

Goshawk: A Novel Efficient, Robust and Flexible Blockchain Protocol

Cencen Wan, Shuyang Tang, Yuncong Zhang, Chen Pan,
Zhiqiang Liu**, Yu Long**, Zhen Liu**, Yu Yu**

Department of Computer Science and Engineering,
Shanghai Jiao Tong University, China
{ilu_zq, longyu, liuzhen}@sjtu.edu.cn, yuyu@cs.sjtu.edu.cn

Abstract. Proof of Work (PoW), a fundamental blockchain protocol, has been widely applied and thoroughly testified in various decentralized cryptocurrencies, due to its intriguing merits including trustworthy sustainability, robustness against Sybil attack, delicate incentive-compatibility, and openness to any participant. Meanwhile, PoW-powered blockchains still suffer from poor efficiency, potential selfish mining, to-be-optimized fairness and extreme inconvenience of protocol upgrading. Therefore, it is of great interest to design new PoW-driven blockchain protocol to address or relieve the above issues so as to make it more applicable and feasible. To do so, we present Goshawk, a novel hybrid consensus protocol, in which a two-layer chain structure with two-level PoW mining strategy and a ticket-voting mechanism are elaborately combined. We show that this newly-proposed protocol has the merits of high efficiency, strong robustness against “51%” attack of computation power, as well as good flexibility for future protocol updating. As far as we know, Goshawk is the first blockchain protocol with these three key properties. Last but not the least, this scheme has been implemented and deployed in the testnet of the public blockchain project Hcash¹ for months, and has demonstrated its stability and high efficiency with such real-world test.

1 Introduction

To date, Bitcoin [27] and a variety of other cryptocurrencies have drawn much attention from researchers and fintech industry. Their attractive innovations show great promise of fundamental change in payments, economics and politics around the world [28,39]. Recently, cryptocurrencies’ global market capitalizations have reached more than \$250 billions [5]. The blockchain technique, which is the underlying technique of various decentralized cryptocurrencies, is an ingenious combination of multiple technologies such as peer-to-peer network, consensus protocol over a distributed network, cryptographic schemes, and so on. This technique provides a decentralized way to securely manage ledgers, which is fundamental for building trust in our social and economic activities.

Proof of Work (PoW), which relies on computational puzzles (a.k.a. moderately hard functions) introduced by Dwork and Naor[12], is a blockchain protocol used to maintain the consistency of distributed ledger in a decentralized setting so as to prevent fraud and double-spending attacks. So far it has been implemented in 250 cryptocurrencies or more such as Bitcoin, Ethereum, and so on, serving as the underlying blockchain protocol. PoW has amazing features including trustworthy sustainability, robustness against Sybil attack, delicate incentive-compatibility, and openness to any participant (i.e., participants could join and leave dynamically), though it still needs to be improved in the following aspects:

** Corresponding Authors

¹ <https://github.com/HcashOrg>

- *Efficiency.* The transaction throughput of PoW-driven blockchain does not scale well. For instance, Bitcoin supports very limited transaction throughput (say, up to 7 transactions per second [2]), while the demand from practical applications is much higher (MasterCard and VISA are reported to process 1200 to 56000 transactions per second).
- *Fairness.* PoW-based blockchains have been criticized for the potential of centralization of computation power [28]. Even a minor enhancement in fairness is welcome, since it provides fewer incentives for miners to join forces to enjoy the advantage of mining in a larger pool. This mitigates the centralization of the mining power, thus improving the security property of blockchains.
- *Robustness.* It is known that in PoW protocol, selfish mining attack [15,13,29,36] allows that adversaries deviating from the protocol may gain a disproportionate share of reward, much more than they deserve. Besides, PoW protocol is intrinsically subject to “51%” attack of computation power.
- *Flexibility.* In practice, it is extremely difficult to fulfill blockchain protocol evolution. For example, modification to scale up existing protocol is a raging debate in the Bitcoin community [18,6,17,31].

Till now, many attempts have been made to address or mitigate the issues related to PoW protocol so as to make it more powerful. One approach is to reduce the block interval to shorten latency. However, this approach compromises certain stability or security of the decentralized system, which has been proven by the practice of Ethereum [8]. Specifically, the short block interval (12s averagely) adopted in Ethereum brings instability to the system. To solve this issue, Ethereum implements the GHOST protocol [37] which maintains the main chain at a fork by choosing the side whose sub-tree contains more work (accumulated over all blocks in the sub-tree). GHOST improves the mining power utilization and fairness under high contention, but has the weakness that in some cases, no single node has enough information to determine which is the main chain. The second approach is to enlarge the block. It improves throughput, but aggravates communication burden to the network, which in turn increases the stale block rate, and finally damages the security of PoW-based blockchain [19]. The third approach is to use sharding mechanism to achieve a sweet spot between PoW and classical Byzantine consensus protocol [25], which leads to throughput scaling. The key idea in this approach is to partition the network into smaller committees, each of which processes a disjoint set of transactions. Each committee has a reasonably small number of members so they can run a classical Byzantine consensus protocol to decide their agreed set of transactions in parallel. The fourth approach is to perform transactions off the chain, such as lightning network[32], raiden network [1], and so on [11,26,21]. These works allow for extensive payment networks where transactions can be performed efficiently and scalably without trusted middlemen, especially targeting on fast micropayments. Moreover, Eyal et al. proposed Bitcoin-NG [14], a scalable blockchain protocol, by introducing a two-layer chain structure which consists of keyblocks and microblocks. Bitcoin-NG boosts transaction throughput by decoupling PoW protocol into two planes: leader election and transaction serialization. Once a miner generates a keyblock by solving the computational puzzle, he is elected as a potential leader and entitled to serialize transactions into microblocks unilaterally until a new leader is chosen. Although the above approaches provide some interesting ideas for improving PoW protocol, they mainly focus on the efficiency issue related to PoW.

