are evidences to prevent cache-node's malicious addition, deletion and modification.
- In the mining-phase, from $T_{n,2}$ to $T_{n+1,0}$, miners solve the hash-puzzle by varying the *nonce* as follow

$$\mathsf{H}(\mathsf{H}(block-body)||Target_2||merkle-root|| \quad (2)$$

$$T_{n,2}||nonce) < Target_2.$$

When some miner finds a solution of PoW, the finder immediately broadcasts his block-header including the solution to other nodes. After receiving a block-header, other nodes will verify the block-header's validation. If the block-header is valid, cache-nodes and the finder will almost synchronously broadcast the corresponding block-body. Otherwise, others will reject this block-header until a valid block-header appears. Besides, if no one propagates a valid block-header in the mining-phase, all record-nodes will generate a Zeroblock, a block in which the block-body contains no transactions.

- At the time of receiving $T_{n+1,0}$, all nodes join in the $n+1$-th epoch.

D. MAJORITY-RULE

In this subsection, we introduce how to resolve forks in CP-consensus, though it occurs with a very low probability. The method is called as *majority-rule*.

Majority-rule: A block-body is valid if it is stored in a majority of record-nodes.

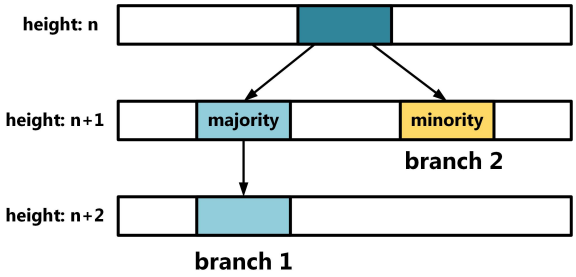

Fig. 4: The block validation. A block-body stored in a majority of cache-nodes is valid and later block should follows this block-body. While a block-body stored in a minority of cache-nodes will be ignored and called the orphan block. As described in above figure, *branch 1* becomes a part of the main chain and *branch 2* is ignored.

When a fork occurs in the $n+1$-th epoch, the blockchain may generates branches as Fig.4. Upon the majority-rule, later block should follows the block including this block-body. Moreover, while a block-body stored in a minority of record-nodes will be ignored and the corresponding block called the orphan block. As mentioned in Fig.4, *branch 1* becomes a part of the main chain. However, *branch 2* is ignored and called the *orphan chain*.

For encouraging record-nodes to follow the protocol, CP-consensus imitates Bitcoin that the block finder gets a mount of coin via a special transaction inserted in the block.

E. INCENTIVE AND PUNISHMENT FOR CACHE-NODES

For encouraging honest cache-nodes, CP-consensus rewards a cache-node according to the number of block-bodies stored in the cache-node. While each of record-node maintains a reputation system for supervising cache-nodes' wrong actions. For example, a cache-node broadcasts the *list* of block-bodies hash values and signature of the *list* too late; some block-body stored by a cache-node contains some invalid transactions; some other data of a block-body is invalid. If the reputation score of a cache-node is subtracted as 0, then record-nodes will not propagate any block-body to this cache-node. Generally, if a cache-node publishes a wrong block-body, then its *list* will be propagated to most record-nodes and these record-nodes will punish the cache-node. In particular, we suggest that if a cache-node publish an invalid block-body, then all record-nodes subtracts at least a half reputation score.

F. $Target_1$ ADJUSTMENT

In CP-consensus, $Target_1$ is used to select a subset of record-nodes to become miners. Generally, the number of record-nodes is unfixed. However, we hope that the average number of miners in each epoch is stable. To achieve this, all record-nodes periodically adjust the $Target_1$ per M blocks.

For instance, the number of record-nodes is K and the maximum value of hash function $\mathsf{H}(\cdot)$ is V. For achieving the average number of miners is k per epoch, we have $Target_1 = v$ and

$$v = \frac{k}{K} \cdot V.$$

Consequently, an arbitrary string is a solution of Eq.(1) with the probability $\frac{k}{K}$. In this way, by using Eq.(1) to select miners, the average number of miners will be k per epoch. As a result, a honest cache-node store k block-bodies, on average, per epoch.

