2-hop Blockchain: Combining Proof-of-Work and Proof-of-Stake Securely

Tuyet Duong[1⋆], Lei Fan[2], Jonathan Katz[3⋆⋆], Phuc Thai[1⋆⋆⋆], and Hong-Sheng Zhou[1⋆⋆⋆]

[1] Virginia Commonwealth University, {duongtt3,thaipd,hszhou}@vcu.edu
[2] Shanghai Jiao Tong University, fanlei@sjtu.edu.cn
[3] George Mason University, jkatz2@gmail.com

Abstract. Bitcoin-like blockchains use a proof-of-work (PoW) mechanism, where security holds if the majority of the computing power is under the control of honest players. However, this assumption has been seriously challenged recently, and Bitcoin-like systems fail if this assumption is violated. In this work we propose a novel *2-hop* blockchain protocol that combines PoW and proof-of-stake (PoS) mechanisms. Our analysis shows that the protocol is secure as long as the honest players control a majority of the *collective* resources (which consist of both computing power and stake). In particular, even if the adversary controls more than 50% of the computing power, security still holds if the honest parties hold sufficiently high stake in the system. As an added contribution, our protocol also remains secure against adaptive adversaries.

1 Introduction

Cryptocurrencies like Bitcoin [28] have been a phenomenal success. At the heart of Bitcoin is a global, distributed ledger, called a *blockchain*, that records transactions in successive time windows. The blockchain is maintained by a peer-to-peer network of *miners* via a so-called proof-of-work (PoW) mechanism: in each time window, cryptographic puzzles (also called proof-of-work puzzles [15, 1]) are generated, and all miners are incentivized to solve those puzzles; the first miner who finds a puzzle solution is allowed to extend the blockchain with a block of transactions. The more computing power a miner invests, the better its chances of solving a puzzle first.

⋆ Work supported in part by a research gift from IOHK.
⋆⋆ Portions of this work were done while at the University of Maryland, and were performed under financial assistance award 70NANB19H126 from U.S. Department of Commerce, National Institute of Standards and Technology.
⋆⋆⋆ Work supported in part by NSF award #1801470, and a research gift from Ergo Platform.

Bitcoin is an *open* system; any player who invests a certain amount of computing resources is allowed to join the effort of maintaining the blockchain. This feature, along with a smart incentive strategy, have helped the system attract a huge amount of computing resources over the past several years.

The *Nakamoto consensus protocol* underlying Bitcoin has recently been proven secure in various models. In particular, Garay et al. [19] and Pass et al. [30] showed that, assuming the majority of mining power is controlled by honest miners, Nakamoto consensus satisfies several important security properties. On the other hand, if an adversary controls a majority of the computational power in the network, security of Bitcoin cannot be guaranteed.

While it is appealing to assume that the majority of computing power in a blockchain network is honest, this assumption has been seriously challenged in recent years. For example, in 2014 the mining pool GHash.io exceeded 50% of the computational power in the Bitcoin network [21]. In 2017, one mining pool controlled 50% of the mining power in the Zcash system.[1] Currently, many of the top Bitcoin mining pools are in China; at times, they have collectively controlled more than 60% of the mining power in the Bitcoin ecosystem.[2] Efforts have been made to address this crisis, with some work [27] trying to discourage the formation of mining pools. However, these ideas have not seen much adoption in practice, and it is anyway unclear whether they would prevent certain types of attacks (e.g., nation states who wish to disrupt a cryptocurrency).

In part to address these issues, other design paradigms for blockchains have been considered. The most prominent such designs are based on *proof-of-stake* (PoS) mechanisms, which require miners to own a certain amount of coins ("stake") in order to extend the blockchain; the probability that a particular miner is allowed to extend the blockchain in any iteration is proportional to the amount of stake it owns.

1.1 Our Results

We propose a *hybrid* blockchain protocol that uses a combination of PoW and PoS mehanisms. We prove that security of the blockchain holds as long as the honest parties control a majority of the collective resources in the system, where these collective resources consist of both computing power and stake. As an additional contribution, and in contrast to several

[1] See `https://twitter.com/kyletorpey/status/910622595388715020`.
[2] See `https://www.buybitcoinworldwide.com/mining/pools`.

other PoS protocols that have been proposed, we show that our protocol tolerates an *adaptive* adversary who can decide which parties to corrupt during the course of the protocol execution. Source code for our protocol is publicly available (`https://bitbucket.org/twinscoinccs/twinscoin`), and an experimental evaluation of the protocol has been done [10]. Our focus here is on definitions and proofs of security. In what follows, we give an overview of the underlying ideas.