On the other hand, alternative blockchain protocols have been introduced to replace PoW. Among them the most promising ones may be the *Proof of Stake* (PoS) [34,23] and its variants such as Snow White [7], Ouroboros [22], Ouroboros Praos [10] and Algorand [20]. PoS protocol grants the right of generating blocks to stakeholders instead of miners with computational power. Specifically, in PoS protocol, rather than miners investing computational resources in order to participate in the leader election (i.e., block generation) process, they instead run a process that randomly selects one of them proportionally to the stake that each possesses according to the current blockchain

ledger. The rationale behind PoS is that stakeholders are motivated to maintain the consistency and security of blockchain, since the value of their stake will shrink when these properties are compromised. Although PoS protocol owns intriguing potential, its practicality, applicability and robustness still need to be examined extensively via a mass of public blockchains implementing PoS as their underlying protocol before it is widely admitted. Another interesting direction is to adopt DAG(Directed Acyclic Graph)-based framework instead of blockchain structure to acquire high throughput by exploiting the high concurrency nature of DAG structure [9,24,33]. However, to date, there has not been any rigorous security guarantee for DAG-based distributed ledger technology, thus the security of this technology needs to be investigated further.

As PoW protocol has already demonstrated its practicality – PoW-powered blockchains currently account for more than 90% of the total market capitalization of existing digital cryptocurrencies, and its importance in permission-less network was also stated by Pass and Shi in [30], it is of great interest to strength PoW further by addressing or mitigating the related issues mentioned above. Nevertheless, it can be seen that the current state of the art in improving PoW protocol is still far from satisfactory.

1.1 Our Contribution

In this work, we propose Goshawk, the first brand-new candidate of PoW-powered blockchain protocol with high efficiency, strong robustness, as well as good flexibility. Goshawk is actually a hybrid consensus protocol, in which a two-layer chain structure with two-level PoW mining strategy and a ticket-voting mechanism are combined delicately. More specifically, we adopt the two-layer chain structure (i.e., keyblocks/microblocks) given in Bitcoin-NG, and further improve it by introducing two-level PoW mining strategy (i.e., keyblocks and microblocks with two different mining difficulties, respectively). This guarantees the high throughput of our scheme, while obviating the vulnerability to the attack of microblock swamping in Bitcoin-NG. Furthermore, we borrow the idea of the ticket-voting approach presented in DASH[3] and Decred[4], and refine this idea by formalizing it into a more rigorous mechanism, then we combine this mechanism with the above chain structure elaborately to attain strong security and good flexibility. Security analysis of our scheme shows that it is incentive-compatibility, and robust against selfish mining and “51%” attack of computation power. Besides, we demonstrate that our scheme also allows good flexibility for future protocol updating effectively. At last, this scheme has been implemented and deployed in the testnet of the public blockchain project Hcash for months, and has demonstrated its good stability and promising scalability with such real-world test. This also suggests the interesting potential that our scheme could be employed in next-generation cryptocurrencies.

1.2 Paper Organization

The remainder of the paper is organized as follows. Sec.2 presents Goshawk, a novel hybrid consensus protocol. Then we analyze the security of this protocol in Sec.3. Further, we introduce a two-phase upgrade process to demonstrate the flexibility of Goshawk in Sec.4. The protocol evaluation and performance test of Goshawk in a real-world setting are shown in Sec.5. Finally, we conclude our work in Sec.6.

2 The Goshawk Protocol

The Goshawk protocol extends the Bitcoin-NG scheme, which improves the scalability of Bitcoin by introducing a two-layer chain structure consisting of keyblocks and microblocks, while avoiding

the microblock swamping attack in Bitcoin-NG². Tab. 1 presents some of the notations used in this section.

Notation	Description
D	The confirmations required for tickets maturity
E	The maximum number of tickets per keyblock
\mathcal{F}	The price adjustment function
L	System optimal size of ticket pool
N	The number of tickets selected by each keyblock
P	The Ticket price
\mathcal{P}	The function mapping keyblock to N tickets
R	The initial key height of the ticket-voting mechanism
S	Total amount of stakes
T	The average keyblock generation interval
TP	The ticket pool

Table 1. Table of Notations

2.1 Two-level Mining Mechanism

Similar to the idea of subchains [38,35], we propose a *two-level mining mechanism*, in which we set two levels of difficulty for computation puzzle, to address the microblock swamping attack on Bitcoin-NG. Solving the puzzle with low difficulty allows a miner to generate a *microblock*. If the solution simultaneously meets the standard of higher difficulty, this block is called a *keyblock*. We set the ratio of mining difficulty between keyblock and microblock by m . If the average keyblock generation interval is T , then the average microblock generation interval is $t = T/m$.

A *fork* happens when multiple blocks follow the same parent. In that case, we say the blockchain has more than one *branches*. We define *main chain* as the branch containing the most keyblocks. If there are more than one branches that satisfy the condition above, a miner will select one of them randomly as the main chain. This is called the *longest keyblock chain rule*. We define the *height* of a block (either keyblock or microblock) as the number of blocks before it and the *key height* of a block as the number of keyblocks before it, in the same branch.

Definition 1 presents the structure of the block in our improved scheme.

Definition 1 (Block Structure I). We define a block, denoted by B , as the following tuple

$$B = (H_{tip,B}, H_{tip,K}, h, k, \{tx\}, n)$$

where

- $H_{tip,B}$ is the hash of previous block (either keyblock or microblock);
- $H_{tip,K}$ is the hash of previous keyblock;
- h is the block height;
- k is the key height;
- $\{tx\}$ is the transaction set contained in the block;
- n is the nonce found by the miner.

² Considering the cheap and quick generation of microblocks, a leader can swamp the system with microblocks. Specifically, in Bitcoin-NG, although a minimal interval between two sequential microblocks could be set to avoid massive microblocks in a single microblock chain, the malicious leader could generate tremendous amount of microblock branches. For other parties, since each branch is self-consistent, they have to relay all these branches. This eventually paralyzes the whole network, causing legal transactions and blocks fail to spread.

B is a valid keyblock if $\text{Hash}(B) \leq T_K$, where T_K is the threshold of computation puzzle for keyblock; B is a valid microblock if $T_K < \text{Hash}(B) \leq T_M$, where T_M is the threshold for microblock.

We denote the block being mined by miner P as B_{temp} . Let $H_{\text{temp}} = \text{Hash}(B_{\text{temp}})$. Miner P increments n starting from 0, until $H_{\text{temp}} \leq T_M$. If $H_{\text{temp}} \leq T_K$, P broadcasts B_{temp} as a new keyblock K_{new} , otherwise P broadcast B_{temp} as a new microblock M_{new} . Other participants determine whether a received block is a keyblock or microblock depending on its hash value, and update their main chain according to the longest keyblock chain rule. It can be easily inferred that in our two-level mining mechanism, for miners mining both keyblocks and microblocks, no additional computation power needs to be consumed compared with miners only mining keyblocks.