While, after M blocks, as the number of record-nodes changes, the number of record-nodes will be K'. Because the $Target_1$ is v still, so the average number of miners will be $k' = \frac{v}{V} \cdot K'$ per epoch. To control the average number of miners per epoch to come back to k, all record-nodes need to adjust target$_1 = v'$ and v'= $v \cdot \frac{k}{k'}$. In this way, the average number of miners will be k per epoch still. The correctness is proved as follow.

$$\frac{k'}{K'} = \frac{v}{V} \Leftrightarrow \frac{k \cdot \frac{k'}{k}}{K'} = \frac{v}{V} \Leftrightarrow \frac{k}{K'} = \frac{v \cdot \frac{k}{k'}}{V}$$

$$\Leftrightarrow \frac{k}{K'} = \frac{v'}{V} \Leftrightarrow v' = v\frac{k}{k'}$$

G. PARAMETERS SETTING

We assume that the time of an epoch is T seconds; the time of mining-phase is t seconds; the maximum value of hash function $\mathsf{H}(\cdot)$ is V; the $Target_2$ is v_2; the range of *nonce* is r; the upper bound of hash rate of the whole system is $Bound_{hash}$; the average number of miners in the mining-phase is k. Essentially, v_2, r and k should be set to satisfy that, in a mining-phase, the probability of finding blocks should be close to 100% and the $Bound_{hash}$ should be acceptable. For a mining-phase, related probabilities are calculated as follows.

- The probability of a random string being the solution of Eq.(2) is calculated as

$$p = \frac{v_2}{V}. \tag{3}$$

- The probability of a miner finding no block in a mining-phase is calculated as

$$P_1 = (1-p)^r. \tag{4}$$

- The probability of k miners finding no block in a mining-phase is calculated as

$$P_2 = P_1^k. \tag{5}$$

- The probability of k miners finding at least one blocks in a mining-phase is calculated as

$$P_3 = 1 - P_2. \tag{6}$$

The upper bound of hash rate of the whole system can be calculated as

$$Bound_{hash} = \frac{r \cdot k}{t}. \tag{7}$$

H. INSTANCE

We assume that $T = 600$ seconds, $t = 240$ seconds, $k = 15$, $V = 2^{256}$, $v_2 = p \times 2^{256}$, $r = 10^s$ and $p = 10^{-d}$. For example, when $r = 10^{13}$, then $Bound_{hash} = 6.25 \times 10^{11}$ hash/s. Comparing the Bitcoin's hash rate $7{\times}10^{18}$ hash/s, our protocol is very electricity-saving. Furthermore, results of P_3 are described in Table I and Fig.5. Moreover, results of $Bound_{hash}$ are described in Table II.

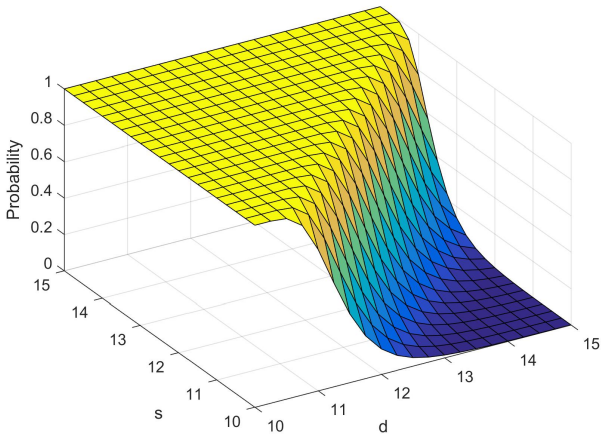

Fig. 5: The probability plot of finding blocks. k=15, $V = 2^{256}$, $v_2 = 10^{-d} \times 2^{256}$ and $r = 10^s$. s and d change from 10 to 15. P_3 can be calculated with Eq.(3), Eq.(4), Eq.(5) and Eq.(6).

TABLE I: The probabilities of 15 miners finding at least one blocks

P_3 r \ p	10^{-10}	10^{-11}	10^{-12}	10^{-13}	10^{-14}	10^{-15}
10^{10}	0.9999996	0.7768698	0.1392891	0.0148926	0.0014976	0.0001498
10^{11}	0.9999999	0.9999996	0.7768624	0.1393321	0.0148762	0.0014976
10^{12}	0.9999999	0.9999999	0.9999996	0.7769738	0.1391888	0.0148762
10^{13}	0.9999999	0.9999999	0.9999999	0.9999996	0.7766021	0.1391888
10^{14}	0.9999999	0.9999999	0.9999996	0.9999999	0.9999996	0.7766021
10^{15}	0.9999999	0.9999999	0.9999999	0.9999996	0.9999999	0.9999996

k=15, $V = 2^{256}$, $v_2 = 10^{-d} \times 2^{256}$ and $r = 10^s$. s and d change from 10 to 15. P_3 can be calculated with Eq.(3), Eq.(4), Eq.(5), Eq.(6).

