TwinsCoin: A Cryptocurrency via Proof-of-Work and Proof-of-Stake

Alexander Chepurnoy[*] *Tuyet Duong*[†] *Lei Fan*[‡] *Hong-Sheng Zhou*[§]

Abstract

We design and implement TwinsCoin, the first cryptocurrency based on a provably secure and scalable public blockchain design using both proof-of-work and proof-of-stake mechanisms. Different from the proof-of-work based Bitcoin, our construction uses two types of resources, computing power and coins (i.e., stake). The blockchain in our system is more robust than that in a pure proof-of-work based system; even if the adversary controls the majority of mining power, we can still have the chance to secure the system by relying on honest stake. In contrast, Bitcoin blockchain will be insecure if the adversary controls more than 50% of mining power.

Our design follows a recent provably secure proof-of-work/proof-of-stake hybrid blockchain by Duong et al. (ePrint 2016). In order to make our construction practical, we enhance Duong et al.'s design. In particular, we introduce a new strategy for difficulty adjustment in the hybrid blockchain and provide an analysis of it. We also show how to construct a light client for proof-of-stake cryptocurrencies and evaluate the proposal practically.

We implement our new design. Our implementation uses a recent modular development framework for blockchains, called Scorex. It allows us to change only certain parts of an application leaving other codebase intact. In addition to the blockchain implementation, a testnet is deployed. Source code is publicly available.

[*]IOHK. Email: alex.chepurnoy@iohk.io.

[†]Virginia Commonwealth University. Email: duongtt3@vcu.edu.

[‡]Shanghai Jiao Tong University. Partial work done while visiting the Cryptography Lab at Virginia Commonwealth University. Email: fanlei@sjtu.edu.cn.

[§]Virginia Commonwealth University. Email: hszhou@vcu.edu.

1 Introduction

The emergence of decentralized cryptocurrencies like Bitcoin [40] has the potential to significantly reshape the future of financial transactions and distributed interactions in general, and eventually bring us a much more organized and documented digital world. In the Bitcoin system, a public distributed ledger, called *blockchain*, is maintained by a peer-to-peer network of nodes called Bitcoin *miners* via the proof-of-work (PoW) mechanism [2, 15]. Essentially, the proof-of-work mechanism enables an *open blockchain*, where miners are allowed to join and leave the system at any moment.

Garay et al. [18] and then Pass et al. [44] investigated the blockchain security in the cryptographic setting and have shown that, assuming the *majority of mining power* in the Bitcoin system is controlled by the honest miners, the blockchain indeed satisfies several important security properties (which are critical for blockchain-based applications). In contrast, if the assumption of "honest majority of computing power" does not hold, i.e., the adversary dominates the computing resources in the system, then Bitcoin blockchain will not be trustworthy any more.

In practice, it is not clear if assumption of "honest majority of computing power" always holds. Here are few examples. GHash.io, the largest mining pool at the moment, exceeded 50% of Bitcoin's mining power in 2014 [20]. For now, several top mining pools (e.g., F2Pool, AntPool, BTCC, BW) collectively control about 60% mining power. It is unclear if they could be influenced by certain party or collude. Novel ideas were introduced [38] to discourage the formation of mining pools. Unfortunately, if the adversary controls the majority of mining power, they can still be able to dominate the system (which makes the system not trustworthy). We also remark that blockchain protocols could be vulnerable during the process of splitting. As an example, in Summer 2016 Ethereum network had been split into two forks, Ethereum and Ethereum Classic. In the early days of Ethereum Classic one of the biggest Ethereum miners, Chandler Guo, threatened it with "51% attack" [22]. Similarly, in the face of a possible splitting in Bitcoin, a group of miners supporting Bitcoin Unlimited was threatening "CoreCoin" with the same adversarial majority mining power attack [12]. How can users protect their assets from a majority of mining power which is willing to destroy the system? This leads to our research goal:

Research Goal: *Develop Bitcoin-like open blockchains so that they can be secure even when the adversary controls more than 50% computing power.*

1.1 Our Approach

1.1.1 Singling out a suitable candidate blockchain

The research goal above seems to be impossible to reach without additional resources or assumptions. It is popular in the cryptocurrency community to suggest to use coins (i.e., stake) in order to protect the blockchain. Such method is called *proof-of-stake (PoS)*. In a high level, proof-of-stake requires protocol players to prove ownership of a certain amount of stake in the system. Only those who can provide such a proof can participate in the process of maintaining the blockchain. Naturally, we could have two approaches to achieve the research goal: (i) blockchain via a pure PoS mechanism, and (ii) blockchain via a hybrid (PoW and PoS) mechanisms.

We are aware that it is difficult to give a complete security analysis for a complex, real-world protocol such as Bitcoin. However, the well-received provable security approach still deserves our attention. If a full-fledged security analysis of a blockchain protocol is not feasible at the moment, we could focus on a much simplified "core" of the targeted protocol. Provable security of the simplified core, once established, will help gain significant confidence on the security of the target blockchain. With this in mind, we next make effort to single out a candidate, i.e., a provably secure blockchain core, so that we may be able to "upgrade" the candidate core to the full-fledged blockchain, and eventually achieve our research goal.

Existing provably secure pure proof-of-stake schemes do not scale. Blockchain researchers have made attempts to construct provably secure, scalable blockchains via pure PoS mechanisms; see [6, 31, 36][1]. However, these existing solutions cannot scale to a large number of network nodes in an *open* setting where participants can freely join or leave the system at any time they want.

[1]These are concurrent efforts of our work.

In particular, in these existing provably secure, pure PoS proposals, a majority voting (or the equivalent) is required to enable new participants to join the system. In contrast, without this mechanism, the adversary could corrupt elected leaders at a later point and produce an "alternative" blockchain which could be longer than the blockchain in the system (even if the adversary does not control the majority of stake). Unfortunately, this voting process requires most of existing players to participate in the protocol, which cannot scale to a large scale network. At the moment of writing, it remains unclear if we can construct a provably secure, scalable open blockchain via any pure PoS mechanism.

We here remark that these recent provably secure pure PoS-based protocols are useful for relatively small networks. But they not suitable for (potentially big) open networks because of performance issues.

2-hop blockchain is scalable but not practical yet. Duong et al. [14] design the first provably secure and scalable open blockchain, called "2-hop blockchain", via combining proof-of-work and proof-of-stake. In their design, in addition to miners, a new type of players, stakeholers (stakeholders) are introduced. A winning miner cannot extend the blockchain immediately as in Bitcoin. Instead, the winning miner provides a base which enables a stakeholder to be "selected" to extend the blockchain. That is, a miner and then a stakeholder jointly extend the blockchain with a new block. Note that, in Bitcoin, winning miners directly extend the blockchain.

In the 2-hop blockchain protocol PoW-chains and PoS-chains are plaited together in every time step, and these PoW/PoS-chains are extended alternately. Intuitively, the 2-hop blockchain can be viewed as a proof-of-stake scheme which uses a proof-of-work chain as a *biased* random beacon. This critical idea enables the 2-hop scheme to achieve almost the same efficiency as the original Bitcoin scheme. Also, the 2-hop blockchain can scale to a large size of network.

Unfortunately, Duong et al.'s 2-hop blockchain, as it is, has a significant weakness: the resulting blockchain protocols are expected to be executed in the *static* protocol execution environments. That is, in each round, the invested/involved physical computing resources as well as virtual resources (i.e., stake/coins) in the system are always the same. This does not reflect the reality. Take Bitcoin as an example. In the past years, the amount of computing resources that invested for each unit of time period has increased dramatically. In Bitcoin, Nakamoto creatively introduced the *difficulty adjustment* mechanism to address this issue. By setting a target of extending the blockchain with a new block in each 10 minutes, and by measuring the total time for extending a fixed number (e.g., 2016) of blocks, each miner can be aware if more computing power has been invested. If so, then each miner will increase the difficulty, and vice versa. Unfortunately, adaptive difficulty adjustment mechanism is missing in the 2-hop blockchain. We remark that blockchains without adaptive difficulty adjustment mechanism can often be easily broken. Nevertheless, the provably secure, and scalable 2-hop blockchain can serve as a good starting point for us to achieve our research goal.

1.1.2 From 2-hop blockchain to TwinsCoin

Adaptive difficulty adjustment. In our TwinsCoin system design, we need to make our blockchain protocols to be adaptive to the protocol execution environments. Note that here we have to address *two* types of resources, computing power and stake. It seems that we need to measure the system through two different ways. However, in the original 2-hop design, the total number of proof-of-work blocks (PoW-blocks) is equal to the total number of proof-of-stake blocks (PoS-blocks). It is not clear how to use Nakamoto's difficulty adjustment mechanism directly for our purpose.

In order to address this issue, we change the blockchain structure. Our new blockchain structure allow us to "measure" the system thru two different ways. Now we record more PoW-blocks, called *attempting proof-of-work blocks*; these blocks are generated due to a successful first hop, but a failure second hop. For the sake of presentation, we call PoW-blocks *successful* if they are generated due to a successful first hop along with a successful second hop. Once we have two types of PoW-blocks, we can measure the ratio between these two types of PoW-blocks. This idea eventually allows us to design a new difficulty adjustment mechanism.

Moving the underlying blockchain to the non-flat setting. In the original 2-hop blockchain design [14], the resulting blockchain protocols are expected to be executed in the idealized flat-model. Note that, in the flat model, each proof-of-work miner holds the same amount of computing power and each proof-of-stake holder has the same amount of stake (i.e., coins). Apparently, this flat model does not reflect the reality. Strictly sticking to the flat model will only allow us to design non-practical blockchains.

Blockchain is the core component of our TwinsCoin system. Note that, the designed and then implemented blockchain in this paper essentially consists of two parts, the PoW-chain, and the PoS-chain, and they always go along together, like twins, and we coin our system as *TwinsCoin*.

1.2 Our Contributions

In this paper we provide a clear roadmap for designing provably secure yet scalable blockchain protocols which can be secure even when the adversary controls more than 50% computing power in the system. We prove viability of our design in practice with the help of an actual implementation done. In details, our contributions are as follows:

- Garay et al. [19] extend their previous model [18] by addressing adaptive difficulty adjustment in the Bitcoin system. In our work, we further extend their model by allowing difficulty adjustment for both proof-of-work and proof-of-stake. To the best of our knowledge, there has not been yet a formal definition, design, and analysis for adaptive difficulty for proof-of-stake, or hybrid proof-of-work/proof-of-stake blockchains. Furthermore, before our work, there is no treatment in non-flat setting.

- We propose a way to make light clients possible in a proof-of-stake setting. As far as we know this is the first concrete proposal backed by a practical evaluation.

- We provide experimental results on different characteristics of certain parts of our proposal. In particular, we obtain concrete numbers for a race of adversarial and honest chains as initial estimations on security against adversarial majority of mining power. Some properties of proof-of-stake difficulty adjustment function are also obtained in a different experiment. The light client proposal has been examined in a set of experiments measuring verification time as well as an overhead needed for a simulated blockchain. We also provide a full-fledged implementation of our design in the form of TwinsCoin cryptocurrency. Source code of the implementation is available. More details on the implementation and the experiments can be found in Section 5.

Organization. In Section 2, we present our threat model and assumptions. In Section 3, we recall the most relevant work that TwinsCoin system builds on: Nakamoto's blockchain and 2-hop blockchain. Next, in Section 4, we present the details of our TwinsCoin design, and then in Section 5, we provide the implementation and evaluation details. Future work and related work are provided in Section 6 and in Section 7, respectively. Finally, some supporting materials are provided in Appendix A, and B.

2 Threat Model and Assumptions

We consider a standard multi-player asynchronous communication setting (e.g., Canetti's formulation of the "real world" execution [10]), with the relaxations that *(i)* the underlying communication graph is not fully connected, and *(ii)* the underlying communication channel is reliable but not authenticated. In this setting, an adversary is not allowed to stop the messages from being delivered, but he may "spoof" the source of a message and impersonate the (well-behaved) sender of the message, or to "delay" messages from any participants. Moreover, the adversary is "rushing" in the sense that, in any given round, he is allowed to see all honest (well-behaved) miners' messages before deciding his strategy.

We consider a model which allows both proof-of-work miners and proof-of-stake holders, and the difficulty for proof-of-work and proof-of-stake can be adaptively adjusted. We remark that all previous models (which will be reviewed below) are idealized. In the reality, each different honest miner may have a different amount of computing power or stake. Our protocol is described in a non-flat model where each stakeholder is associated with a different amount of coins.