Fork. Since mining microblock is relatively easy, microblock forks happened frequently. However, once a new keyblock is created, all honest nodes will follow the chain with the most keyblocks and such forks vanish. The new scheme will also experience keyblock forks, which will happen rarer than microblocks. The duration of such a fork may be long and the fork finally dissolved in several keyblock confirmations. Though the works for microblocks contribute nothing to the selection of branches, it is hard enough for spamming. According to the *common prefix property* described in [16], we declare a block is *stable* if we prune all of the blocks after it in the main chain, the probability that the resulting pruned chain will not be mutual prefix of other honest miners' main chain is less than a security parameter $2^{-\lambda}$.

2.2 Ticket-voting Mechanism

In our scheme, we borrow the ticket-voting mechanism from DASH and Decred. The core idea is stakeholders lock their stakes for purchasing *ticket* to proportionally obtain rights for a future *vote*. The ticket structure is presented in Definition 2 and vote structure is presented in Definition 3.

Definition 2 (Ticket Structure). We define a special transaction as a *ticket*, denoted by tk , as the following tuple

$$tk = (\langle \text{Hash}(tx), i, sig \rangle, \langle P, pk \rangle)$$

where

- $\langle \text{Hash}(tx), i, sig \rangle$ is the input of ticket;
- tx is a transaction as the source of funding for ticket purchasing ;
- i is the order of output in tx . The amount of stakes in this output must larger than P ;
- sig is the signature used to verify input;
- $\langle P, pk \rangle$ is the output of the ticket which will lock P stakes for ticket purchasing;
- P is a certain amount (called ticket price) of stakes locked for purchasing ticket;
- pk is a public key.

Only keyblock could contain tickets, and the number of tickets contained in each keyblock is limited to E such that no one can spam tickets into blockchain.

Definition 3 (Vote Structure). We define a special transaction as a *vote*, denoted by vt , as the following tuple

$$vt = (\langle \text{Hash}(tk), sig \rangle, \langle V, pk \rangle)$$

where

- $\langle \text{Hash}(tk), sig \rangle$ is the input of vote;
- tk is the ticket for voting;
- $\langle V, pk \rangle$ is the output of the vote which will refund the locked stakes;
- V is a specific amount of rewards.

Ticket Pool. A ticket is called *mature*, denoted by mtk , if the containing keyblock is followed by at least D keyblocks, well after the containing keyblock becomes stable. If a mature ticket was spent on voting, we denote it by stk . We call the set of all unspent mature tickets the *ticket pool*, denoted by TP , i.e. $TP = \{mtk\} \setminus \{stk\}$.

Validation Rule. A keyblock is considered valid only if a majority (more than half) of its votes are collected by its successive keyblock. We stipulate that miners can only mine after a validated keyblock (or a microblock preceded by a validated keyblock) by collecting votes as a validation proof for this keyblock. We call this the *validation rule*. If a keyblock is not validated by majority votes, miners would ignore this keyblock.

The ticket-voting mechanism is divided into four steps.

- Participants purchase tickets which will be added to the ticket pool after D confirmations.
- Each latest keyblock is mapped into N random tickets from TP via a function \mathcal{P} which is defined in Definition 4.
- Owners of selected tickets issue *votes*, if the corresponding keyblock is valid.
- Miners collect votes and select mining strategies according to the validation rule.

Definition 4 (Mapping Function). We define a mapping function

$$\mathcal{P}(TP, \text{Hash}(K)) = \{tk_j\}_{j \in [N]} = \underset{tk_1, tk_2, \dots, tk_N \in TP}{\operatorname{argmax}} \left\{ \sum_{i=1}^N \text{Hash}(\text{Hash}(tk_i) \oplus \text{Hash}(K)) \right\}$$

where K is a keyblock.

Each ticket can only be chosen once and then be removed from ticket pool even if the owner missed the voting, in which case the ticket is considered *missed*. The owner of a selected ticket will be refunded with the stake locked by this ticket, and a voted (not missed) ticket additionally brings the owner a specific amount of rewards.

Ticket Price Adjustment Function. The ticket price is adjusted by the function $\mathcal{F}(|TP|, P, L)$ which takes as input the size of TP , the current ticket price P and a parameter L . \mathcal{F} returns a new price P' which increases exponentially compared to P if $|TP| > L$, and P' decreases when $|TP| < L$. Therefore, when $|TP| > L$, users are more reluctant to purchase tickets, and when $|TP| < L$ users are more willing to. In this way, \mathcal{F} keeps the size of TP close to L , thus on average each ticket waits time $(L/N + D) \times T$ before it is chosen.

A stakeholder with a p fraction of total stakes gains a disproportionate advantage by engaging in ticket purchasing if others do not devote all their stakes into tickets. To reduce such advantage, L should be large enough such that $L \times P \approx S/f$, where S is the total amount of stakes and f is a constant greater than 1. A stakeholder who holds β fraction of the tickets in TP has a probability of $M(N, \beta)$ to reach majority in the chosen tickets of a keyblock, where $M(N, \beta) = \sum_{i=\lfloor N/2 \rfloor + 1}^N \binom{N}{i} \beta^i (1 - \beta)^{N-i}$.

On startup, PoW is the only consensus protocol because ticket pool is empty at the beginning of the chain. The ticket-voting mechanism begin at key height R which is selected such that $R = L/E + D$.

2.3 Goshawk: Hybrid Consensus Scheme

We combine the improved two-layer blockchain structure with the ticket-voting mechanism mentioned above to construct our novel hybrid consensus scheme, Goshawk. The new block structure is presented in Definition 5.

Definition 5 (Block Structure II). We define a block, denoted by B , as the following tuple

$$B = (H_{tip,B}, H_{tip,K}, h, k, \{tx\}, \{tk\}, \{vt\}, n)$$

where $H_{tip,B}$, $H_{tip,K}$, h , k , $\{tx\}$, n are as in Definition 1.

Compared to the mining process described in Section 2.1, in this combined scheme, the miner P needs to take the following additional steps.