TABLE II: The upper bounds of *Hash-rate*

r	10^{10}	10^{11}	10^{12}
$Bound_{hash}$	6.25×10^8/s	6.25×10^9/s	6.25×10^{10}/s
r	10^{13}	10^{14}	10^{15}
$Bound_{hash}$	6.25×10^{11}/s	6.25×10^{12}/s	6.25×10^{13}/s

$r = 10^s$, t=240 seconds and s changes from 10 to 15. The $Bound_{hash}$ can be calculated with Eq.(7).

VII. SECURITY ANALYSIS

In this section, we analyze the security of CP-consensus.

A. BLOCK PROPAGATION DELAY

In this subsection, we analyze the block propagation delay of CP-consensus. According to [19], in Bitcoin, the block interval is 600s; the number of public nodes is 6000; the median block propagation time is 8.7s; the average block size is 534.8KB.

Network size and propagation rate. For comparing with Bitcoin, we assume that, in CP-consensus, the number of record-nodes is 6000 and the number of cache-nodes is 63. Furthermore, the information propagation rate in CP-consensus is the same as Bitcoin.

Propagation model. We assume that if a node does not complete the block propagation from it to one of its neighbors, then it will not propagate the block to the other neighbor. In other words, the propagation method of a node is "one-by-one". Furthermore, when a node receives a block, it will immediately relay the block to its neighbors which have not received the block with the method "one-by-one" still. Besides, we also assume that the block size propagated in the network is constant S KB and the time of propagating the block, from a node to its neighbor node, is constant t seconds. Therefore, if a block has been propagated l times, then the number of nodes receiving the block is

$$N(l) = n \cdot 2^l,$$

where n is the number of nodes synchronously propagating the block at the beginning of the propagation.

Propagation delay. For Bitcoin, if a block is found, the finder is the single broadcaster at the beginning of propagation. Upon the above propagation model, if a block is propagated d times, then the number of nodes receiving the block is

$$N_B(d) = 2^d. \tag{8}$$

Therefore, the number of times of propagating a block to 6000 nodes can be calculated as follow.

$$d_B = log_2 6000 = 12.55.$$

According to [19], the median block propagation time of Bitcoin is 8.7s. It means that a block is propagated to 3000 nodes with about 8.7s. Moreover, according to Eq.(8), propagating a block to 3000 nodes needs about $d_B - 1 = 11.55$ times of propagation. Therefore, each propagation between two connected nodes costs $\frac{8.7}{d_B - 1} \approx 0.7532s$ on average. Furthermore, the propagation rate between two connected nodes is about

$$\frac{0.7532s}{534.8KB} = 0.00141s/KB.$$

For CP-consensus, the propagation rate between two connected nodes is also 0.00141s/KB; the size of block-header is 1 KB; the number of times of propagating a block-header to all record-nodes is 12.55. Therefore, a block-header is propagated to 3000 record-nodes with $0.00141s \times 11.55 \approx 0.0163s$ and a block-header is propagated to 6000 record-nodes with $0.00141s \times 12.55 \approx 0.0177s$. Due to designs of CP-consensus as mentioned in Sec.VI-E, a block is valid if its block-header passes the verification. Therefore, the block verification delay, in CP-consensus, is limited only by the size of block-header. Consequently, the longest block-header propagation delay in CP-consensus is 0.0177s, which is much effective than Bitcoin.

After checking the validation of a block-header, the corresponding block-body will be broadcasted by the finder and all cache-nodes. Therefore, the number of times of propagating a block to 6000 record-nodes is

$$64 \times 2^{l_{CP}} = 6000 \Rightarrow l_{CP} \approx 6.55.$$

Now we calculate the time of propagating a block, in CP-consensus, to 6000 record-nodes. If the average size of block is 10 M=10240 KB, then the time of propagating a block to 6000 record-nodes can be calculated as follow.

$$10240KB \times 0.00141s/KB \times 6.55 = 94.57s.$$

Propagating a valid block should be done within mining-phase or recording-phase. Therefore, to ensure that the time of propagating the block is enough, the time of recording-phase should be set as more than 94.57s. For example, by considering the efficiency of recording transactions, the time of recording-phase can be set as 240s.