Our main idea is to have two coupled blockchains, one (denoted $\mathcal{C}$) based a PoW mechanism and another (denoted $\tilde{\mathcal{C}}$) using a PoS-based approach; we refer to *PoW-miners* who extend the former and *PoS-holders* (or *stakeholders*) who extend the latter, but of course one entity may play both roles. These two blockchains are extended alternately, so that their respective heights are always within one block of each other. Roughly, the overall (logical) blockchain is extended by first having a PoW-miner extend $\mathcal{C}$ and then having a PoS-holder extend $\tilde{\mathcal{C}}$.

A pictorial illustration of our blockchain structure is given in Figure 1. Rectangular blocks correspond to $\mathcal{C}$, while circular blocks correspond to $\tilde{\mathcal{C}}$. (Our 2-hop blockchain can be bootstrapped[3] from an already existing blockchain, denoted by grey blocks in the figure.) Intuitively, $\mathcal{C}$ serves as a (possibly biased) random beacon for choosing a stakeholder to extend $\tilde{\mathcal{C}}$. If Nakamoto consensus is a 1-hop protocol, then ours is a *2-hop protocol*.

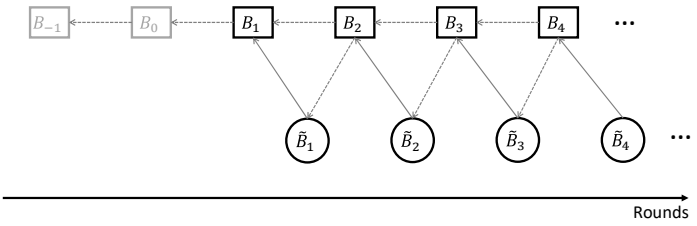

Fig. 1: 2-hop blockchain structure. Rectangular blocks denote the PoW chain, and circular blocks denote the PoS chain. Grey blocks (which need not be present) represent a pre-existing blockchain.

In more detail, say we have a PoW chain $\mathcal{C}$ consisting of blocks $B_1, \ldots, B_i$ and a PoS chain $\tilde{\mathcal{C}}$ consisting of blocks $\tilde{B}_1, \ldots, \tilde{B}_i$; when we refer to a *chain pair* we mean a pair of valid chains $\mathcal{C}, \tilde{\mathcal{C}}$ of the same height.

[3] This also implies that our design could be used as a strategy for converting a PoW-based blockchain into a pure PoS one, via a sequence of hard forks.

A new *block pair*—consisting of a new block B_{i+1} on $\mathcal{C}$ and a new block $\tilde{B}_{i+1}$ on $\tilde{\mathcal{C}}$—is generated as follows:

1. A PoW-miner extends $\mathcal{C}$ in the usual way, but building on both $\mathcal{C}$ and $\tilde{\mathcal{C}}$. That is, a miner first computes $h = \mathsf{hash}(B_i, \tilde{B}_i)$, and then attempts to find a suitable nonce ω such that $\mathsf{H}(h, \omega) < \mathtt{T}$ for some target $\mathtt{T}$. The new block on $\mathcal{C}$ takes the form $B_{i+1} = \langle h, \omega \rangle$.
2. Each PoS-holder holds two pairs of signing keys $(\mathsf{vk}, \mathsf{sk})$ and $(\mathsf{vk}', \mathsf{sk}')$, where the first is for a *unique* signature scheme and the second can be for an ordinary signature scheme. A PoS-holder can extend $\tilde{\mathcal{C}}$ if $\tilde{\mathsf{H}}(B_{i+1}, \tilde{\omega}, \mathsf{vk}) < \tilde{\mathtt{T}}$, where $\tilde{\omega} = \mathsf{Sign}_{\mathsf{sk}}(B_{i+1})$ and $\tilde{\mathtt{T}}$ is the current target. A new block takes the form $\tilde{B}_{i+1} = \langle (B_{i+1}, \tilde{\omega}, \mathsf{vk}), X, \sigma, \mathsf{vk}' \rangle$, where $X \in \{0,1\}^*$ is the payload of the block (also denoted as $\mathsf{payload}(\tilde{B}_{i+1})$); and $\sigma = \mathsf{Sign}'_{\mathsf{sk}'}((B_{i+1}, \tilde{\omega}, \mathsf{vk}), X)$.

While we defer a detailed discussion of the security of our protocol to Section 4, we observe some interesting properties of our protocol here:

- We can obtain adaptive security since the identity of the PoS-holder who can extend $\tilde{\mathcal{C}}$ is hidden before it publishes the new block; once the new PoS block is published and incorporated in the chain, it is too late for adversarial corruption to have any effect.
- Our protocol can resist a *nothing-at-stake* attack in which a malicious PoS-holder attempts to generate new blocks on multiple forks simultaneously. The reason is that in our 2-hop protocol the PoS chain builds on a PoW chain that, in general, will have fewer forks.
- Our protocol also resists *long-range attacks* in which an adversary tries to create a long chain, starting from the genesis block, that overtakes the main chain. This is because although creating a long PoS chain may be feasible, creating a long PoW chain is computationally infeasible.