- In addition to the transaction $\{tx\}$, B_{temp} also contains a set of ticket purchasing transactions $\{tk\}$, which were collected and stored locally by P similar to ordinary transactions;
- P collects at least $\lfloor N/2 \rfloor + 1$ votes for the previous keyblock (whose hash is $H_{tip,K}$), and put this set of votes $\{vt\}$ into B_{temp} . If P fails to collect enough votes for $H_{tip,K}$, she abandons this keyblock and continue to mine after the previous keyblock.
- The $\{tk\}$ and $\{vt\}$ will be ignored if B_{temp} turns out to be a microblock, since they can only be contained in keyblocks.

When a newly generated keyblock K_{new} travels around the network, the stakeholders chosen by this keyblock check this keyblock, issuing and broadcasting votes if it is valid. Other miners collect these votes and switch to mine after K_{new} as soon as the votes satisfy the majority rule. The mining process is described in Algorithm 1. The structure of Goshawk is shown in Fig.1.

Algorithm 1 Mining process in Goshawk

```

1: procedure MINING
2: loop:
3:    $B_{temp} \leftarrow H_{tip,B} || H_{tip,K} || h || k || n || \{tx\} || \{tk\} || \{vt\}$ 
4:   if  $\text{Hash}(B_{temp}) \leq T_K$  then
5:      $K_{new} \leftarrow B_{temp}$ 
6:      $P$  broadcast  $K_{new}$ 
7:   else if  $\text{Hash}(B_{temp}) \leq T_M$  then
8:      $M_{new} \leftarrow B_{temp}$ 
9:      $P$  broadcast  $M_{new}$ 
10:  end if
11:  if  $P$  received  $K_{new}$  and  $\text{ReceiveMajorityVotesOf}(K_{new})$  and  $\text{IsTipOfMainChain}(K_{new})$  then
12:     $H_{tip,B} \leftarrow \text{Hash}(K_{new})$ 
13:     $H_{tip,K} \leftarrow \text{Hash}(K_{new})$ 
14:     $h \leftarrow \text{GetHeightOf}(K_{new}) + 1$ 
15:     $k \leftarrow \text{GetKeyHeightOf}(K_{new}) + 1$ 
16:  else if  $P$  received  $M_{new}$  and  $\text{ReceiveMajorityVotesOf}(\text{GetPreviousKeyBlockOf}(M_{new}))$ 
17:    and  $\text{IsTipOfMainChain}(M_{new})$  then
18:     $H_{tip,B} \leftarrow \text{Hash}(M_{new})$ 
19:     $H_{tip,K} \leftarrow \text{Hash}(\text{GetPreviousKeyBlockOf}(M_{new}))$ 
20:     $h \leftarrow \text{GetHeightOf}(M_{new}) + 1$ 
21:     $k \leftarrow \text{GetKeyHeightOf}(M_{new})$ 
22:  end if
23:   $n \leftarrow n + 1$ 
24:  goto loop.
25: end procedure

```

Incentive Mechanism. We divide the block reward into two parts, giving a ratio w of the rewards to keyblock miners, and giving voters the rest, therefore each voter earns $(1 - w)/N$ of

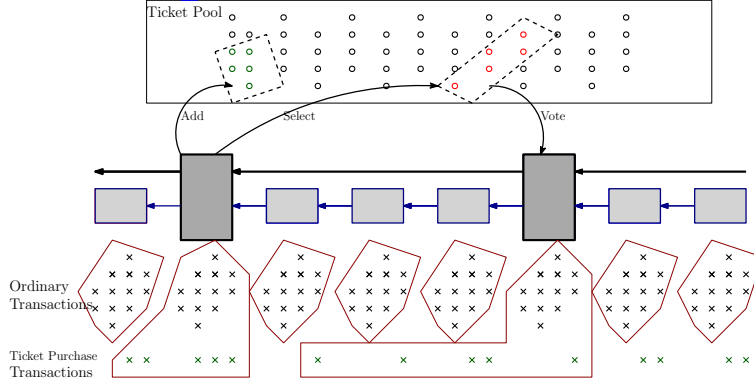


Fig. 1. The structure of Goshawk. The tickets are denoted by dots, the transactions are denoted by cross marks, the keyblocks are denoted by big rectangles, and the microblocks are denoted by small rectangles. A keyblock contains transactions and add tickets into ticket pool. Meanwhile, a keyblock pseudo-randomly selects tickets which will be removed from ticket pool. Chosen stakeholders vote for keyblock to validate it and votes will be contained by the next keyblock.

the block reward. The block rewards are spendable only after the containing keyblock is followed by D keyblocks. To encourage keyblock miners to collect as many votes as they can, the actual block reward a miner earned is based on how many votes she collects. For example, if M votes are collected, $(1 - w)M/N$ is the precise reward. If a voter misses to vote, she also misses the reward. Microblock miners share the transaction fees, which is split into three parts, where 60% is given to the miner whose block (either keyblock or microblock) contains the transaction, 30% to the next block and 10% to the next keyblock.

3 Security Analysis

Our protocol has two goals. One is the incentive compatibility. All rational participants would operate honestly since they benefit nothing from deviating the protocol. Another one is robustness against "51%" attack and selfish mining attack.

3.1 Incentive Compatibility

We show that our Goshawk scheme is incentive-compatible (i.e. each participant benefits nothing from deviating from the protocol) under the assumption that all participants are rational.

Strategy of rational participants. In this part, we show that all rational participants obey the mining rule. That is, rational participants always mine on the latest validated block. In another word, any participant gains no marginal revenue by deviating from the rule above (i.e. the Nash equivalence of Goshawk). The latest valid block can be a keyblock or a microblock, in the following, we will discuss the rational strategies for each case respectively. Tab. 2 presents some of the notations used subsequently.