B. FAIRNESS

In this subsection, we discuss the fairness of CP-consensus.

For Bitcoin. Firstly, due to the obvious block propagation delay, earlier block receivers have a longer time to compute the cryptopuzzle than later receivers. Secondly, if a malicious attacker has an enough computational power, then he can hide his valid blocks and selectively publish a part of them at some special time. Therefore, honest nodes, who strictly follows the protocol, are disadvantaged.

For CP-consensus. Firstly, selection of miners is fair. In particular, all record-nodes do not know who will be a miner in the next mining-phase until they synchronously receive the the corresponding CP-timestamp from CP-consensus. Secondly, computing the PoW is fair. Specifically, on the one hand, no one can hide any block to obtain any advantages. On the other hand, all miners begin to compute the PoW synchronously and no one can compute the PoW in advance.

C. RESISTANCE TO FORK-BASED ATTACKS

1) SELFISH MINING: Selfish miners generate a secret chain and honest miners generate a main chain. When secret chain is not shorter than main chain, selfish miners will selectively publish his secret fork according to the difference between secret chain and main chain in order to achieve more block revenue. For completing a selfish mining attack, selfish miners should generate a block in some round more early than honest nodes. Otherwise, selfish miners will follow blocks generated by honest miners. Furthermore, we assume that the length of secret chain is n and the length of main chain is m.

In Bitcoin, selfish mining attack strategies [20] are as follows: (1) when n=1 and $m = 0$, selfish miners do not publish the secret chain and continuously mine on the secret chain. However, if honest miners find a block. After that, $n = 1$ and $m = 1$. At this time, selfish miners immediately publish the secret chain. (2) when $n - m = 2$, but honest miners find a block. After that, $n - m$=1. At this time, selfish miners immediately publish the secret chain; (3) when $n - m > 2$, selfish miners selectively publish a part of the secret chain and continuously mine on the secret chain.

It should be pointed that, in strategies (2) and (3), selfish miners can obtain the whole block rewards of his secret chain. While, in strategy (1), selfish miners can obtain the reward of the secret chain with a certain probability.

In CP-consensus, for strategies (2) and (3), the validation of block-header can decide the validation of the corresponding block-body and only the block-header propagation delay causes the fork. Furthermore, because the time of propagating a block-header to 6000 record-nodes is about 0.0177s, so a fork will happen with a very little probability. However, to analyze the selfish mining attack, we assume: selfish miners find a block more early than honest miners; selfish miners can observe the first propagation of the block-header found by honest miners; selfish miners and honest miners can synchronously broadcast two block-header respectively. Besides, n and m should satisfy that $0 \leq n - m \leq 1$ since a record-node must generate or receive a block in every epoch and honest miners follow the majority-rule. Therefore, selfish mining strategies (2) and (3) can not occur in CP-consensus.

For strategy (1), because honest record-nodes follow the majority-rule, so if, at the fork station, the secret chain is accepted by a majority of record-nodes, then selfish miners win. Otherwise, selfish miners fail. However, when all nodes get into the next epoch, selfish miners will have no advantage to the next block. Therefore, selfish miners have to restart selfish mining attack in each epoch. Consequently, it must be pointed that if a miner generates a block first in a mining-phase, then he should immediately broadcast the corresponding block-header to obtain the block reward. Otherwise, a block generated by other record-nodes will be accepted by a majority of record-nodes with a high probability. Therefore, hiding block does not give any advantages for selfish miners.

2) GENERAL FORK-BASED ATTACKS: To analyze the resistance of fork-based attacks, we make an instance. Let the number of record-nodes be 750 and the average number of miners in the mining-phase be 15. Because miners are randomly selected in each mining-phase, so the probability of a record-node continuously being the miner in m continuous epochs is about $(\frac{1}{50})^m$. For a long-fork-based attack, the probability will be extremely small. Therefore, a attacker cannot accomplish a fork-based attack with an overwhelming probability.

Moreover, as mentioned in above subsection, an attacker must restart his attack in every epoch. Moreover, the length

of fork is at most 1. Therefore, it is too short to perform a fork-based attack (including selfish mining attack).