Previous models. Following the well-received **provable security** approach, Garay et al. [18] provide the first comprehensive security analysis framework for Bitcoin blockchain in the cryptographic setting. To simplify the analysis, Garay et al. consider an idealized "flat-model" where all miners have the same amount of computing power.

Pass et al. [44] recently extend Garay et al's model [18] by considering a more realistic communication network (i.e., partial synchronous network) in which messages from honest players can be delayed with an upper bound units of time. Duong et al. [14] propose a further extended model by considering two types of players, proof-of-work miners and proof-of-stake holders, in blockchain protocols. Very recently, Garay et al. [19], extend their previous model [18] by addressing adaptive difficulty for proof-of-work blockchains. To the best of our knowledge, there has not been yet a formal definition and analysis for adaptive difficulty of proof-of-stake blockchains.

Assumptions. As in [14], we assume that the majority of collective resources (including hashing power and stake) are controlled by well-behaved (honest) players. Furthermore, our system is assumed to be extended from an "already mature blockchain" such as Bitcoin blockchain.

3 Background

3.1 Nakamoto's blockchain

We here briefly review Nakamoto's Bitcoin blockchain [40]. Bitcoin blockchain is based on proof-of-work puzzles [2, 15], which can be abstractly described via the following hash inequality:

$$\mathsf{H}(h_{\mathrm{w}}, w, X) < T$$

where $h_{\mathrm{w}} \in \{0,1\}^{\kappa}$ is the hash of the previous proof-of-work block (κ is a security parameter), w is a suitable solution for this puzzle, X is the record component of the block, and T denotes the current proof-of-work target. See [18] for more details.

Extending the chain. At any point of the protocol execution, each miner attempts to extend the blockchain. More concretely, upon receiving some record X, a miner chooses random $w \in \{0,1\}^{\kappa}$ and checks whether w is a valid solution to the above hash inequality with respect to h_{w}, hash value of the last block in the blockchain; if so, the miner reveals the solution to the system. In Nakamoto's design, multiple miners might find distinct solutions with the same preceding block, in which a blockchain fork will be introduced. To resolve this issue, all well-behaved (honest) miners are expected to follow the longest blockchain in the system.

Security. Garay et al. [18] and Pass et al. [44] have already rigorously analyzed the security of Nakamoto's blockchain in the cryptographic setting. In a nutshell, the Nakamoto's blockchain satisfies certain important security properties such as common prefix, chain quality and chain growth, under the assumption that the majority of computing power is controlled by honest players. Please refer to Appendix A for the definitions of security properties (common prefix, chain quality and chain growth).

3.2 Duong et al's 2-hop blockchain

Nakamoto's blockchain is powered by computing resources, and the blockchain is maintained by only miners. Duong et al. [14] propose 2-hop blockchain — a combination of proof-of-work (PoW) and proof-of-stake

(PoS). There, the blockchain is managed by two types of nodes, called miners and stakeholders (users). Correspondingly, there are two types of chains: proof-of-work chain (PoW-chain), denoted $\mathcal{C}$, and proof-of-stake chain (PoS-chain), denoted $\widetilde{\mathcal{C}}$, in the system. These PoW/PoS chains are securely tied together, as a *chain-pair*. Stakeholders maintain the proof-of-stake chain. On the other hand, miners together manage the proof-of-work chain which is considered as a biased random beacon for choosing stakeholder (to extend the proof-of-stake chain).

Extending the chain-pair. In general, miners (or stakeholders) collect information from the network, perform some validation and generate proof-of-work blocks (PoW-blocks) or proof-of-stake blocks (PoS-blocks), and then share their states to the network.

- *Extending the PoW-chain:* To generate a new PoW-block, each miner first computes the hash $h_w \in \{0,1\}^\kappa$ of the previous PoW-block (the head of the PoW-chain), the hash $h_s \in \{0,1\}^\kappa$ of the head of the PoS-chain. Here, without connecting to the head of the current PoS-chain, the adversary can rewrite any information stored on the chain-pair. The miner then attempts to solve the following hash inequality

  $$\mathsf{H}(h_w, h_s, w) < T$$

  by finding a suitable solution w where T denotes the current proof-of-work target. If he succeeds in finding the proper w, he then generates a new PoW-block which includes the hash values h_w, h_s and the solution w, and shares this PoW-block to other players in the network.
- *Extending the PoS-chain:* In the 2-hop design, each PoW-block is used for selecting new stakeholders (to generate new PoS blocks). More concretely, if there is a new PoW block B in the system, any stakeholder whose verification key vk satisfies the following hash inequality

  $$\widetilde{\mathsf{H}}(B, \mathsf{vk}) < \widetilde{T}$$

  is allowed to generate a new PoS block. That is, the selected stakeholder can use the corresponding signing key sk to sign the new PoS-block. Note that here, $\widetilde{T}$ is the current proof-of-stake target.

Please see Figure 1 for an illustration of 2-hop blockchain.

Resolving Fork. If multiple players (miners or stakeholder) create blocks (PoW-blocks or PoS-blocks) with the same preceding block, the chain is forked into branches, forming a tree. Other players may subsequently add new valid blocks to any of these branches. To resolve forks, the protocol specifies which chain-pair

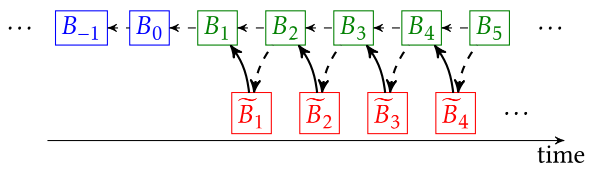

Figure 1: 2-hop blockchain structure
Here, dot arrows denote the first hops, and solid arrows denote the second hops. Green blocks B_i's denote the proof-of-work blocks, and red blocks $\tilde{B}_i$'s denote the corresponding proof-of-stake blocks. Note that 2-hop blockchain is bootstrapped from an existing blockchain and the blue blocks are from the "mature blockchain".

is the best valid chain-pair. The criterion is that the winning chain is the longest valid one.

Security. Duong et al. [14] have already analyzed the security of their 2-hop blockchain in the cryptographic model. Under the assumption that the majority of *collective* resources (computing power and stake) are honest, then important security properties such as common prefix, chain quality and chain growth can be guaranteed. Similar to the analysis in [18, 44], their security analysis is in the flat model.

4 Main Design

Starting with a provably secure "core", the 2-hop blockchain [14], we develop our TwinsCoin system by following the steps below.

- First, we redesign the structure of the 2-hop blockchain in [14] with the goal of enabling a new mechanism for adjusting difficulties. We call this a modified 2-hop blockchain. Please see Section 4.1.
- Second, based on the modified 2-hop blockchain, we introduce a new mechanism to adjust the difficulties for both PoW and PoS chains. Please see Section 4.2.
- In addition, from a different angle, we extend the 2-hop blockchain to the non-flat setting where stakeholders may have different amounts of stake. Please see Section 4.3 for details.
- By combining these above steps, we have the blockchain for our TwinsCoin. This new blockchain is in the non-flat model with adaptive difficulty adjustment.
- Finally, we enhance the TwinsCoin system with light clients. Please see Section 4.5.

4.1 A modified blockchain structure

Besides PoS-blocks and PoW-blocks, our modified blockchain specifies two types of PoW-blocks: *attempting blocks* and *successful blocks* with respect to the local view of each player in the system. If a node receives a new PoW-block without a corresponding PoS-block, the PoW-block becomes an *attempting block* with respect to the view of the player. In opposite, if a node receives a new PoW-block along with a corresponding PoS-block, the PoW-block is now considered as a *successful block* in the node's local view.

We now present the blockchain structure in our modified 2-hop blockchain. Please also see Figure 2.

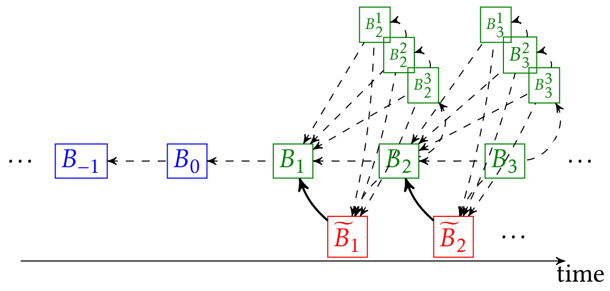

Figure 2: A modified 2-hop blockchain structure
Here, dot arrows (that link to the previous successful block and attempting blocks) denote the first hops, and solid arrows denote the second hops. Green blocks B_i's denote the successful proof-of-work blocks, B_i^j's denote the attempting proof-of-work blocks, and red blocks $\tilde{B}_i$'s denote the corresponding proof-of-stake blocks. Note that the blue blocks are from the "mature blockchain".

Proof-of-Work block structure. Let $\mathsf{H}(\cdot)$ be a cryptographic hash function with output in $\{0,1\}^\kappa$. Now we have two different types of PoW-blocks: attempting blocks and successful blocks. The structure of attempting and successful blocks are the same. However, they have different meaning: (1) no PoS-block refers to an attempting block, and (2) we only count the successful blocks when calculating the real length of a proof-of-work chain. A PoW-block B is a tuple of the form $\langle h_\mathrm{w}, h_\mathrm{s}, h_\mathrm{a}, w\rangle$, where $h_\mathrm{a} \in \{0,1\}^\kappa$ denotes the digest of the latest seen attempting proof-of-work block, $w \in \{0,1\}^\kappa$ is a random nonce, and block B satisfies

$$\mathsf{H}(h_\mathrm{w}, h_\mathrm{s}, h_\mathrm{a}, w) < T$$

where the parameter $T \in \mathbb{N}$ denotes the *current proof-of-work target* of the block.

A PoW-chain $\mathcal{C}$ consists of a sequence of ℓ concatenated successful PoW-blocks. We denote $\mathsf{head}(\mathcal{C})$ as to the *topmost successful PoW-block* B_ℓ in blockchain $\mathcal{C}$. That is, we do not consider an attempting block as the head of the proof-of-work chain, and the head of a PoW-chain is not necessary the topmost PoW-block since the topmost PoW-block could be an attempting block. Moreover, we only link a new proof-of-work block to the head of the proof-of-work chain. This implies that there may multiple attempting blocks attaching to a successful block in our proof-of-work chain. If a proof-of-work blockchain $\mathcal{C}$ is a prefix of another blockchain $\mathcal{C}'$, we write $\mathcal{C} \preceq \mathcal{C}'$.

Extending a proof-of-work blockchain. Since the PoW structure is changed, mechanism to extend a PoW chain is slightly different from 2-hop design. Here, each miner will only mine from the latest *successful* proof-of-work block.

We illustrate the high level idea as in Figure 3. In this figure, we consider a miner attempting to mine from a valid block-pair consisting of a PoW-block B and a PoS-block $\widetilde{B}$. There, in the first attempt, a new PoW-block B_2^1 is linked to B (by the hash h_w of B) and to $\widetilde{B}$ (by the hash h_s of $\widetilde{B}$); however, the corresponding PoS-block of B_2^1 is not seen by the miner. Therefore, we say B_2^1 is an attempting PoW-block. In the second attempt, a new PoW-block B_2^2 is produced without any corresponding PoS-block, this block also becomes an attempting block. The third attempt is a successful one, and a new successful PoW-block with the corresponding PoS-block is attached to the chain.

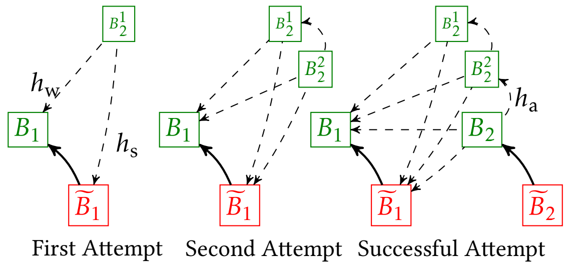

Figure 3: Generating new PoW-blocks

Resolving fork. In 2-hop blockchain design, fork is resolved by choosing the chain-pair with the longest PoW-chain. In our design, a PoW-block could be an attempting or successful block. Furthermore, successful blocks are much more important than attempting blocks. Therefore, the winning chain-pair is the one with the most number of successful blocks, that is, the one that required (in expectancy) the most successful mining power.

Security. We emphasize that, this section only focuses on the structure of the blockchain. We slightly modify

2-hop blockchain to construct a more practical cryptocurrency system. The security proof for this modified version is directly implied from 2-hop blockchain security.

Please refer to Appendix B for more details on our modified 2-hop blockchain.