Case 1. A keyblock as the latest valid block. As shown in Fig. 2, when the latest valid block is a keyblock, one participant may mine after the latest keyblock (block E), or after the previous microblock (block D) to reach a higher revenue. We compare the expected revenues with two strategies above and prove that following the longest keyblock rule (i.e. mining after block E) is the rational choice. In the following discussion, we use q to denote the probability of one participant's generating the next block and the block is keyblock, then its probability for her to

Notation	Description
a	The fraction of transaction fee included in one block for the current block owner
b	The fraction of transaction fee included in one block for the next block owner
B	The block reward for a keyblock
c	The fraction of transaction fee included in one block for the next keyblock owner
F	Total transaction fees included in one block
m	The difficulty ratio, i.e. the ratio of difficulties of mining a keyblock and mining a microblock
q	The probability for one miner to generate the next block, and this block is keyblock

Table 2. Table of Notations

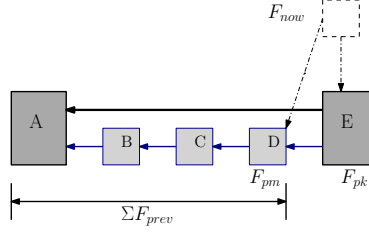


Fig. 2. Mining Strategy in Case 1.

generate the next block and the block is a microblock is $(m - 1)q$. Also, we assume each keyblock contains block reward B , and each block (keyblock or microblock) contains the same amount of transaction fee F . For distinction, we denote the transaction fee in a microblock by F_{pm} , the fee in a keyblock by F_{pk} , and the fee in the block currently being mined by F_{now} . Moreover, ΣF_{prev} (ideally $\Sigma F_{prev} \approx mF$) denotes the sum of all transaction fees included from the previous keyblock (block A in Fig. 2) to the previous microblock (block D).

	Mining a keyblock	Mining a microblock
Block E	$(a + c) \times F_{pk} + b \times F_{now} + B$	$a \times F_{pk} + b \times F_{now}$
Block D	$c \times \Sigma F_{prev} + a \times F_{pm} + b \times F_{now} + B$	$a \times F_{pk} + b \times F_{now}$

Table 3. Revenues Following Block E or D .

Hence, the expected revenue following the right block (block E) is

$$\begin{aligned}
 R &= q \times ((a + c) \times F_{pk} + b \times F_{now} + B) + (m - 1)q \times (a \times F_{pk} + b \times F_{now}) \\
 &= q \times ((a + c) \times F + b \times F + B) + (m - 1)q \times (a \times F + b \times F) \\
 &= ((1 - c) \times m + c)q \times F + q \times B.
 \end{aligned}$$

By deviating the rule, a miner may generate a block after block D . In this case, we regard that the probability of its block's conquering the existing block is smaller than $1/2$. This is simple to understand since less than half participants switches to an alternative chain when the chain is forked into two branches. Due to this, the expected revenue via mining after block D is

$$\begin{aligned}
 R' &< \frac{1}{2} \times q \times (c \times \Sigma F_{prev} + a \times F_{pm} + b \times F_{now} + B) + \frac{1}{2} \times (m - 1)q \times (a \times F_{pk} + b \times F_{now}) \\
 &= 0.5mq \times F \times (a + b) + 0.5q \times c \times \Sigma F_{prev} + 0.5q \times B.
 \end{aligned}$$

Obviously, ΣF_{prev} is related to the number of microblocks between two keyblocks. Let X be a random variable which denotes the number of microblocks between two keyblocks. Then, X follows a geometric distribution with parameter m . Thus, we have: $P(X = k) = \frac{1}{m} (1 - \frac{1}{m})^{k-1}$. For a given parameter θ , we can get:

$$P_1 := \sum_{k=\theta}^{\infty} P(X = k) = \sum_{k=\theta}^{\infty} \frac{1}{m} (1 - \frac{1}{m})^{k-1} = (1 - \frac{1}{m})^{\theta \times m - 1}$$

This probability P_1 increases in m , and we have $\lim_{m \rightarrow \infty} P_1 = e^{-\theta}$. Thus, we get $P_1 < e^{-\theta}$. If we set parameter $\theta = 5$, the probability that there are $5m$ microblocks between two keyblocks is under 0.62%, which is small. According to above analysis, $\Sigma F_{\text{prev}} \leq 5mF$, (ideally $\Sigma F_{\text{prev}} \approx mF$). Thus,

$$\begin{aligned} R' &< 0.5mq \times F \times (a + b) + 0.5q \times c \times \Sigma F_{\text{prev}} + 0.5q \times B \\ &< 0.5mq \times F \times (1 + 4c) + 0.5q \times B. \end{aligned}$$

Letting $R > R'$, we get $\frac{1}{2}(1 + 4c) < 1 - c$ and hence $c < \frac{1}{6}$. In our implementation, we select $a = 0.3, b = 0.6, c = 0.1$. Then, the expected revenue following the right block (block E) is

$$R \approx ((1 - c) \times m + c)q \times F + q \times B = (0.9m + 0.1)q \times F + q \times B.$$

And the expected revenue via mining after block D is

$$R' < 0.5mq \times F \times (1 + 4c) + 0.5q \times B \approx 0.7mq \times F + 0.5q \times B.$$

Obviously $R' < R$, which leads to the conclusion that the rational strategy is to follow the right block E .

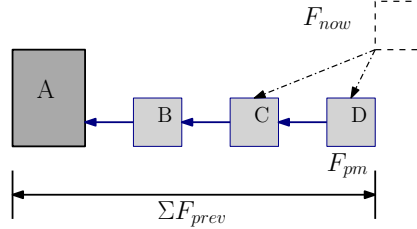


Fig. 3. Mining Strategy in Case 2.

Case 2. A microblock as the latest valid block. In the second case (as shown in Fig. 3), one miner may mine a block following C or D . However, all the revenues received by following C can also be received by following D . Moreover, the miner will lost transaction fees of block D , if she mines after block C and successfully finds a keyblock. For this reason, rational participants always mine after the latest block in this case.

3.2 Robustness

Fault-tolerance property. We assume a worst adversary who tries to undermine the system by proposing an invalid block without considering its own merits.

In this part, microblocks are not considered since they have nothing to do with the forks of the main chain. Therefore, we directly use “block” in place of “keyblock” when no ambiguity exists. In

a purely PoW-based cryptocurrency like Bitcoin, the probability of one participant’s undermining the system is roughly same as the fraction of its computation power among all participants. This is the fault-tolerance property of PoW. However, in a hybrid scenario, the description of the fault-tolerance property is more sophisticated. To begin with, we propose a definition.

Definition 6 (φ -fault-tolerance). *For a binary function $\varphi : [0, 1] \times [0, 1] \rightarrow [0, 1]$, a cryptocurrency scheme achieves φ -fault-tolerance, if and only if for any adversary with α fraction of total computation power and β fraction of total stake, its probability of successfully proposing an invalid block should be no greater than $\varphi(\alpha, \beta)$.*

From this definition, we can formally analyze the fault-tolerance of our newly proposed Goshawk consensus scheme.