D. CONCENTRATION OF COMPUTATIONAL POWER

CP-consensus uses two methods to prevent the concentration of computational power.

- **The range of *nonce* is finite.** Essentially, if a miner has tried all choices of *nonce* before the end of mining-phase and does not find a solution, then he has nothing to do in the following time until next epoch comes. Therefore, a miner with a very large computational power has a certain amount of advantages, but his advantages are finite.
- **In each mining-phase, miners are randomly selected and they are a fraction of record-nodes.** Thus, if there is a mining pool, then most record-nodes of this pool will have nothing to do in many mining-phases. Moreover, this is an large loss for a mining pool.

VIII. CONCLUSION

In this paper, we present CP-consensus which is a blockchain protocol based on synchronous timestamps (CP-timestamp) of the Compass satellite. Firstly, record-nodes and cache-nodes can synchronously begin and end in each phase. Therefore, we propose a quasi-synchronous network for blockchain. Secondly, the block propagation delay is significantly reduced via adopting cache-nodes. Furthermore, the block verification can be performed before receiving the whole block. Thirdly, CP-consensus has a high throughput by setting a larger block size since the block verification delay is limited only by the block-header size. Fourthly, CP-consensus resists fork-based attacks and consumes a small amount of computational power. Finally, parameters setting and the security of CP-consensus are discussed.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China (NSFC) (Nos. 61370194, 61502048).