4.2 Blockchain with adjustable difficulty

4.2.1 Nakamoto's difficulty adjustment for PoW

In Bitcoin, in order to keep the block extension with a steady rate, the system adjusts the PoW hash target adaptively. The intuition is that the lower target means lower probability to get a valid PoW block by calling a hash function. This intuition provides a method to control the block generation rate by a difficulty target adjustment scheme. For some time interval, if the chain extension rate is higher than expected, the difficulty target need to be decreased to make the successful probability lower.

The difficulty target is adjusted every m blocks. In Bitcoin system, we have $\mathsf{m} = 2016$. We define a time period of m blocks (precisely, difference between timestamps of the last and the first blocks in the sequence) as an *epoch*. Let t be the expected time of an epoch. For example, in Bitcoin a new valid block is to be generated every 10 minutes on average. Then we have $\mathsf{t} = \mathsf{m} \times 10$ minutes, which is approximately 14 days for an epoch. Let t_i be the the actual duration of the i-th epoch. Let T_i be the difficulty target in the i-th epoch. We have the difficulty target in the $(i+1)$-th epoch as follows:

$$T_{i+1} = \frac{t_i}{\mathsf{t}} T_i$$

From the equation above we can observe that, if $t_i > \mathsf{t}$ then $T_{i+1} > T_i$ and vice-versa. In the case that $t_i > \mathsf{t}$, the miners spend longer time to obtain m blocks in the i-th epoch. Therefore, the system difficulty target should be increased so that the miners can find new blocks faster in the next epoch. This *negative feedback* mechanism makes the system stable.

Next, in order to increase readability, we first discuss PoS difficulty target adjustment. Then, we will turn to the discussions about PoW difficulty target adjustment.

4.2.2 Our difficulty adjustment for PoW/PoS

For the same consideration as in Bitcoin, we propose two adaptive difficulty target mechanisms for PoW and PoS chains in order to keep the chain-pair growth with a stable rate.

First, we use T to denote the PoW difficulty target in general. We also use T_i as the PoW difficulty target in the i-th epoch. PoW difficulty target is used to control the probability of finding a new valid PoW-block for each attempt. Secondly, we use $\widetilde{T}$ to denote the PoS difficulty target in general. We also use $\widetilde{T}_i$ as the PoS difficulty target in the i-th epoch. PoS difficulty target is used to control the probability that a PoW-block is *successfully* mapped to a valid stakeholder.

PoS difficulty target adjustment. To extend a PoS-chain, a stakeholder uses the inequality $\widetilde{\mathsf{H}}(B, \mathsf{vk}) < \widetilde{T}$ to test if he is qualified to sign a PoS-block. Here, B is a new PoW-block and vk is the verification key of the stakeholder. We assume the expectation of the probability that a PoW-block is successfully mapped to a stakeholder is $\mathsf{R} < 1$. Suppose there are m_i PoW-blocks that are generated in the i-th epoch. Let $\widetilde{T}_i$ be the PoS difficulty in the i-th epoch. The PoS difficulty in the $(i+1)$-th epoch is defined by the following equation:

$$\widetilde{T}_{i+1} = \frac{m_i \mathsf{R}}{\mathsf{m}} \widetilde{T}_i \tag{1}$$

We interpret the PoS difficulty adjustment by the following. If $\frac{m_i}{\mathsf{m}}\mathsf{R} > 1$, then the probability that the PoW-block is mapped to a stakeholder is lower than the expectation R. The PoS target $\widetilde{T}_{i+1}$ will be increased. Similarly, if $\frac{m_i}{\mathsf{m}}\mathsf{R} < 1$, then the probability that the PoW-block is mapped to a stakeholder is higher than the expectation R. The PoS target $\widetilde{T}_{i+1}$ will be decreased.

It is easy to see this is a negative feedback algorithm that can maintain the successful probability close to R. As far as we know this is the first difficulty adjustment procedure for proof-of-stake.

PoW difficulty target adjustment. The difficulty adjustment strategy for PoW-chain in our system is similar to the original Bitcoin system except that we additionally consider the influence of PoS-chain generation to the PoW-chain.

As discussed in Section 4.1, in order to generate a valid block a miner need to try different w to satisfy a hash inequality, i.e., $\mathsf{H}(h_{\mathrm{w}}, h_{\mathrm{s}}, h_{\mathrm{a}}, w) < T$. It is easy to see that if we increase T, the probability to generate a valid PoW-block of one hash query will increase and vice versa. We assume the probability that a PoW-block is successfully mapped to a stakeholder is $\mathsf{R} < 1$. The difficulty is adjusted every m PoW-blocks of the chain-pair. We can take the typical value $\mathsf{m} = 2016$ as in Bitcoin system. We use m_i to denote the total number of attempting (and successful) PoW-blocks that are generated in the i-th epoch. If some PoW-blocks are not be successfully mapped to stakeholders, we would have $m_i > \mathsf{m}$. Similar to Bitcoin, we also use t to denote the expected time span for an epoch, and t_i do denote the actual time span for the i-th epoch.

Let T_i be the difficulty target in the i-th epoch. The difficulty target in the $(i+1)$-th epoch is defined by

$$T_{i+1} = \frac{\mathsf{m} t_i}{m_i \mathsf{t} \mathsf{R}} T_i$$

The difficulty adjustment for PoW is based on two "factors", t_i and m_i. The logic is as follows:

- If $t_i < \mathsf{t}$, we know that the i-th epoch is shorter than expected. In this case, we need to increase T_{i+1} to decrease the PoW difficulty in the $(i+1)$-th epoch. Similarly, if $t_i > \mathsf{t}$, we need to reduce T_{i+1} to increase the PoW difficulty.
- If $m_i > \mathsf{m}/\mathsf{R}$, we know that the probability that PoW-block can be mapped to a stakeholder is lower than the expectation R. In this case, the PoS difficulty (see PoS difficulty adjustment) would be decreased and we need to increase PoW difficulty (by reducing T_{i+1}) to keep the balance between PoS and PoW. Similarly, if $m_i < \mathsf{m}/\mathsf{R}$, the PoS difficulty would be increased, then we need to decrease PoW difficulty (by increasing T_{i+1}).

4.2.3 Security analysis

In this section we provide an informal security analysis to demonstrate that the malicious players are not able to attack the
TwinsCoin system by taking advantage of the difficulty adjustment mechanism.

In [14], the authors give a formal security proof for 2-hop blockchain (without considering difficulty adjustment). There, they assume that the probability that a PoW block is generated in a round is very low. A potential attack is that the malicious players can increase the difficulty target to make it is easier to generate a PoW block. In the remaining of this section, we will prove this attack does not work for our modified 2-hop blockchain, under certain assumption. We note that a formal security proof for our modified 2-hop blockchain with difficulty adjustment in the cryptographic setting is challenging and we leave it for further work.

The intuition is that in order to increase the difficulty target, the malicious players can stop to contribute new PoW-blocks and PoS-blocks from some moment. This will make the block extension rate lower. With our difficulty adjustment algorithm, the difficulty target will increase to speed up the block generation. At some later point, the malicious players will begin to work and sign under the current difficulty target. All the players can generate more blocks with the current difficulty target.

Before our analysis, we will define some notations. We assume all of the computing power can make n hash queries to generate PoW-blocks in an epoch. The ratio of honest computing power is ρ and the ratio of malicious computing power is $1-\rho$. We also assume the total amount of coins is $\hat{n}$ and the honest ratio is $\hat{\rho}$ the malicious ratio is $1-\hat{\rho}$. For simplicity we only discuss the average case.

Lemma 4.1. *Suppose the malicious players stop to contribute the block generation to increase the difficulty target. In the i-th epoch, the system reaches stable target regime. Let t_i be the actual time span of the i-th epoch. Suppose in the $(i+1)$-th epoch, the malicious players try to extend chains with all of resources. Let Let t_{i+1} be the actual time span of the $(i+1)$-th epoch. We have $t_{i+1} = t_i \rho \hat{\rho}$.*

Proof. For the i-th epoch is stable, we have $\hat{T}_i = \hat{T}_{i+1}$. We get $m_i = \frac{\mathsf{m}}{\mathsf{R}}$. The PoW-blocks generation ratio is $\frac{\mathsf{m}}{\mathsf{R}} \frac{1}{t_i}$. For the malicious will begin to generate PoW in the $(i+1)$-th epoch, the PoW-blocks generation ratio will increase to $\frac{\mathsf{m}}{\mathsf{R}} \frac{1}{t_i} \frac{1}{\rho}$.

Suppose there are m_{i+1} PoW-blocks are generated in the $(i+1)$-th epoch with the time t_{i+1}. We have $m_{i+1} = \frac{\mathsf{m}}{\mathsf{R}} \frac{t_{i+1}}{t_i} \frac{1}{\rho}$. By the definition of epoch, m_{i+1} PoW-blocks are mapped to m PoS-blocks by honest and malicious stakeholders together. For the current PoS difficulty target is adjusted for honest stake only in i-th epoch, the ratio that a PoW-block is mapped to a stakeholder is $\mathsf{R}\frac{1}{\hat{\rho}}$. We have $\mathsf{m} = m_{i+1} \mathsf{R} \frac{1}{\hat{\rho}}$. Putting them together, we have $t_{i+1} = t_i \rho \hat{\rho}$. □

From the Lemma 4.1, the ratio that the malicious players can increase is bounded by the factor $\frac{1}{\rho\hat{\rho}}$.

4.3 PoS blockchain in the non-flat model

4.3.1 Moving PoS blockchain from the flat to the non-flat model

The (modified) 2-hop blockchain in section 4.1 is described in the flat model with a pre-fixed difficulty parameter. That is, a stakeholder is qualified to sign and generate a new PoS-block for a PoW block B when his verification key vk satisfies the inequality $\widetilde{\mathsf{H}}(B, \mathsf{vk}) < \widetilde{T}$. Intuitively, if stakeholders have different amounts of stake (coins), they would have different probabilities to be elected. Thus we improve the construction to be suitable for non-flat model that means the stakeholder can keep different amount of stake in an account. Next, we prove that our construction is fair in the non-flat model so a stakeholder will not get any advantages if he splits his stake to multiple accounts.

We describe the non-flat model with hash inequality first. For a stakeholder, vk is his verification key (public key), and sk is the corresponding signing key. To extend PoS-chain, the stakeholder will choose the best chain-pair with the most "successful" mining power. Let $\langle \mathcal{C}, \widetilde{\mathcal{C}} \rangle$ be the best chain-pair, and $\mathcal{C}$ is PoW-chain, $\widetilde{\mathcal{C}}$ is PoS-chain. If the last PoW-block B on chain $\mathcal{C}$ is a new block in which there is no corresponding PoS-block on PoS-chain, then the stakeholder will attempt to generate a new PoS-block as the following: Let $\widetilde{T}$ denote the current difficulty target for the PoS-block generation. We assume the total amount of stake in the whole system is $\hat{n}$, we also assume the length of the output of hash function $\widetilde{\mathsf{H}}(\cdot)$ is κ. Let $p = \frac{\widetilde{T}}{2^\kappa}$, we assume $\hat{n}p < 1$. Let v be the number of coins in the account of a stakeholder with vk. If the inequality

$$\widetilde{\mathsf{H}}(B, \mathsf{vk}) < v\widetilde{T}$$

is satisfied, the stakeholder with the key-pair $(\mathsf{sk}, \mathsf{vk})$ wins a chance to sign the corresponding PoS-block for PoW-block B. As far as we know this is the first concrete non-flat treatment for PoS-chain.

4.3.2 Security analysis

We provide here a security analysis for the non-flat model. The players may take more than 1 coin in an account under non-flat model. We will argue that if a stakeholder puts more than 1 coin in his account, this will not change the probability for PoS-block mapping. Intuitively, from our non-flat construction, if a stakeholder puts more coins in an account, he has a higher probability to be selected; therefore, he does not need to split his coins into multiple accounts.

Lemma 4.2. *Let $p = \frac{\widetilde{T}}{2^\kappa}$ and κ is the length of hash output. Let $\hat{n}$ be the total number of coins. We assume $\hat{n}p < 1$. For any stakeholder with account $(\mathsf{sk}, \mathsf{vk})$, we assume there are v coins in this account, where $v < \hat{n}$. We have $\Pr[\widetilde{\mathsf{H}}(B, \mathsf{vk}) < v\widetilde{T}] = vp$.*

Proof. From the definition we have $v\widetilde{T} = vp2^\kappa$. For $\hat{n}p < 1$ and $v < \hat{n}$, we have $v\widetilde{T} < 2^\kappa$. Since $\widetilde{\mathsf{H}}(B, \mathsf{vk})$ produces the output uniformly in $(0, 2^\kappa)$, we have $\Pr[\widetilde{\mathsf{H}}(B, \mathsf{vk}) < v\widetilde{T}] = vp$. $\square$

From the Lemma 4.2, we have the probability that a stakeholder is selected to generate a PoS-block is proportional to the amount of stake he controls.