Theorem 1 (Fault-tolerance of Goshawk). *Goshawk achieve an $\frac{\alpha\gamma(\beta)}{1-\alpha-\gamma(\beta)+2\alpha\gamma(\beta)}$ -fault-tolerance, where $\gamma(\beta) = \sum_{i=\lfloor N/2 \rfloor + 1}^N \binom{N}{i} \beta^i (1-\beta)^{N-i}$, N is number of tickets each block selects.*

Proof. Since $\varphi(\alpha, \beta)$ is an upper-bound of adversary’s advantage, we can assume that all malicious computation power and stakes are held by one single adversary. By the definition of fault-tolerance, the adversary with computation power of rate α and stake of rate β tries to mine an invalid block and proposes this block (i.e. the malicious block is voted by most corresponding ticket voters, since honest voters will not vote to invalid blocks, this equals to having at least half voters controlled by this adversary). Also, the adversary does not vote on all blocks generated by honest parties.

For simplicity, we define the following three events.

- E_A : A keyblock is found by the adversary, and most of its corresponding tickets are controlled by the adversary.
- E_B : A keyblock is found by the adversary, while most of its corresponding tickets are at hands of honest parties.
- E_C : A keyblock is found by an honest participant, while most of its corresponding tickets are controlled by the adversary.

From here, we can calculate the upper-bound of adversary’s chance of proposing an invalid block. Obviously,

$$\varphi(\alpha, \beta) = \sum_{i=1}^{\infty} (\Pr[E_B \vee E_C])^{i-1} \Pr[E_A] = \sum_{i=1}^{\infty} (\alpha(1-\gamma) + (1-\alpha)\gamma)^{i-1} \alpha\gamma = \frac{\alpha\gamma}{1-\alpha-\gamma+2\alpha\gamma},$$

where γ is the probability that most corresponding tickets regarding to one block is held by the adversary: $\gamma(\beta) = \sum_{i=\lfloor N/2 \rfloor + 1}^N \binom{N}{i} \beta^i (1-\beta)^{N-i}$.

We can observe that when $\gamma = 1$, the adversary can successfully deny any blocks not proposed by herself, and hence $\varphi(\beta) = 1$. On the contrary, when $\gamma = 0$, any adversary block is denied by honest participants, and therefore $\varphi(\beta) = 0$. These are satisfied in case of $\varphi(\alpha, \beta) = \frac{\alpha\gamma(\beta)}{1-\alpha-\gamma(\beta)+2\alpha\gamma(\beta)}$.

For any adversary with α fraction of total computation power and β fraction of total stake, to perform a “51%” attack, it should at least attain $\varphi(\alpha, \beta) > \frac{1}{2}$. That is, $\frac{\alpha\gamma}{1-\alpha-\gamma+2\alpha\gamma} > \frac{1}{2} \iff \alpha > 1-\gamma$. Assuming that $\beta = 20\%$, $N = 5$, then $\gamma \approx 6\%$, and the adversary must have over $1-\gamma \approx 94\%$ total computation power to successfully launch a “51%” attack.

Selfish Mining Resistance. In a purely PoW-based cryptocurrency system, the selfish mining can be relatively easily performed by continuously mining in a separated environment, and is thereby hard to notice, and hard to prevent. For instance, an adversary with more than 1/3 total hash rate (instead of 1/2) can launch the selfish mining attack. However, in Goshawk, a

block has to be validated by corresponding voters. That is to say, to secretly mining a continuous sequence of blocks, a block is only “useful” when its corresponding tickets are mostly held by itself. Formally, to prevent adversary’s launching the selfish mining attack, instead of purely PoW-based cryptocurrencies’ $\alpha < \frac{1}{3}$ (see explanation in [15]), we have an upper bound $\varphi(\alpha, \beta) < \frac{1}{3}$. That is, $\frac{\alpha\gamma}{1-\alpha-\gamma+2\alpha\gamma} < \frac{1}{3} \iff \alpha < \frac{1-\gamma}{1+\gamma}$. Supposing $\beta = 20\%$, $N = 5$, then $\gamma \approx 6\%$, and the adversary has to attain $\frac{1-\gamma}{1+\gamma} \approx 89\%$ overall computation power to launch the selfish mining attack.

4 Flexibility of Protocol Upgrade

Algorithm 2 two-phase upgrade process

```

1: procedure UPGRADE
2:   isVote  $\leftarrow$  0
3:   voteBegin  $\leftarrow$  0
4:   expire  $\leftarrow$  3
5: loop:
6:   if keyHeight mod  $W = 0$  and MeetUpgradeRequirement() then
7:     voteBegin  $\leftarrow$  1
8:   end if
9:   if voteBegin = 1 and isVote = 0 and TicketIsSelected() then
10:    VoteForUpgrade()
11:    isVote = 1
12:  end if
13:  if keyHeight mod  $W = 0$  and voteBegin = 1 then
14:    if VoteFailed() then return false
15:    else if VotePassed() then
16:      ActiveUpgradeAfterNextRCI() return false
17:    else
18:      if expire > 0 then
19:        expire  $\leftarrow$  expire - 1
20:        isVote = 0
21:      else return false
22:    end if
23:  end if
24: end if
25: goto loop.
26: end procedure

```

A *hardfork change* is a change to the blockchain protocol that makes previously invalid rules valid, and therefore requires all participants to upgrade. Any alteration to blockchain which changes the blockchain structure (including block hash), difficulty algorithm, voting rules or enlarges the scope of valid transactions is a hardfork change. These hardfork changes are inevitable on the evolution of blockchain ecology. However, it is extremely difficult to implement hardfork changes in blockchain protocol. For example, modification to scale up existing protocol is a raging debate in the Bitcoin community. The reason why hardfork changes are difficult to implement is that stakeholders can not participate fairly in the protocol upgrade events which normally is determined by a small group of powerful parties such as core developers, wealthy participants and influential organizations. If some participants refuse to upgrade, a permanent fork will emerge.

Inspired by DASH and Decred, we introduce a two-phase upgrade process to grant fairness decision-making power to each stakeholder via ticket-voting mechanism on protocol upgrade, acti-

vating the hardfork changes which win the vote. We denote every W keyblock intervals by a Rule Change Interval (RCI). The two-phase upgrade process is described in Algorithm 2.