REFERENCES

[1] Nakamoto S.: Bitcoin: A peer-to-peer electronic cash system[J] (2008)
[2] https://blockchain.info/
[3] Litecoin. https://litecoin.org.
[4] Schwartz D., Youngs N., Britto A.: The Ripple protocol consensus algorithm[J]. Ripple Labs Inc White Paper 5 (2014)
[5] Wood G.: Ethereum: A secure decentralised generalised transaction ledger[J]. Ethereum Project Yellow Paper 151 (2014)
[6] Dwork C., Naor M.: Pricing via processing or combatting junk mail[C]. In Annual International Cryptology Conference, pp. 139-147. Springer, Berlin, Heidelberg (1992)
[7] King S, Nadal S. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake[J]. self-published paper, August, 2012, 19.
[8] J. Kwon. Tendermint: Consensus without mining, 2014. 2
[9] S. Park, K. Pietrzak, A. Kwon, J. Alwen, G. Fuchsbauer, and P. Gazi. Spacemint: A cryptocurrency based on proofs of space. Cryptology ePrint Archive, Report 2015/528, 2015. http://eprint:iacr:org/2015/528. 2, 5, 26
[10] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld. Proof of activity: Extending bitcoin's proof of work via proof of stake. In Proceedings of the ACM SIGMETRICS 2014 Workshop on Economics of Networked Systems, NetEcon, 2014. 2
[11] Blocki J, Zhou H S. Designing proof of human-work puzzles for cryptocurrency and beyond[C]//Theory of Cryptography Conference. Springer Berlin Heidelberg, 2016: 517-546.
[12] Castro M, Liskov B. Practical Byzantine fault tolerance[C]//OSDI. 1999, 99: 173-186.
[13] S. King and S. Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. 2012. https: //peercoin.net/assets/paper/peercoin-paper.pdf.
[14] CryptoManiac. Proof of stake. NovaCoin wiki, 2014. https://github.com/novacoin-project/novacoin/wiki/Proof-of-stake.
[15] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld. Proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract]. SIGMETRICS Perform. Eval. Rev., 42(3):34C37, Dec. 2014.
[16] http://www.szdih.com/
[17] Pappalardo G, Di Matteo T, Caldarelli G, et al. Blockchain Inefficiency in the Bitcoin Peers Network[J]. arXiv preprint arXiv:1704.01414, 2017.
[18] Decker C, Wattenhofer R. Information propagation in the bitcoin network[C]//Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on. IEEE, 2013: 1-10.
[19] Gervais A, Karame G O, Wst K, et al. On the security and performance of proof of work blockchains[C]//Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2016: 3-16.
[20] Ittay Eyal and Emin Gun Sirer. Majority is not enough: Bitcoin mining is vulnerable. In Financial Cryptography and Data Security, pages 436C454. Springer, 2014.
[21] Ghassan O. Karame, Elli Androulaki, and Srdjan Capkun. Double-spending fast payments in Bitcoin. In Proc. ACM Conf. Computer and Communications security (CCS), 2012.
[22] Heilman E. One weird trick to stop selfish miners: Fresh Bitcoins, a solution for the honest miner[C]//International Conference on Financial Cryptography and Data Security. Springer, Berlin, Heidelberg, 2014: 161-162.
[23] Ittay Eyal. The miner's dilemma. arXiv preprint arXiv:1411.7099, 2014.
[24] Liao K, Katz J. Incentivizing blockchain forks via whale transactions[C]//Bitcoin Workshop. 2017.
[25] Bellare M, Rogaway P. Random oracles are practical: A paradigm for designing efficient protocols[C]//Proceedings of the 1st ACM conference on Computer and communications security. ACM, 1993: 62-73.
[26] Croman K, Decker C, Eyal I, et al. On scaling decentralized blockchains[C]//International Conference on Financial Cryptography and Data Security. Springer Berlin Heidelberg, 2016: 106-125.
[27] How a Visa transaction works. http://web.archive.org/web/20160121231718/http://apps.usa.visa.com/merchants/become-a-merchant/ how-a-visa-transaction-works.jsp, 2015.
[28] Garay J, Kiayias A, Leonardos N. The bitcoin backbone protocol: Analysis and applications[C]//Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Berlin, Heidelberg, 2015: 281-310.
[29] Miller A, Jansen R. Shadow-Bitcoin: Scalable Simulation via Direct Execution of Multi-threaded Applications[J]. IACR Cryptology ePrint Archive, 2015, 2015: 469.
[30] Sallal M, Owenson G, Adda M. Bitcoin network measurements for simulation validation and parametrisation[C]//11th International Network Conference. 2016.
[31] Neudecker T, Andelfinger P, Hartenstein H. A simulation model for analysis of attacks on the bitcoin peer-to-peer network[C]//Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on. IEEE, 2015: 1327-1332.
[32] Eyal I, Gencer A E, Sirer E G, et al. Bitcoin-NG: A Scalable Blockchain Protocol[C]//NSDI. 2016: 45-59.
[33] Heilman E. One weird trick to stop selfish miners: Fresh Bitcoins, a solution for the honest miner[C]//International Conference on Financial Cryptography and Data Security. Springer, Berlin, Heidelberg, 2014: 161-162.
[34] Micali S. ALGORAND: the efficient and democratic ledger[J]. arXiv preprint arXiv:1607.01341, 2016.
[35] Solat S, Potop-Butucaru M. ZeroBlock: Preventing Selfish Mining in Bitcoin[J]. arXiv preprint arXiv:1605.02435, 2016.
[36] Tang G, Liu L, Cao J, et al. Performance Analysis for Time Synchronization with Compass Satellite Common-View[C]//China Satellite Navigation Conference (CSNC) 2012 Proceedings. Springer, Berlin, Heidelberg, 2012: 483-490.
[37] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In CRYPTO'92, pages 139-147, 1992.
[38] Sapirshtein A, Sompolinsky Y, Zohar A. Optimal selfish mining strategies in bitcoin[C]//International Conference on Financial Cryptography and Data Security. Springer, Berlin, Heidelberg, 2016: 515-532.
[39] Heilman E, Kendler A, Zohar A, et al. Eclipse Attacks on Bitcoin's Peer-to-Peer Network[C]//USENIX Security Symposium. 2015: 129-144.

Lijing Zhou received the B.Sc degree in mathematics and applied mathematics from Inner Mongolia Normal University, China, in 2009, and the M.Sc degree in cryptography from Xidian University, China, in 2013, and he is currently pursuing the Ph.D degree in information security from Beijing University of Posts and Telecommunications. His current research interests include blockchain technology and privacy and cryptography.

Licheng Wang received B.Sc degree in computer science from Northwest Normal University, China, in 1995, and M.Sc degree in mathematics from Nanjing University, China, in 2001, and received Ph.D degree in cryptography from Shanghai Jiaotong University, China, in 2007. He is currently associate professor with Beijing University of Posts and Telecommunications.

Yiru Sun received the B.Sc degree in mathematics and applied mathematics from Inner Mongolia Normal University, China, in 2009, and the M.Sc degree in cryptography from Xidian University, China, in 2014, and she is currently pursuing the Ph.D degree in information security from Beijing University of Posts and Telecommunications. Her current research interests include blockchain technology and privacy and quantum cryptography.