- If the stakeholder puts his v coins in one account, for any PoW block B, the probability that he is selected to sign the corresponding PoS block is vp.
- If the stakeholder puts his v coins in v accounts and every account has one coin, for any PoW-block B, the probability that an account is selected to sign the corresponding PoS-block is p. The outputs of hash function $\widetilde{\mathsf{H}}(B, \mathsf{vk})$ are independent for different vk. The total probability that the stakeholder is selected is also vp.

That is, the probability a stakeholder is selected in the non-flat model is equal to the accumulated probability that he distributes the stake to different accounts in the flat-model. For a stakeholder, the probability that he is selected only depends on the total amount of stake (coins) he controls.

4.4 Blockchain design in TwinsCoin

In this subsection, we summarize our blockchain design in TwinsCoin. Starting with the 2-hop blockchain (Section 3.2), we first propose a slightly modified version (Section 4.1) by changing the structure of the proof-of-work chain as well as the format of proof-of-work blocks. Then we improve this modified 2-hop blockchain further, *(i)* by enabling difficulty adjustment for both PoW and PoS chains (see Section 4.2), and *(ii)* by introducing an alternative method to choose stakeholders to extend the PoS chain in the non-flat setting where players may have different amounts of coins. By combining these improvements, we complete the blockchain design in our TwinsCoin system. Please refer to Figure 4 for the roadmap of blockchain design. In next subsection, we will return to the client design in our TwinsCoin.

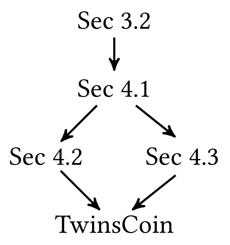
$$\begin{tikzcd}
 & \text{Sec 3.2} \arrow[d] & \\
 & \text{Sec 4.1} \arrow[dl] \arrow[dr] & \\
\text{Sec 4.2} \arrow[dr] & & \text{Sec 4.3} \arrow[dl] \\
 & \text{TwinsCoin} &
\end{tikzcd}$$

Figure 4: Roadmap for blockchain design in TwinsCoin.

4.5 Light client design in TwinsCoin

Checking the validity of a proof-of-work block requires a constant-time operation (i.e., two SHA256 invocations); thus for a blockchain with n blocks, the time to check the validity is linear ($O(n)$). In contrast, in all the proof-of-stake protocols checking a balance of a block generator is needed in order to verify the validity of blockchain. Currently, holding the whole balance sheet is needed for the verification. However, this creates lots of difficulty for light clients. Verification time (if a balance sheet is indexed by a public key) for a block is about $O(\log S)$, where S is a size of the balance sheet. Once balance sheet becomes too big to be stored in random-access memory, performance could be degraded significantly. In Bitcoin, once block size limit is reached, size of a balance sheet (more precisely, unspent outputs set) is growing roughly linearly with time (a corresponding graph could be found at [9]), thus verification time for n PoS blocks is $O(n \cdot \log n)$. The concerns of heavy validation could be the solid arguments against switching from a Proof-of-Work chain to the hybrid one.

As a solution, we propose to authenticate the balance sheet as 2-party dynamic authenticated (public key $\rightarrow$ balance) dictionary. In the paper [45] an authenticated data structure of this kind is proposed to be used in order to avoid holding all the state for the full nodes. We are applying the principle further in order to make light clients feasible in a Proof-of-Stake environment.

A root value after processing the transactions in a block is to be included in a blockheader of a PoS-block. Also, a stakeholder generating a block is including an authenticating path for his output against a root of a previous PoS-block. However, verification time for a block remains the $O(\log S)$, with $O(n \cdot \log n)$ for a chain, but a constant factor would be much smaller. We back the claim with an experiment provided in Section 5.2.2. It is not needed to hold the whole state in order to check whether a PoS-block was generated in a valid way, as by using a 2-party authenticated dynamic dictionary it becomes possible to check proofs and get a new root value without holding the whole dataset. As a drawback, block size would be increased by $O(\log S)$ bytes, we provide concrete numbers in Section 5.2.2.

5 TwinsCoin System

We provide a full-fledged implementation of TwinsCoin. Details on the implementation are provided in Section 5.1. We run several experiments on particular aspects of our design in order to empirically evaluate the claims made throughout the paper and also study some aspects of proof-of-stake difficulty readjustment function in Section 5.2. We also run fully functional TwinsCoin nodes over a testing network which is described in Section 5.3.

5.1 Implementation

We implement TwinsCoin using the Scorex framework [25] in Scala language. Our implementation is full-fledged. Therefore, it is possible to run the testing network without any code modifications. Our implementation is opensourced [1] and published under public domain CC0 license.

There are a few open source modular blockchain development tools available, such as Scorex [25], Sawtooth Lake [26], and Fabric [24]. We choose to use Scorex 2.0 [25] because this is the only existing tool which supports two (or more) types of blocks.

The idea of a modular design for a cryptocurrency was first proposed by Goodman in Tezos whitepaper [21]. The whitepaper (Section 2 of it) proposes to break a cryptocurrency design into three protocols: network, transaction and consensus. In many cases, however, these layers are tightly coupled and it is hard to describe them separately. For example, in a proof-of-stake cryptocurrency a balance sheet structure, which is heavily influenced by a transaction structure, is used in a consensus protocol. To split the layers clearly, Scorex 2.0 has finer granularity. In particular, in order to support hybrid blockchains as well as more complicated structures than a chain (such as SPECTRE [48]), Scorex 2.0 does not even have a notion of the blockchain as a core abstraction. Instead, it provides an abstract interface to a *history* which contains *persistent modifiers*. The history is a part of a *node view*, which is a quadruple of $\langle$*history*, *minimal state*, *vault*, *memory pool*$\rangle$. The minimal state is a data structure and a corresponding interface providing an ability to check a validity of an arbitrary transaction for the current moment of time with the same result for all the nodes in the network having the same history. The minimal state is to be obtained deterministically from an inital pre-historical state and the history. The vault is the node-specific information, for example, a node user's wallet. The memory pool holds unconfirmed transactions being propagated across the networks by nodes before got into blocks.

The whole node view quadruple is to be changed atomically by applying whether a persistent node view modifier or an unconfirmed transaction. Scorex provides guarantees of atomicity and consistency for the application while a coin developer needs to provide implementations for the abstract parts of the quadruple as well as a family of persistent modifiers. Our implementation is introducing two kinds of persistent modifiers, PoW-Blocks and PoS-Blocks.

Our implementation has simpler transactions than Bitcoin [55]: while a TwinsCoin transaction has multiple inputs and outputs, like in Bitcoin, an output contains only a public key of a spender and a value (so no support for Bitcoin Script [53] or another authentication language is provided). To spend an output, one needs to sign its bytes in a referring input. In order to prevent replay attacks, we also associate an output with an unique nonce value, which is a result of *hash(all the transaction bytes without nonces || output index in the transaction)*. Like in Bitcoin reference implementation, the minimal state in the TwinsCoin is the current unspent outputs set (UTXO [7] set). In both the systems, with the UTXO set it is possible to decide whether an arbitrary transaction valid against it or not. By processing a block, a node is removing outputs spent in the block from the set and put there newly created unspent outputs.

We also have implemented block generation functionality directly inside the node software. Iteration over nonce space in Proof-of-Work mining component is artificially limited in order to reduce the load of an evaluation environment and model non-flat mining networks easily. Thus, a number of hash function calls per second is to be set explicitly in code. As in Bitcoin, a proof-of-work function is about to find a hash value with a certain property of a block header with a nonce field included. We use Blake2b hash function with 256 bits output to have 128-bit security level. In our implementation miners start to work on a next attempting block right after previous one seen and before corresponding PoS-block arrives. Thus the mining component working all the time except PoS-block processing phases.

Rollbacks are possible in a blockchain system if a better fork found. We store all the blocks ever got from the network (Bitcoin does the same), so an implementation of the history interface is just switching a pointer to a new best chain in case of a fork. For the minimal state (the UTXO set) as well as for the wallet we need to restore an old version of possibly big dataset before applying blocks from a new best chain after a common one. To simplify previous database snapshot restoring, we are using a versioned key-value database engine IODB [33]. IODB has been built to be used in blockchain systems, so it provides batch updates only and a rollback to an arbitrary snapshot in the past within depth to be set during database creation.

We are reusing peer-to-peer network from the Scorex without any changes. Nodes in the network send announces about their blocks and transactions with INV messages like in Bitcoin [8]. A new block is announced with the same mechanism; thus, propagation time for miners is worse than in Bitcoin network where miners have direct low-latency links to each other and push a header of a new block immediately.

Our implementation is compact, just about 2,300 lines of code, thanks to the frameworks used and concise Scala language.

5.2 Experiments

In this section, we investigate some aspects of the TwinsCoin proposal with the help of targeted experiments. First experiment is about a competition of two chains, an honest and an adversarial, similar to described in the original Nakamoto paper ([40], Section 11). The goal of a second experiment is to measure efficiency of the light client proposal from the Section 4.5. Third experiment examines proof-of-stake difficulty readjustment procedure in a simulated environment.

5.2.1 Chain race experiment

In the original Bitcoin paper [40], Nakamoto evaluated his proposal by considering competition of two chains, one of an adversary and another of an honest miner showing that the adversarial chain cannot overtake the honest one until it is backed by majority of mining power. As there are two resources providing a possibility to generate a block in our proposal, we simulate an adversary possessing different amounts of total hashing power and also total stake.

In our experiment, an adversarial and an honest parties work on separate TwinsCoin instances generating chain-pairs of length 10 (so 10 PoW-blocks and also 10 PoS-blocks). A party which generates its chain-pair first wins the race. The honest party owns 100 outputs locking the same amount of money while the adversary has only some fraction of that. Difficulties are static during the experiment, time to generate a PoW-block on average is overwhelmingly big in comparison with time needed to process a block, and proof-of-stake difficulty is set to have about 1 output chosen on average for the honest party. The code could be found in the file *PrivateChain.scala*.

The result is presented in Figure 5. We run every experiment 20 times and consider that the adversary succeeds if he wins at least once. Gray area in the figure shows adversarial success.

The results show that even with 70% of total mining power, the adversary also needs for about 20% of total stake to generate a better chain than the honest party's. Given Bitcoin capitalization of $20 billion at the moment of writing, 20% of stake is about $4 billion. However, not all the stake is online so security projected into money would be about lower level. It is hard to define precisely how much stake is online in Bitcoin, and how much it would be in case of PoS rewards being granted for

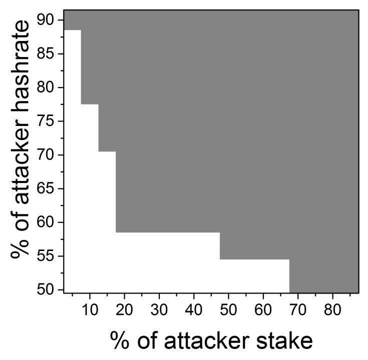

Figure 5: Influence of attacker's stake to his hashrate.

that. Also, we have observed only the simplest kind of attack in this experiment. The adversary can do better, for example, by exploiting network-level protocol with Eclipse attacks [23]. Nevertheless, a malicious miner needs to spend a lot of money to overtake the honest parties.

5.2.2 Light validation experiment

We estimate practically how efficient is the light validation procedure proposed in the Section 4.5. For that, we compare two chain validators. A *full validator* is operating with full state residing in a persistent key-value database. A *light validator* is checking lookup proofs from blockheaders. Both validators are performing lookup operations only.

For the experiment, we use authenticated AVL+ trees from [45]. The full validator code is using disk-based database MvStore with 128 MB in-memory LRU cache. The experiment is started with a balance sheet of about 46 million (public key → balance) pairs (where a public key is about 32 bytes, a value is about 8 bytes). We then try bigger sheets, up to 92 million pairs in size. Thus the testing dataset starts from a size like Bitcoin UTXO set of today [9] and finishes with twice of that size. We use a machine with i7 processor, 16 GB RAM and HDD disk (5400 RPM) for the experiment.

Figure 6 shows running times for both the validators. The results show that our light validator is running in effective constant time negligible to the running time of the full validator (the running time of the light validator is about 20-25 microseconds per operation).