First Phase. The first phase is to meet the upgrade requirement over the network. After the hardfork code which initially disables new functions is released, a majority of participants need to upgrade firstly. We break down the hardfork changes into two categories: changes of mining and changes of voting. For changes of mining, At least x percent of the last W keyblocks must have the latest block version. For changes of voting, y percent of the votes in the last W keyblocks must have the latest vote version. Once upgrade thresholds are met, the voting is scheduled to begin from the first keyblock of the next RCI.

Second Phase. The second phase is the actual voting. There are at most $W \times N$ votes cast during a single RCI. The final keyblock of the RCI tallies the votes within the RCI, and determines outcomes prior to the next keyblock being mined. Possible outcomes are following:

- If votes fail to meet the Yes (or No) majority threshold (i.e., z percent of votes are Yes (or No)), the voting process keeps on for the next RCI.
- If votes reach the Yes majority threshold, the voting process exits and the hardfork changes will activate after next RCI (the next RCI is set aside for unupgraded users to upgrade).
- If votes reach the No majority threshold, the voting process exits and the hardfork changes will never activate.
- If the voting process never reaches the majority vote threshold in Z rounds of RCI, the voting process expires and the hardfork changes will never activate.

With the help of the two-phase upgrade process, stakeholders fairly participate in the protocol upgrade. Successful hardfork changes, which obtain the majority of votes, smoothly accomplish implementation, while failed changes would naturally be buried. The upgrade for the benefit of the majority achieves the healthy evolution of the blockchain ecology.

5 Protocol Evaluation and Performance Test

Implementation. This scheme has been implemented by Hcash. The source code of Hcash can be found in Github³. We deployed a global network (the *testnet*) to test our code of Hcash. The testnet was maintained for three months, during which we have simulated various possible attacks and a pressure test on this network. Results show that our scheme is practical and robust within all scenarios under our considerations.

The Testnet. The testnet was deployed and maintained from September 29th to December 21st of 2017. The block size was set to be 2MB and keyblocks were generated every 5 minutes. The difficulty of mining a microblock was $\frac{1}{32}$ that of keyblock, i.e., $T_M/T_K = 32$ (except for the pressure test, where the block size and T_M/T_K were variables). The expected volume of the ticket pool was 40960 tickets. Each keyblock was voted by 5 randomly selected tickets, adding at most 20 new tickets into the ticket pool. Each ticket became mature after the generation of 128 new keyblocks.

We deployed 9 nodes as *DNSSeeds* via cloud services provided by *Alibaba* and *Amazon*⁴, located in Beijing, San Francisco, Shanghai, Shenzhen, Sidney, Singapore, and Tokyo, respectively. In particular, 25 nodes were physically deployed in Shanghai to constitute the network. Moreover, during the test period of three months, hundreds of nodes were detected to join and leave the network dynamically from over ten countries worldwide. In another word, the testnet had experienced complex conditions, hence its robustness has been thoroughly tested.

³ <https://github.com/HcashOrg/hcashd>

⁴ <https://aws.amazon.com/>, <https://www.alibabacloud.com/en>

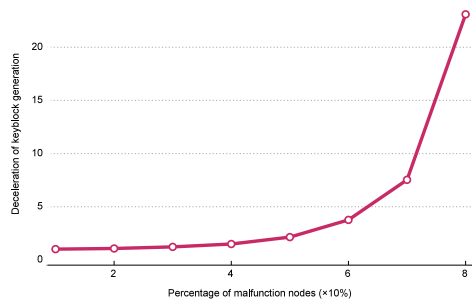


Fig. 4. Deceleration of Chain Growth under Different Percentage of Malfunction Voters

Malfunction of Voters. As described in our protocol, each keyblock is validated by certain voters, each corresponding to one randomly selected element of the ticket pool. In practice, a certain fraction of selected voters might be malfunction nodes, who fail to broadcast its vote due to either a breakdown or malicious purposes. In this case, some keyblock may not be validated by enough votes and hence the growth rate of the chain is reduced. To simulate this, we randomly had certain voters withhold their votes. As a result of our simulations, Fig. 4 shows the deceleration rate of chain growth (the resultant growth rate of keyblocks over the rate without malfunction) varying according to different percentages of malfunction voters. Obviously, such a malfunction affects the chain grow rate to only a minor extent even if 20% voters fall into a malfunction.

Blockchain	Keyblock Interval	Block Size	Microblock Interval	Transaction Size	Throughput (TPS)
Bitcoin	10 min	1 MB ●	–	250 B ○	7 ●
Ethereum	15 s	–	–	–	25 ★
Decred	5 min	0.384 MB ●	–	250 B ○	5 ●
Goshawk	5 min	2 MB ●	18.75 s	250 B ○	270 ★
Goshawk	5 min	8 MB ●	9.38 s	250 B ○	1550 ★

Table 4. Throughput Comparison where values marked by ● stand for upper bounds, ○ for lower bounds, ★ for measurements.

The Pressure Test. We launched a pressure test to measure the scalability of Goshawk. During our test, the expected keyblock interval was constantly 5 minutes along with various block sizes and difficulty ratios. We deployed 28 nodes, of whom 4 took part in the PoS via ticket purchasing and voting, 20 took part in the PoW via mining and 4 kept producing an overloaded amount of transactions. This test proceeded for four days, and the results are compared with Bitcoin, Ethereum and Decred as shown in Tab. 4.

6 Conclusion

Past experience has proven that PoW fits for various permission-less blockchains very well as a powerful distributed agreement protocol, though it still needs to be improved in the aspects of efficiency, fairness, robustness and flexibility. Consequently, many attempts have been made to address or mitigate the issues related to PoW, while the current state of art focuses on the solutions to one or a few parts of the issues and is still far from satisfactory.