For the light validator, a proof of a block generator's balance is to be included into a block. We obtain proof sizes for different sizes of balance sheet, they are provided

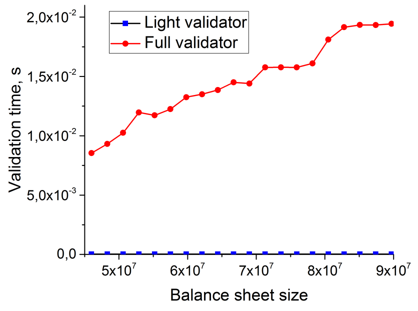

Figure 6: Balance lookup time.

in the Figure 7. The proof size for a Bitcoin-like balance sheet size (46M elements) is about 960 bytes and remains no more than kilobyte when balance sheet is about 92M elements.

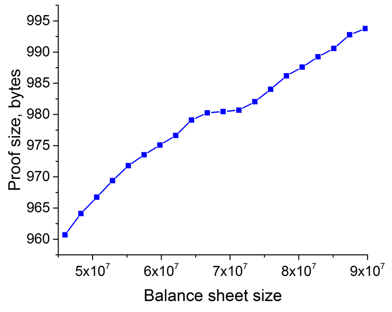

Figure 7: Balance lookup proof size.

5.2.3 Proof-of-stake difficulty experiments

In the Proof-of-Stake difficulty readjustment formula (see Formula 1) we use the R parameter to show the probability that a PoW-block is a successful block. In other words, R indicates the probability that a PoW-block is *successfully mapped* to a stakeholder. Also, $1 - R$ value shows how many attempting blocks an average successful proof-of-work block has included. However, in a distributed environment, real values could be different from the planned ones because of propagation effects. To know the difference we made an experiment where proof-of-stake block was sent to proof-of-work miner not immediately but with a random delay distributed uniformly from 0 to 5% of target generation time. We measure number of attempting blocks included into a successful blocks. Results provided in the Figure 8 show that in this scenario experimental numbers are close to the planned ones.

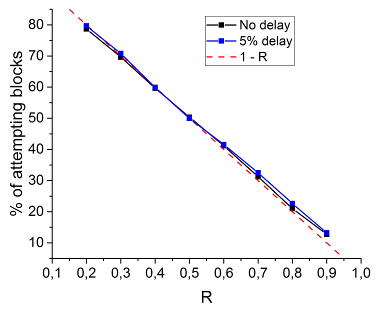

Figure 8: Percentage of Attempting Blocks

We found that, given a constant stake, target value is changing with number of stakeholders growing. The dependency is presented in the Figure 9.

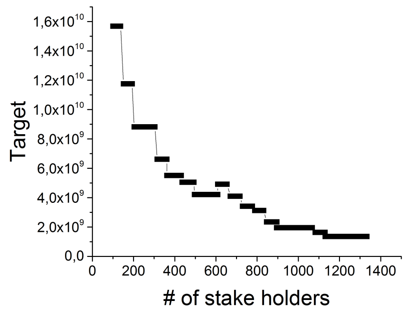

Figure 9: $\tilde{T}$ Value

5.3 Testnet

We launch TwinsCoin full-fledged implementation over a publicly accessible testing network (so-called "testnet" in cryptocurrency jargon). For that, we deploy TwinsCoin to tens of machines, including AWS (Amazon) instances in the US, Europe, Asia, as well as physical machines in Germany and Russia. Each node in the network is connected to 10 random neighbors. Every machine is participating in proof-of-work blocks mining, few machines have public keys with stake on-board so generating proof-of-stake blocks also. Target time between proof-of-work blocks is 10 minutes and R = 0.8.

User interface. Scorex generates a user interface (UI) automatically and few requests regarding common functionality are available with no any efforts needed. In order to add specific functionality, a coin developer needs to specify handlers for the additional requests. The interface is available in a web browser. With the help of the UI, a TwinsCoin user is able to see the hybrid blockchain contents, network peers and their statuses, wallet public keys and corresponding balances. It is also possible to create a transaction (to send tokens) via the user interface.

Monetary policy and incentives. The currency in the network has no emission, so all the coins are issued in the initial state. Proof-of-Stake block generators are getting transaction fees as rewards while Proof-of-Work miners are getting nothing (as finding a proper incentives structure is left for further research).

6 Future Work

There are still some issues to resolve before launching a production-ready system. We highlight the most important ones below.

Incentives. How to properly incentivize participants in a cryptocurrency is a non-trivial task. As an example, in April 2016, Lerner reported [35] about *uncle mining* issue regarding improper rewards for uncle blocks in the GHOST [49] component found in the Ethereum protocol. In TwinsCoin, rewards should be given to stakeholders generating PoS-blocks and also to generators of successful as well as attempting PoW-blocks. Finding a proper balance is left for further research.

Switching. Changing rules in a cryptocurrency is a hard topic. There are two ways, *softfork* and *hardfork*, to upgrade the design recognized in the community. A softfork [54] is such a protocol upgrade that only previously valid rules are made invalid. Nodes not being updated will recognize the new blocks as valid, thus a softfork is backward-compatible. To activate a softfork a majority of mining power is required. A hardfork [52] is such a protocol upgrade that makes previously invalid rules valid. A hardfork requires all the users to upgrade.

If swapping from a PoW-chain to the hybrid solution is not hard-coded since a launch of a coin, upgrade would be non-trivial and probably requires hardfork, but hardforks are better to be avoided in a mature system. We leave identifying suitable switching mechanisms for further research.

7 Related Work

Closely related work. Combining proof-of-work and proof-of-stake has been studied in [4, 5, 13, 32]. Unfortunately, no rigorous security analysis has ever been provided for these proposals. Duong et al [14] provide the first provably secure, proof-of-work and proof-of-stake hybrid blockchain protocol. In addition, Duong et al's proposal is easy to implement and does not involve a trusted entity.

Independent work on secure pure proof-of-stake. Several recent and concurrent efforts (e.g., [6,31,36]) have made to construct provably secure, scalable blockchains via pure PoS. Unfortunately, these solutions cannot scale to a large number of network nodes (e.g., billion nodes) in an *open* setting where participants can freely join or leave the system at any time they want.

Bitcoin and provable security. Anonymous digital currency was introduced by Chaum [11]. However, the first decentralized and practical currency system, Bitcoin [40], was developed many years later, using the so-called proofs-of-work puzzles [2, 15]. Recently, the security of Bitcoin system has been analyzed in the rational setting, e.g., [16, 17, 28, 41, 46, 47], and in cryptographic setting [14, 18, 19, 29, 30, 44, 49].

Cryptocurrencies via alternative physical resources. Alternative consensus techniques via different physical resources (e.g., space/memory) have been investigated to replace computing power. For instance, the physical storage resource is used in [37,43]. Interestingly, hybrid proof system utilizing both computational and space resources—*proofs of space time*—has been introduced in [39]. Intel proposes the use of trusted hardware for blockchain protocols in [27].

Additional cryptocurrencies via virtual resources. We review proof-of-stake (PoS) based blockchain. Since the idea in an online forum [50], several pure PoS schemes that have been proposed and/or implemented in real world including [3, 34, 42, 51]. As mentioned above, in [4, 5, 13, 32], the proof of work and proof of stake can be combined together. However, no rigorous security analysis has ever been provided for these proposals. Fortunately, proof-of-stake related, provably secure blockchains have been developed very recently; please see [6, 31, 36] and [14].

Acknowledgements

We would like to thank Dmitry Meshkov for helping with coding and testing the system.

References

[1] Twinscoin source code. https://bitbucket.org/TwCoin/twinscoin.

[2] A. Back. Hashcash — A denial of service counter-measure. 2002. http://hashcash.org/papers/hashcash.pdf.

[3] I. Bentov, A. Gabizon, and A. Mizrahi. Cryptocurrencies without proof of work. In *3rd Workshop on Bitcoin and Blockchain Research - Financial Cryptography*, 2016.

[4] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld. Proof of activity: Extending Bitcoin's proof of work via proof of stake. Cryptology ePrint Archive, Report 2014/452, 2014. http://eprint.iacr.org/2014/452.

[5] 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):34–37, Dec. 2014.

[6] I. Bentov, R. Pass, and E. Shi. Snow white: Provably secure proofs of stake. In *Cryptology ePrint Archive, Report 2016/919*, 2016. http://eprint.iacr.org/2016/919.

[7] Bitcoin Developer Guide. UTXO definition. https://bitcoin.org/en/developer-guide#term-utxo.

[8] Bitcoin Wiki. Protocol documentation. https://en.bitcoin.it/wiki/Protocol_documentation.

[9] Blockchain.info. Number of Unspent Transaction Outputs. 2017. https://bitcoin.org/en/developer-guide#term-utxo.

[10] R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[11] D. Chaum. Blind signatures for untraceable payments. In D. Chaum, R. L. Rivest, and A. T. Sherman, editors, *CRYPTO'82*, pages 199–203. Plenum Press, New York, USA, 1982.

[12] CryptocoinsNews. Bitcoin Market Needs Big Blocks, Says Founder of BTC.TOP Mining Pool. 2017. https://t.co/fS5sy7jpPD.

[13] CryptoManiac. Proof of stake. *NovaCoin wiki*, 2014. https://github.com/novacoin-project/novacoin/wiki/Proof-of-stake.

[14] T. Duong, L. Fan, and H.-S. Zhou. 2-hop blockchain: Combining proof-of-work and proof-of-stake securely. In *Cryptology ePrint Archive, Report 2016/716*, 2016. https://eprint.iacr.org/2016/716.

[15] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In E. F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 139–147. Springer, Heidelberg, Aug. 1993.

[16] I. Eyal. The miner's dilemma. In *2015 IEEE Symposium on Security and Privacy*, pages 89–103. IEEE Computer Society Press, May 2015.

[17] I. Eyal and E. G. Sirer. Majority is not enough: Bitcoin mining is vulnerable. In N. Christin and R. Safavi-Naini, editors, *FC 2014*, volume 8437 of *LNCS*, pages 436–454. Springer, Heidelberg, Mar. 2014.

[18] J. A. Garay, A. Kiayias, and N. Leonardos. The bitcoin backbone protocol: Analysis and applications. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 281–310. Springer, Heidelberg, Apr. 2015.

[19] J. A. Garay, A. Kiayias, and N. Leonardos. The bitcoin backbone protocol with chains of variable difficulty. In *Cryptology ePrint Archive, Report 2016/1048*, 2016. https://eprint.iacr.org/2016/1048.

[20] D. Goodin. Bitcoin security guarantee shattered by anonymous miner with 51% network power. 2014. http://arstechnica.com/.

[21] L. M. Goodman. Tezos: A self-amending crypto-ledger. position paper.

[22] C. Guo. "I am Chandler Guo, a 51% attack on Ethereum Classic (ETC) is coming with my 98G hashrate". 2016. https://twitter.com/chandlerguo/status/757191880740184064.

[23] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg. Eclipse attacks on bitcoin's peer-to-peer network. In *USENIX Security*, pages 129–144, 2015.

[24] IBM Corp. Hyperledger-Fabric. 2016. https://github.com/hyperledger/fabric.

[25] Input Output Hong Kong. The Scorex Project. 2016. https://github.com/input-output-hk/Scorex.

[26] Intel. Hyperledger-Sawtooth Lake. 2016. https://github.com/hyperledger/sawtooth-core.

[27] Intel. Proof of elapsed time (PoET). 2016. https://intelledger.github.io/introduction.html.

[28] A. Kiayias, E. Koutsoupias, M. Kyropoulou, and Y. Tselekounis. Blockchain mining games. In *Proceedings of the 2016 ACM Conference on Economics and Computation (EC)*, pages 365–382, 2016.

[29] A. Kiayias and G. Panagiotakos. Speed-security tradeoffs in blockchain protocols. Cryptology ePrint Archive, Report 2015/1019, 2015. http://eprint.iacr.org/2015/1019.

[30] A. Kiayias and G. Panagiotakos. On trees, chains and fast transactions in the blockchain. Cryptology ePrint Archive, Report 2016/545, 2016. http://eprint.iacr.org/2016/545.

[31] A. Kiayias, A. Russell, B. David, and R. Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Cryptology ePrint Archive, Report 2016/889*, 2016. http://eprint.iacr.org/2016/889.

[32] S. King and S. Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. 2012. https://peercoin.net/assets/paper/peercoin-paper.pdf.

[33] J. Kotek. IODB storage engine. https://iohk.io/blog/scorex/iodb-storage-engine/.

[34] J. Kwon. Tendermint: Consensus without mining. 2014. http://tendermint.com/docs/tendermint.pdf.

[35] S. Lerner. Uncle mining, an ethereum consensus protocol flaw. 2016. https://t.co/YnnkzROp02.

[36] S. Micali. ALGORAND: the efficient and democratic ledger. *CoRR*, abs/1607.01341, 2016.

[37] A. Miller, A. Juels, E. Shi, B. Parno, and J. Katz. Permacoin: Repurposing bitcoin work for data preservation. In *2014 IEEE Symposium on Security and Privacy*, pages 475–490. IEEE Computer Society Press, May 2014.

[38] A. Miller, A. E. Kosba, J. Katz, and E. Shi. Nonoutsourceable scratch-off puzzles to discourage bitcoin mining coalitions. In I. Ray, N. Li, and C. Kruegel:, editors, *ACM CCS 15*, pages 680–691. ACM Press, Oct. 2015.

[39] T. Moran and I. Orlov. Proofs of space-time and rational proofs of storage. Cryptology ePrint Archive, Report 2016/035, 2016. http://eprint.iacr.org/2016/035.

[40] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008. https://bitcoin.org/bitcoin.pdf.

[41] K. Nayak, S. Kumar, A. Miller, and E. Shi. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. Cryptology ePrint Archive, Report 2015/796, 2015. http://eprint.iacr.org/2015/796.

[42] NXT Community. Nxt whitepaper. 2014. https://www.dropbox.com/s/cbuwrorf672c0yy/NxtWhitepaper_v122_rev4.pdf.

[43] S. Park, K. Pietrzak, A. Kwon, J. Alwen, G. Fuchsbauer, and P. Gaži. Spacemint: A cryptocurrency based on proofs of space. Cryptology ePrint Archive, Report 2015/528, 2015. http://eprint.iacr.org/2015/528.

[44] R. Pass, L. Seeman, and A. Shelat. Analysis of the blockchain protocol in asynchronous networks. Cryptology ePrint Archive, Report 2016/454, 2016. http://eprint.iacr.org/2016/454.

[45] L. Reyzin, D. Meshkov, A. Chepurnoy, and S. Ivanov. Improving authenticated dynamic dictionaries, with applications to cryptocurrencies. 2016. http://eprint.iacr.org/2016/994.

[46] A. Sapirstein, Y. Sompolinsky, and A. Zohar. Optimal selfish mining strategies in bitcoin. In *Financial Crypto*, 2016.

[47] O. Schrijvers, J. Bonneau, D. Boneh, and T. Roughgarden. Incentive compatibility of bitcoin mining pool reward functions. In *Financial Crypto*, 2016.

[48] Y. Sompolinsky, Y. Lewenberg, and A. Zohar. SPECTRE: A fast and scalable cryptocurrency protocol. In *IACR Cryptology ePrint Archive*, 2016. `http://eprint.iacr.org/2016/1159`.

[49] Y. Sompolinsky and A. Zohar. Secure high-rate transaction processing in bitcoin. In R. Böhme and T. Okamoto, editors, *FC 2015*, volume 8975 of *LNCS*, pages 507–527. Springer, Heidelberg, Jan. 2015.

[50] User QuantumMechanic. Proof of stake instead of proof of work. 2011. `https://bitcointalk.org/index.php?topic=27787.0`.

[51] P. Vasin. Blackcoin's proof-of-stake protocol v2. 2014. `http://blackcoin.co/blackcoin-pos-protocol-v2-whitepaper.pdf`.

[52] B. Wiki. Hard fork. `https://en.bitcoin.it/wiki/Hardfork`.

[53] B. Wiki. Script. `https://en.bitcoin.it/wiki/Script`.

[54] B. Wiki. Softfork. `https://en.bitcoin.it/wiki/Softfork`.

[55] B. Wiki. Transaction. `https://en.bitcoin.it/wiki/Transaction`.

A Security properties for blockchains

To capture the security of blockchain protocols, several fundamental security properties for them, namely *common prefix property* [18, 44], *chain quality property* [18] and *chain growth property* [29], have been defined.

Definition A.1 (Chain Growth Property)**.** *The chain growth property* $\mathcal{Q}_{cg}$ *states that for any honest player* P *with the local chain* $\mathcal{C}$ *at round* r *and* $\mathcal{C}'$ *at round* r' *where* $s = r' - r > 0$*, in the execution of the blockchain protocol* Π*. It holds that* $\mathsf{len}(\mathcal{C}') - \mathsf{len}(\mathcal{C}) \geq g \cdot s$ *where* g *is the growth rate and* $\mathsf{len}(\mathcal{C})$ *denotes the number of blocks on the chain.*

Definition A.2 (Common Prefix Property for PoS-chain)**.** *The common prefix property* $\mathcal{Q}_{cp}$ *with parameter* $\kappa \in \mathbb{N}$ *states that for any two honest players* P_1 *at round* r_1 *and* P_2 *at round* r_2*, where* $r_1 \leq r_2$*, with the local chains* $\mathcal{C}_1$*,* $\mathcal{C}_2$*, respectively, in the execution of the blockchain protocol* Π*, it holds that* $\mathcal{C}_1[1, \ell_1] \preceq \mathcal{C}_2$ *and* $\ell_1 = \mathsf{len}(\mathcal{C}_1) - \Theta(\kappa)$ *where* $\mathsf{len}(\mathcal{C})$ *denotes the number of blocks on the chain.*

Definition A.3 (Chain Quality Property)**.** *The chain quality property* $\mathcal{Q}_{cq}$ *with parameters* $\mu \in \mathbb{R}$ *and* $\ell \in \mathbb{N}$ *states that for any honest player* P *with chain* $\mathcal{C}$ *in the execution of the blockchain protocol* Π*, it holds that for large enough* ℓ *consecutive blocks of* $\mathcal{C}$ *the ratio of honest blocks is at least* μ *for* $\mu \in (0, 1)$*.*

B Our modified 2-hop blockchain

In Section 4.1, we describe the modified version of 2-hop blockchain. Here, we provide more details.

As in the 2-hop blockchain design [14], we here also have two tightly coupled blockchains: *proof-of-work blockchain* (PoW-chain) and *proof-of-stake blockchain* (PoS-chain). A PoW-chain is defined as a sequence of temporally ordered PoW-blocks; however, our PoW-chain is different from one in the 2-hop design as there are two different types of PoW-blocks in our system: *attempting PoW-blocks* and *successful PoW-blocks* (see Section B.1 for more details). *Proof-of-Stake blockchain* consists of PoS-blocks and maintained by a set of *stakeholders*.

We summarize the frequently used notations in Table 1.

Table 1: Table of notations

Notation	Description
κ	security parameter.
$\mathcal{C}_{\mathsf{init}}$	an initial blockchain.
$B, \mathcal{C}$	a PoW-block, a PoW-chain.
$\widetilde{B}, \widetilde{\mathcal{C}}$	a PoS-block, a PoS-chain
$\mathcal{B}$	a set of proof-of-work blocks
$\langle \mathcal{C}, \widetilde{\mathcal{C}} \rangle$	a chain-pair consisting of PoW-chain $\mathcal{C}$ and PoS-chain $\widetilde{\mathcal{C}}$.
$\mathbb{C}$	a set of chain-pairs.
X	information stored in blockchain.
Σ	digital signature scheme $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$.
$(\mathsf{sk}, \mathsf{vk})$	a signing and verification key-pair where $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^\kappa)$.
$\textsc{Broadcast}(\cdot)$	unauthenticated send-to-all functionality.

We suppose that there exists an initial blockchain $\mathcal{C}_{\mathsf{init}}$ as the starting point, and $\mathcal{C}_{\mathsf{init}}$ is known to all participants in the TwinsCoin system.

Now we present the behaviors of miners and stakeholders in our system. In general, miners and stakeholders collect blockchain information from the broadcast channel, perform some validation and generating blocks, and then share their states with the network through broadcast. We now explicitly describe the two procedures for miners and stakeholders as follows.

Miners. Initially, each miner sets the set of chain-pairs $\mathbb{C}$ to empty (i.e., ϵ). A miner then continues execution as follows. First, the player copies all chain-pairs received from the network into his local state. The miner then selects the best chain-pair in his view by calling the BestValid algorithm (see Algorithm 6) over the set of chain-pairs. Upon deriving the best chain-pair $\langle \mathcal{C}, \widetilde{\mathcal{C}} \rangle$ (based on length and validity), the miner calls PoW algorithm (see Algorithm 3) to make a bounded number of hash queries in an attempt to find a proof of work solution and extend the PoW-chain $\mathcal{C}$. If it is successful, then his local best chain-pair is extended by one block, the PoW-chain in the pair is updated, and then the best chain-pair is broadcast to the network. This procedure is formally presented by Algorithm 1.

Algorithm 1: Main Protocol: miner.

```
$\mathbb{C} \leftarrow \epsilon$;
while True do
    $\mathbb{C} \leftarrow$ all chain-pairs received from the network.
    $\langle \mathcal{C}, \widetilde{\mathcal{C}} \rangle \leftarrow \mathsf{BestValid}(\mathbb{C})$
    $\mathcal{C}_{\text{new}}, \leftarrow \mathsf{PoW}(\langle \mathcal{C}, \widetilde{\mathcal{C}} \rangle)$
    if $\mathcal{C} \neq \mathcal{C}_{\text{new}}$ then
        $\mathcal{C} \leftarrow \mathcal{C}_{\text{new}}$
        $\mathbb{C} \leftarrow \mathbb{C} \cup \langle \mathcal{C}, \widetilde{\mathcal{C}} \rangle$
        BROADCAST($\langle \mathcal{C}, \widetilde{\mathcal{C}} \rangle$);
```

Stakeholders. The initialization of every stakeholder is the same as miners except that each valid stakeholder is parameterized by a unique pair of signing and verification keys $(\mathsf{sk}_j, \mathsf{vk}_j)$ and amount of coins v_j. (In this appendix, the blockchain protocol is described in the flat model, and we assume $v_j = 1$.)

Consider a stakeholder, in each time step, any chain-pairs and payload X from the network are copied into the stakeholder's local state. The stakeholder then selects the best chain-pair on his view through the BestValid algorithm (see Algorithm 6) over a set of chain-pairs $\mathbb{C}$. Then, upon deriving the best chain-pair the stakeholder attempts to extend the PoS-chain of the best chain-pair through the PoS algorithm (see Algorithm 4). If this stakeholder is the lucky stakeholder, then he will produce and sign a new block and extend the PoS-chain, and finally broadcasting the newly extended chain-pair to the network.

Algorithm 2: Main Protocol: stakeholder. The stakeholder algorithm is parameterized by signing and verification keys $(\mathsf{sk}, \mathsf{vk})$.

```
$\mathbb{C} \leftarrow \epsilon$;
while True do
    $\mathbb{C} \leftarrow$ all chain-pairs received from the network.
    $X \leftarrow$ all payloads received from the network.
    $\langle \mathcal{C}, \widetilde{\mathcal{C}} \rangle \leftarrow \mathsf{BestValid}(\mathbb{C})$
    $\widetilde{\mathcal{C}}_{\text{new}} \leftarrow \mathsf{PoS}(\mathsf{sk}, \mathsf{vk}, X, \langle \mathcal{C}, \widetilde{\mathcal{C}} \rangle)$
    if $\widetilde{\mathcal{C}} \neq \widetilde{\mathcal{C}}_{\text{new}}$ then
        $\widetilde{\mathcal{C}} \leftarrow \widetilde{\mathcal{C}}_{\text{new}}$
        $\mathbb{C} \leftarrow \mathbb{C} \cup \langle \mathcal{C}, \widetilde{\mathcal{C}} \rangle$
        BROADCAST($\langle \mathcal{C}, \widetilde{\mathcal{C}} \rangle$);
```

B.1 Proof-of-Work Blockchain

B.1.1 Attempting and Successful Blocks

The players listen to and receive many chain-pairs (consisting of new PoW-blocks) from the network. For each player, if he receives a new PoW-block without its corresponding PoS-block, the new PoW-block becomes an *attempting block* with respect to the view of the player. On the other hand, if he receives a new PoW-block with the corresponding PoS-block, this PoW-block is now considered as a *successful block* in the player's local view.