In this paper, we proposed Goshawk, the first novel PoW-driven blockchain protocol with high efficiency, strong robustness, as well as good flexibility. Goshawk is actually a hybrid consensus protocol, in which a two-layer chain structure with two-level PoW mining strategy and a ticket-voting mechanism are combined delicately. More specifically, we adopted the two-layer chain structure

(i.e., keyblocks/microblocks) given in Bitcoin-NG, and further improved it by introducing two-level PoW mining strategy (i.e., keyblocks and microblocks with two different mining difficulties, respectively). This guarantees the high throughput of our scheme, while obviating the vulnerability to the attack of microblock swamping in Bitcoin-NG. Furthermore, we borrowed the idea of ticket-voting approach presented in DASH and Decred, and refined this idea by formalizing it into a more rigorous mechanism, then we combined this mechanism with the above chain structure elaborately to attain strong security and good flexibility. Security analysis of our scheme showed that it is incentive-compatibility, and robust against selfish mining and “51%” attack of computation power. Besides, a two-phase upgrade process was introduced to demonstrate good flexibility of our scheme in protocol upgrading. Finally, our scheme offered good stability and promising scalability in the real-world testnet of the public blockchain project Hcash and suggested strong usability in next-generation cryptocurrencies.

References

1. Raiden network. <http://raiden.network/>.
2. Scalability. Bitcoin wiki, 2015. <https://en.bitcoin.it/wiki/Scalability>.
3. Dash official documentation. Dash Core Group, Inc., 2017. <https://docs.dash.org>.
4. Decred documentation. Decred Technology website, 2017. <https://docs.decred.org/>.
5. Cryptocurrency market capitalizations, retrieved Apr.2018. <https://coinmarketcap.com/>.
6. G. Andresen. Bitcoin improvement proposal 101, 2015. <https://github.com/bitcoin/bips/blob/master/bip-0101.mediawiki>.
7. I. Bentov, R. Pass, and E. Shi. Snow white: Provably secure proofs of stake. *IACR Cryptology ePrint Archive*, 919, 2016.
8. H.-H. Buerger. Ethereum White Paper, 2016. <https://github.com/ethereum/wiki/wiki/White-Paper>.
9. A. Churyumov. Byteball: A decentralized system for storage and transfer of value, 2016. <https://byteball.org/Byteball.pdf>.
10. B. David, G. Peter, A. Kiayias, and A. Russell. Ouroboros Praos- An adaptively-secure, semi-synchronous proof-of-stake protocol. *IOHK paper*, 2017.
11. C. Decker and R. Wattenhofer. A fast and scalable payment network with bitcoin duplex micropayment channels. In A. Pelc and A. A. Schwarzmann, editors, *Stabilization, Safety, and Security of Distributed Systems*, pages 3–18, Cham, 2015. Springer International Publishing.
12. C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In E. F. Brickell, editor, *Advances in Cryptology — CRYPTO’ 92*, pages 139–147, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.
13. I. Eyal. The miner’s dilemma. In *Proceedings - IEEE Symposium on Security and Privacy*, volume 2015-July, pages 89–103, 2015.
14. I. Eyal, A. E. Gencer, E. G. Sirer, and R. van Renesse. Bitcoin-NG: A Scalable Blockchain Protocol. In *usenix*, 2015.
15. I. Eyal and E. G. Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8437, pages 436–454, 2014.
16. J. Garay, A. Kiayias, and N. Leonardos. The Bitcoin backbone protocol: Analysis and applications. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9057, pages 281–310, 2015.
17. J. Garzik. Bitcoin improvement proposal 102, 2015. <https://github.com/bitcoin/bips/blob/master/bip-0102.mediawiki>.
18. J. Garzik. Making decentralized economic policy, 2015. <http://gtf.org/garzik/bitcoin/BIP100-blocksizechangeproposal.pdf>.
19. A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun. On the Security and Performance of Proof of Work Blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security - CCS’16*, pages 3–16, 2016.

20. Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles, SOSP '17*, pages 51–68, New York, NY, USA, 2017. ACM.
21. R. Khalil and A. Gervais. Revive: Rebalancing off-blockchain payment networks. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS '17*, pages 439–453, New York, NY, USA, 2017. ACM.
22. A. Kiayias, A. Russell, B. David, and R. Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 10401 LNCS, pages 357–388, 2017.
23. S. King and S. Nadal. PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake. *Ppcoin.Org*, 2012.
24. S. D. Lerner. Dagcoin: A cryptocurrency without blocks, 2015. <https://bitslog.files.wordpress.com/2015/09/dagcoin-v41.pdf>.
25. L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena. A secure sharding protocol for open blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS '16*, pages 17–30, New York, NY, USA, 2016. ACM.
26. A. Miller, I. Bentov, R. Kumaresan, and P. McCorry. Sprites: Payment channels that go faster than lightning. *CoRR*, abs/1702.05812, 2017.
27. S. Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. *Www.Bitcoin.Org*, page 9, 2008.
28. A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder. *Bitcoin and Cryptocurrency Technologies Introduction to the book*. 2016.
29. K. Nayak, S. Kumar, A. Miller, and E. Shi. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In *Proceedings - 2016 IEEE European Symposium on Security and Privacy, EURO S and P 2016*, pages 305–320, 2016.
30. R. Pass and E. Shi. Rethinking large-scale consensus. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 115–129, Aug 2017.
31. M. E. Peck. Adam back says the bitcoin fork is a coup, 2015. <http://spectrum.ieee.org/tech-talk/computing/networks/the-bitcoin-for-is-a-coup>.
32. J. Poon and T. Dryja. The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments. *Technical Report (draft)*, page 59, 2016. <https://lightning.network/lightning-network-paper.pdf>.
33. S. Popov. The tangle, 2016. https://www.iotatoken.com/IOTA_Whitepaper.pdf.
34. QuantumMechanic. Proof of Stake Instead of Proof of Work. *GitHub*, 2011.
35. P. R. Rizun. Subchains: A Technique to Scale Bitcoin and Improve the User Experience. *Ledger*, 1:38–52, 2016.
36. A. Sapirshstein, Y. Sompolinsky, and A. Zohar. Optimal selfish mining strategies in bitcoin. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9603 LNCS, pages 515–532, 2017.
37. Y. Sompolinsky and A. Zohar. Accelerating Bitcoin’s Transaction Processing. Fast Money Grows on Trees, Not Chains. *IACR Cryptology ePrint Archive*, 881:1–31, 2013.
38. TierNolan. Decoupling transactions and pow, 2013. <https://bitcointalk.org/index.php?topic=179598.0>.
39. F. Tschorsch and B. Scheuermann. Bitcoin and beyond: A technical survey on decentralized digital currencies. *IEEE Communications Surveys and Tutorials*, 18(3):2084–2123, 2016.