B.1.2 Proof-of-Work Block Structure

Let $\mathsf{H}(\cdot), \mathsf{G}(\cdot)$ be cryptographic hash functions with output in $\{0,1\}^\kappa$ where κ is the security parameter. As introduced, we have two different types of PoW-blocks: attempting blocks and successful blocks. The structure of attempting and successful blocks are the same. However, they have different meaning as follows: (1) no PoS-block links to an attempting block, and (2) we only count the successful blocks when calculating the real length of a proof-of-work chain. A PoW-block B is a tuple of the form $\langle ctr, h_\text{w}, h_\text{s}, h_\text{a}, w \rangle$, where

- Notation ctr denotes a positive integer, $1 \leq ctr \leq \mathsf{q}$, and $\mathsf{q} \in \mathbb{N}$ is the maximum number of random oracle queries from each player per unit of time,
- Notation h_w denotes the digest of the previous proof-of-work block, where $h_\text{w} \in \{0,1\}^\kappa$,
- Notation h_s denotes the digest of the previous proof-of-stake block, where $h_\text{s} \in \{0,1\}^\kappa$,
- Notation h_a denotes the digest of the latest seen attempting proof-of-work block, where $h_\text{a} \in \{0,1\}^\kappa$
- Notation w is a random nonce, where $w \in \{0,1\}^\kappa$,

and block B satisfies the inequality[2]

$$\mathsf{H}(ctr, \mathsf{G}(h_{\mathrm{w}}, h_{\mathrm{s}}, h_{\mathrm{a}}, w)) < T$$

where the parameters $T \in \mathbb{N}$ denotes the *current proof-of-work target* of the block.

A PoW-chain $\mathcal{C}$ consists of a sequence of ℓ concatenated successful PoW-blocks. We denote $\mathsf{head}(\mathcal{C})$ as to the *topmost successful PoW-block B_ℓ* in blockchain $\mathcal{C}$. That is, we do not consider an attempting block as the head of the proof-of-work chain, and the head of a PoW-chain is not necessary the topmost PoW-block since the topmost PoW-block could be an attempting block. Moreover, we only link a new proof-of-work block to the head of the proof-of-work chain. This implies that there may exist multiple attempting blocks attaching to a successful block in our proof-of-work chain. If a proof-of-work blockchain $\mathcal{C}$ is a prefix of another blockchain $\mathcal{C}'$, we write $\mathcal{C} \preceq \mathcal{C}'$.

B.1.3 Extending a Proof-of-Work Blockchain

Miners maintain our system by extending the PoW-chain. Algorithm 3 formally describes how to extend a proof-of-work blockchain in our system. This algorithm is parameterized by hash functions $\mathsf{H}(\cdot), \mathsf{G}(\cdot)$ as well as two positive integers q, T where q is the maximum number of random oracle queries of each miner per unit of time, and T determines the "current block target" of the PoW. The algorithm works as follows: given a chain-pair $\langle \mathcal{C}, \widetilde{\mathcal{C}} \rangle$, the algorithm computes the hash h_{w} of the previous PoW-block (the head of the PoW-chain), the hash h_{s} of the head of the PoS-chain, and the hash h_{a} of the latest attempting block collected. We emphasize that, the head of the PoW-chain is the "most recent successful block" on the chain (not an attempting block), and without connecting to the head of the current PoS-chain or the latest attempting block, the adversary can rewrite any information stored on the chain-pair. The algorithm then samples a random initial string w of length κ, then it increments ctr and checks if $\mathsf{H}(ctr, \mathsf{G}(h_{\mathrm{w}}, h_{\mathrm{s}}, h_{\mathrm{a}}, w)) < T$; if a suitable $(ctr, h_{\mathrm{w}}, h_{\mathrm{s}}, h_{\mathrm{a}}, w)$ is found then the algorithm succeeds in solving the PoW and extends blockchain $\mathcal{C}$ by one block. We can this new PoW-block is now attached to the header of $\mathcal{C}$; If no suitable $(ctr, h_{\mathrm{w}}, h_{\mathrm{s}}, h_{\mathrm{a}}, w)$ is found, the algorithm simply returns the chain unaltered.

B.2 Proof-of-Stake Blockchain

We remark that this part is almost the same as that in the 2-hop blockchain [14].

[2]A more comprehensive version of the inequality, i.e., $(\mathsf{H}(ctr, \mathsf{G}(h_{\mathrm{w}}, h_{\mathrm{s}}, h_{\mathrm{a}}, w)) < T) \wedge (ctr \leq \mathsf{q})$ can be found in [18].

Algorithm 3: The *proof-of-work* function, parameterized by positive integers q, T and hash functions $\mathsf{H}(\cdot)$, $\mathsf{G}(\cdot)$. The input is $(\langle \mathcal{C}, \widetilde{\mathcal{C}} \rangle)$.

```
function PoW($(\langle \mathcal{C}, \widetilde{\mathcal{C}} \rangle)$)
    Let $\mathcal{B}$ be a set of attempting blocks that attach to $\mathsf{head}(\mathcal{C})$.
    Let $l$ be the number of attempting blocks in $\mathcal{B}$.
    Set $h_{\mathrm{a}} := \perp$ if $\mathcal{B}$ is empty.
    $h_{\mathrm{w}} \leftarrow \mathsf{H}(\mathsf{head}(\mathcal{C}))$; $h_{\mathrm{s}} \leftarrow \mathsf{H}(\mathsf{head}(\widetilde{\mathcal{C}}))$
    $ctr \leftarrow 1$; $w \leftarrow \{0,1\}^\kappa$
    $B \leftarrow \epsilon$
    if $l \neq 0$ then
        Let $B^l$ be the latest PoW-block found $\mathcal{B}$ that attaches to $\mathsf{head}(\mathcal{C})$.
        $h_{\mathrm{a}} \leftarrow \mathsf{H}(B^l)$;
    while $ctr \leq q$ do
        if $(\mathsf{H}(ctr, \mathsf{G}(h_{\mathrm{w}}, h_{\mathrm{s}}, h_{\mathrm{a}}, w)) < T) \wedge (ctr \leq q)$ then
            $B \leftarrow \langle ctr, h_{\mathrm{w}}, h_{\mathrm{s}}, h_{\mathrm{a}}, w \rangle$
            $\mathcal{C} \leftarrow \mathcal{C}B$
                /* Extend proof-of-work chain */
        $ctr \leftarrow ctr + 1$
    return $\mathcal{C}$ ;    /* Return the updated chain */
```

In our system, a digital signature scheme is used by stakeholders to create new valid PoS-blocks. Let $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ be the digital signature scheme. Let $\widetilde{\mathsf{H}}(\cdot)$ be a cryptographic hash function with output in $\{0,1\}^\kappa$ where κ is the security parameter. We now introduce the format of a PoS-block. In our system, each valid PoS-block is coupled with a valid PoW-block. Based on a given PoW-block B, a valid stakeholder with the verification key vk such that

$$\widetilde{\mathsf{H}}(B, \mathsf{vk}) < \widetilde{T}$$

is allowed to generate the corresponding PoS-block $\widetilde{B}$. Here, $\widetilde{T}$ is the *current proof-of-stake target.*

The PoS-block $\widetilde{B}$ is defined as a tuple of the form $\langle B, \mathsf{vk}, X, \sigma \rangle$. Here, $X \in \{0,1\}^*$ is the payload of the proof-of-stake block $\widetilde{B}$ (also denoted as $\mathsf{payload}(\widetilde{B})$); and σ is a signature for (B, X), i.e., $\sigma = \mathsf{Sign}_{\mathsf{sk}}(B, X)$ (sk is the corresponding signing key of vk.)

We define $\mathsf{head}(\widetilde{\mathcal{C}})$ as the topmost PoS-block of the proof-of-stake chain $\widetilde{\mathcal{C}}$. We note that, in PoS-chain, payload is stored, and we use $\mathsf{payload}(\widetilde{\mathcal{C}})$ to denote the information we store in $\widetilde{\mathcal{C}}$. If ℓ is the total number of PoS-blocks in the PoS-chain $\widetilde{\mathcal{C}}$, then we have $\mathsf{payload}(\widetilde{\mathcal{C}}) = \|_{i=1}^{\ell} \mathsf{payload}(\widetilde{B}_i)$, where $\|$ is the concatenation notation.

B.2.1 Extending a Proof-of-Stake Blockchain

In Algorithm 4, we describe how the stakeholders extend PoS-chains. Intuitively, if the stakeholder with verification key vk is the elected one, he can to sign and generate a new PoS-block.

In more details, Algorithm 4 processes as follows. Upon receiving input $(\mathsf{sk}, \mathsf{vk}, X, \langle \mathcal{C}, \widetilde{\mathcal{C}} \rangle)$, the algorithm attempts to extend the specified PoS-chain $\widetilde{\mathcal{C}}$ in the chain-pair $\langle \mathcal{C}, \widetilde{\mathcal{C}} \rangle$; The algorithm then executes the following two main steps:

Step 1—Leader Election. The algorithm collects all attempting blocks that link to the head of $\mathcal{C}$. We denote the set of these attempting blocks as $\mathcal{B}$, and denote l as the number of blocks in $\mathcal{B}$; here, $l = 0$ means this set is empty and there are no attempting blocks that follow $\mathsf{head}(\mathcal{C})$. Then, let B be the latest attempting block in $\mathcal{B}$. If $\mathcal{B}$ is not empty meaning that there exits a block B, the algorithm checks if the verification key vk is a valid key (owns the stake) and the inequality $\widetilde{\mathsf{H}}(B, \mathsf{vk}) < \widetilde{T}$ holds. If yes, the stakeholder with the key pair $(\mathsf{sk}, \mathsf{vk})$ is the winning stakeholder. That is, stakeholders are elected based on this inequality $\widetilde{\mathsf{H}}(B, \mathsf{vk}) < \widetilde{T}$. By this mechanism, the proof-of-work blockchain is treated as a biased random beacon for electing a stakeholder.

Step 2—Signature generation. After the first step, the stakeholder with signing and verification keys $(\mathsf{sk}, \mathsf{vk})$ is the winning stakeholder. The algorithm then generates a signature $\sigma \leftarrow \mathsf{Sign}_{\mathsf{sk}}(B, X)$ and forms a new PoS-block $\widetilde{B} = \langle B, \mathsf{vk}, X, \sigma \rangle$. We then say the stakeholder with the key pair $(\mathsf{sk}, \mathsf{vk})$ extends the specified PoS-chain $\widetilde{\mathcal{C}}$ with the new block $\widetilde{B}$.

The Figure 10 illustrate the structure of a chain-pair after executing Algorithm 4. As shown in the figure, we have B the most recently produced PoW-block, and the new PoS-block $\mathcal{B}$ links to B by storing B in the block.

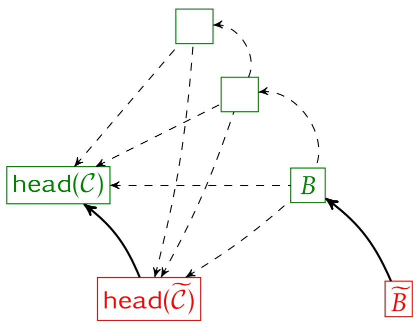

Figure 10: Generating new PoS-blocks

Algorithm 4: The *proof-of-stake* function, parameterized by a signature scheme $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$, a parameter $\widetilde{T}$, a hash function $\widetilde{\mathsf{H}}(\cdot)$. The input is $(\mathsf{sk}, \mathsf{vk}, X, \langle \mathcal{C}, \widetilde{\mathcal{C}} \rangle)$.

```
1  function PoS($\mathsf{sk}, \mathsf{vk}, X, \langle \mathcal{C}, \widetilde{\mathcal{C}} \rangle$)
2      Let $\mathcal{B}$ is the set of all attempting blocks that attach to head($\mathcal{C}$).
3      Let $l$ be the number of attempting blocks in $\mathcal{B}$.
4      Let $B$ be the latest PoW-block in $\mathcal{B}$ that attaches to head($\mathcal{C}$).
       /* If there is one new PoW-block that attaches to head($\mathcal{C}$).  */
5      if $l > 0$ then
6          if $(\widetilde{\mathsf{H}}(B, \mathsf{vk}) < \widetilde{T})$ then
7              $\sigma \leftarrow \mathsf{Sign}_{\mathsf{sk}}(B, X)$
8              $\widetilde{B} \leftarrow \langle B, \mathsf{vk}, X, \sigma \rangle$
9              $\widetilde{\mathcal{C}} \leftarrow \widetilde{\mathcal{C}}\widetilde{B}$
                   /* Extend proof-of-stake chain */
10     return $\widetilde{\mathcal{C}}$;     /* Return the updated chain */
```

B.3 Validating a Chain-pair

Chain-Pair validation is the most important process in our design. Before going to explain our validation algorithm. We formally describe the following predicates used in the $\mathsf{ValidateChainPair}$ algorithm.

Predicate $\mathsf{ValidPoW}^{\mathsf{q},T}_{\mathsf{H},\mathsf{G}}(B, B', \widetilde{B}, \mathcal{B}, \hat{\mathcal{B}})$. This predicate is parameterized by two integers q, T, and two hash functions $\mathsf{H}(\cdot), \mathsf{G}(\cdot)$. The goal of this predicate is to check the validity of a successful PoW-block B upon receiving inputs: the successful block B, another successful block B', a PoS-block $\widetilde{B}$, two sets of attempting PoW-blocks $\mathcal{B}$ and $\hat{\mathcal{B}}$. Here, the block B consists of $\langle ctr, h_{\mathrm{w}}, h_{\mathrm{s}}, h_{\mathrm{a}}, w \rangle$, block B' consists of $\langle ctr', h'_{\mathrm{w}}, h'_{\mathrm{s}}, h'_{\mathrm{a}}, w' \rangle$, the set $\mathcal{B}$ consists of l attempting blocks $\{B^1, \ldots, B^l\}$ where each block $B^i = \langle ctr^i, h^i_{\mathrm{w}}, h^i_{\mathrm{s}}, h^i_{\mathrm{a}}, w^i \rangle$ for $1 \le i \le l$, and the set $\hat{\mathcal{B}}$ consists of $\hat{l}$ attempting blocks $\{\hat{B}^1, \ldots, \hat{B}^{\hat{l}}\}$ where each block $\hat{B}^j = \langle \hat{ctr}^j, \hat{h}^j_{w}, \hat{h}^j_{s}, \hat{h}^j_{b}, \hat{w}^j \rangle$ for $1 \le j \le \hat{l}$. Note that, PoW-blocks in $\mathcal{B}$ and $\hat{\mathcal{B}}$ are in temporal-order. The predicate checks the following.

– B is properly solved if

$$\mathsf{H}(ctr, \mathsf{G}(h_{\mathrm{w}}, h_{\mathrm{s}}, h_{\mathrm{a}}, w)) < T$$

– B links to the previous PoW-block B' if $h_{\mathrm{w}} = \mathsf{H}(B')$.
– B links to the previous PoS-block $\widetilde{B}$ if $h_{\mathrm{s}} = \mathsf{H}(\widetilde{B})$.
– If $\hat{l} > 0$, check whether B links to the latest attempting block in the second set $\hat{\mathcal{B}}$ if $h_{\mathrm{a}} = \mathsf{H}(\hat{B}^{\hat{l}})$.

- If $l > 0$, check whether all attempting blocks in $\mathcal{B}$ are properly solved if for $1 \le i \le l$,

$$\mathsf{H}(ctr^i, \mathsf{G}(h_{\mathrm{w}}^i, h_{\mathrm{s}}^i, h_{\mathrm{a}}^i, w^i)) < T$$

- If $l > 0$, check whether all attempting blocks $B^i = \langle ctr^i, h_{\mathrm{w}}^i, h_{\mathrm{s}}^i, h_{\mathrm{a}}^i, w^i \rangle$ in $\hat{\mathcal{B}}$ are properly linked if
  - B^i links to the previous PoW-block B' if $h_{\mathrm{w}}^i = \mathsf{H}(B')$.
  - B^i links to the previous PoS-block $\widetilde{B}$ if $h_{\mathrm{s}}^i = \mathsf{H}(\widetilde{B})$.
  - B^i links to the previous attempting block in the set $\mathcal{B}$ if $h_{\mathrm{a}}^i = \mathsf{H}(B^{i-1})$ (Note that, we consider $B^0 = B'$.)

The predicate $\mathsf{ValidPoW}_{\mathsf{H,G}}^{\mathsf{q},T}$ outputs 1 if and only the examined block B passes all tests described above.

Predicate $\mathsf{ValidPoS}_{\widetilde{\mathsf{H}}}^{\Sigma,\widetilde{T}}(\widetilde{B}, B)$**.** This predicate is parameterized by a signature scheme Σ, an integer $\widetilde{T}$, and a hash function $\widetilde{\mathsf{H}}(\cdot)$. The goal of this predicate is to check the validity of a PoS-block $\widetilde{B} = \langle B', \mathsf{vk}, X, \sigma \rangle$ upon receiving inputs: a PoS-block $\widetilde{B}$, a PoW-block B.

The predicate checks the following

- $\widetilde{B}$ is generated by an elected stakeholder if $\widetilde{\mathsf{H}}(B, \mathsf{vk}) < \widetilde{T}$
- $\widetilde{B}$ links to the corresponding PoW-block B if $B = B'$
- The signature σ of $\widetilde{B}$ is properly generated if $\mathsf{Verify}_{\mathsf{vk}}(B', X, \sigma) = 1$

The predicate $\mathsf{ValidPoW}_{\widetilde{\mathsf{H}}}^{\Sigma,\widetilde{T}}$ outputs 1 if and only if the examined PoS-block $\widetilde{B}$ passes all tests described above.

Our chain-pair validation algorithm, denoted ValidateChainPair, is introduced to examine if a pair of chains (including a PoW-chain and a PoS-chain) is valid. Intuitively, a valid chain-pair means its members PoW-chain $\mathcal{C}$ and PoS-chain $\widetilde{\mathcal{C}}$ are both valid, respectively. Furthermore, each block of the PoS-chain must contain the valid supporting signature for the corresponding block of the PoW-chain. The algorithm is parameterized by the signature scheme Σ, parameters q, T, $\widetilde{T}$, the hash functions $\widetilde{\mathsf{H}}(\cdot)$, $\mathsf{H}(\cdot)$, $\mathsf{G}(\cdot)$, and the content validation predicate $V(\cdot)$. Note that, the predicate $V(\cdot)$, introduced in [18], determines the proper structure of the information (i.e., payload) that is stored into the PoS-chains.

The ValidateChainPair algorithm takes as input a chain-pair $(\mathcal{C}, \widetilde{\mathcal{C}})$. It first applies the validation predicate $V(\cdot)$ to check the payload in the PoS-chain $\widetilde{\mathcal{C}}$. If the payload is valid, then starting from the head of $\mathcal{C}$, for every successful PoW-block and its corresponding PoS-block in the PoW-chain $\mathcal{C}$, the algorithm applies the predicate $\mathsf{ValidPoW}_{\mathsf{H,G}}^{\mathsf{q},T}$ and $\mathsf{ValidPoS}_{\widetilde{\mathsf{H}}}^{\Sigma,\widetilde{T}}$, respectively, to validate the blocks. If both predicates output 1, the considered block-pair (including a PoW-block and its corresponding PoS-block) are valid. Then, the examining chain-pair is valid if all block-pairs are valid. Please refer to Algorithm 5 for more details.

Algorithm 5: The *chain-pair validation algorithm*, parameterized by a signature scheme $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$, the stake-identity set S, parameters q, T, $\widetilde{T}$, an initial chain $\mathcal{C}_{\mathsf{init}}$, the hash functions $\widetilde{\mathsf{H}}(\cdot)$, $\mathsf{H}(\cdot)$,$\mathsf{G}(\cdot)$ and the content validation predicate $V(\cdot)$. The input is $(\langle \mathcal{C}, \widetilde{\mathcal{C}} \rangle)$.

```
function ValidateChainPair($\langle \mathcal{C}, \widetilde{\mathcal{C}} \rangle$)
    $b \leftarrow V(\mathsf{payload}(\widetilde{\mathcal{C}}))$
    if $b$ = True then
        repeat
            $B \leftarrow \mathsf{H}(\mathsf{head}(\mathcal{C}))$.
            Let $\mathcal{B}$ is the set of all attempting PoW-blocks attaching to head($\mathcal{C}$).
            $\widetilde{B} \leftarrow \mathsf{H}(\mathsf{head}(\widetilde{\mathcal{C}}))$.
            Truncate all PoW-blocks from the head of $\mathcal{C}$ (including the head), and truncate the head of $\widetilde{\mathcal{C}}$.
            /* obtain new heads */
            Let $\hat{\mathcal{B}}$ is the set of all attempting PoW-blocks attaching to the newhead($\mathcal{C}$).
            $b_1 \leftarrow \mathsf{ValidPoW}_{\mathsf{H,G}}^{\mathsf{q},T}(B, \mathsf{head}(\mathcal{C}), \mathsf{head}(\widetilde{\mathcal{C}}), \mathcal{B}, \hat{\mathcal{B}})$
            $b_2 \leftarrow \mathsf{ValidPoS}_{\widetilde{\mathsf{H}}}^{\Sigma,\widetilde{T}}(\widetilde{B}, B)$
            $b = b_1 \wedge b_2$
        until $(\mathcal{C} = \mathcal{C}_{\mathsf{init}}) \vee (b = \mathtt{False})$;
    return $b$
```

B.4 Best Chain-pair Strategy

In this section, we describe the rules in which a single chain-pair is selected for consensus. We introduce an algorithm BestValid for selecting the best chain-pair on a set of chain-pair $\mathbb{C}$. Please refer to Algorithm 6 for more details.

Algorithm 6: The BestValid, parameterized by the $\mathsf{Max}(\cdot, \cdot)$ function. The input is $(\mathbb{C})$

```
function BestValid($\mathbb{C}$)
    $temp \leftarrow \epsilon$
    foreach $\langle \mathcal{C}, \widetilde{\mathcal{C}} \rangle \in \mathbb{C}$ do
        if ValidateChainPair($\mathcal{C}, \widetilde{\mathcal{C}}$) then
            $temp \leftarrow \mathsf{Max}(\langle \mathcal{C}, \widetilde{\mathcal{C}} \rangle, temp)$
    return ($temp$)
```

The BestValid algorithm is executed by miners and stakeholders. It is by the function Max$(\cdot,\cdot)$ (see Algorithm 7) which outputs the chain-pair having the most number of successful PoW-blocks. That is, function Max$(\cdot,\cdot)$ only counts the successful PoW-blocks on the proof-of-work chain.

Algorithm 7: The Max function. The input is $(\langle \mathcal{C}_1, \widetilde{\mathcal{C}}_1\rangle, \langle \mathcal{C}_2, \widetilde{\mathcal{C}}_2\rangle)$

```
function Max($(\langle \mathcal{C}_1, \widetilde{\mathcal{C}}_1\rangle, \langle \mathcal{C}_2, \widetilde{\mathcal{C}}_2\rangle)$)
    Let $\ell_i$ denote the number of successful PoW-blocks in $\mathcal{C}_i$, for $i \in \{1, 2\}$.
    Let $\mathcal{B}_i$ is the set of all attempting PoW-blocks attaching to head($\mathcal{C}_i$), for $i \in \{1, 2\}$.
    Let $l_i$ be the number of attempting blocks in $\mathcal{B}_i$.
    for $i \in \{1, 2\}$ do
        if $l_i \neq 0$ then
            $\ell_i \leftarrow \ell_i + 1$
    if $\ell_1 \geq \ell_2$ then
        return $\langle \mathcal{C}_1, \widetilde{\mathcal{C}}_1\rangle$
    else
        return $\langle \mathcal{C}_2, \widetilde{\mathcal{C}}_2\rangle$
```

Tie breaking. In our system, we break the tie deterministically. In more detail, if the two chain-pairs have the same number of successful PoW-blocks, then the one having the most number of attempting blocks would win. If they even have the same number of attempting blocks, we would hash the head of the proof-of-work chain in each chain-pair, and choose the one with the smaller hash value.

To increase the readability, we demonstrate a toy example to illustrate our typical best chain-pair policy in Figure 11. In the figure, we have two chain-pairs. We present a successful PoW-block as a filled green box, an attempting PoW-block as an unfilled green box, and a PoS-block as a filled red box. There, the first chain-pair only has 4 successful blocks where the second chain-pair has 5 successful blocks. By Algorithm 6, the second chain-pair is the best one.

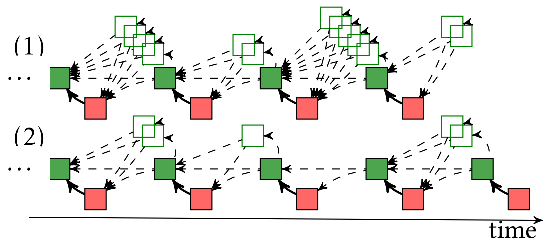

Figure 11: A toy example for illustrating the best chain-pair policy.