1.2 Related Work

Bitcoin and the underlying PoW-based Nakamoto consensus protocol have been analyzed in both rational [17, 16, 32, 33] and adversarial [19, 30, 34, 22, 23] settings. The idea of using proofs of *stake* in place of proofs of work was first introduced in an online forum [5], and several PoS-based protocols have been proposed and implemented in deployed cryptocurrencies [29, 35, 3, 25, 11]. These early proposals lack rigorous security analysis. Subsequent work [9, 24, 31, 12, 13, 18, 20, 8, 2], done concurrently with our own (an early version of this work [14] was posted online in 2016), has

given formal proofs of security for PoS-based blockchain protocols. However, many proof-of-stake solutions [24, 31, 12] are not adaptively secure.

Hybrid PoW/PoS blockchains have been suggested by some previous work [25, 11, 4] and cryptocurrencies (e.g., `https://decred.org`). However, none of these systems has a formal proof of security. In addition, they are easily seen to be insecure against an adaptive adversary.

1.3 Paper Organization

We present our model and definitions in Section 2, and the details of our protocol in Section 3. In Section 4, we provide the high-level idea of our security analysis; full proofs are deferred to the full version of our paper.

2 Preliminaries

2.1 System Model

In order to study the security of Bitcoin-like protocols, Garay et al. [19] and then Pass et al. [30] set up the first cryptographic models by following Canetti's formulation of the "real world" executions [6, 7]. We further extend their models so that more blockchain protocols, e.g., 2-hop blockchains, are allowed. The underlying communication for blockchain protocols is the atomic unauthenticated broadcast in a semi-synchronous setting with an upper bound Δ network delay.

Protocol players. We consider two types of players, PoW-miners and PoS-holders, who may generate two types of blocks, PoW blocks and PoS blocks; We can then define PoW-chain and PoS-chain, for PoW blocks and PoS blocks respectively. These two types of chains could be tied and grow together. In our model, without loss of generality, we assume all PoW-miners have the same amount of computing power and all PoS-holders have the same amount of stake. Note that this is an "idealized model". In the reality, each different honest PoW-miner or PoS-holder may have a different amount of computing power/stake; nevertheless, this idealized model does not sacrifice generality since one can imagine that real-world honest PoW-miners/PoS-holders are simply clusters of some arbitrary number of honest idealized-model PoW-miners/PoS-holders.

Protocol execution. We consider the execution of the blockchain protocol $\Pi = (\Pi^{\mathsf{w}}, \Pi^{\mathsf{s}})$ that is directed by an environment $\mathcal{Z}(1^{\kappa})$ (where Π^{w} and Π^{s} denote the code run by PoW-miners and PoS-holders, and κ

is a security parameter), which activates up to n PoW-miners and up to $\tilde{n}$ PoS-holders. For simplicity, in this paper, we consider the static computing power and stake setting (where the total amount of computing power and stakes invested to the protocol will not change over time). We also consider that, all players, i.e., n PoW-miners and $\tilde{n}$ PoS-holders, are activated at the beginning of the protocol execution by the environment. Note that the environment $\mathcal{Z}$ can "manage" protocol players through an adversary $\mathcal{A}$ who may adaptively corrupt honest parties.

Protocol execution typically consists of two phases, *initialization* phase and *blockchain-extension* phase, and the execution proceeds in *rounds*. In the initialization phase, each PoW-miner can join the protocol execution by investing certain amount of computing power. Similarly, each PoS-holder can join the execution by investing certain amount of stake. Note that, the state of the initialization phase, if needed, can be recorded in the genesis block of blockchain system.

The blockchain-extension phase consists of multiple rounds. In each round, $\mathcal{Z}$ provides inputs for all players and receives outputs from them; the players communicate with each other via the network. More concretely, in each round, each honest player receives an input from the environment $\mathcal{Z}$, and potentially receives incoming network messages (delivered by the adversary $\mathcal{A}$), and then updates its local state; then based on the stored information, the player carries out some (mining) operations; in the case that a new block is generated, the player sends out the new block which will be guaranteed to be delivered to all players in the beginning of the next round.

Let $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$ be a random variable denoting the joint view of all parties in the above execution; note that this joint view fully determines the execution.

2.2 Security Properties

As in the original Bitcoin white paper [28], a proof-of-work blockchain is created and maintained by a set of players called PoW-miners. A PoW blockchain $\mathcal{C}$ consists of a sequence of concatenated PoW-blocks $B_\emptyset \| B_1 \| B_2 \| \cdots \| B_\ell$, where $\ell \geq 0$, and $B_\emptyset$ denotes the genesis block. For each blockchain, we specify several notations such as head, length, and subchain: (I) BLOCKCHAIN HEAD, denoted $\mathsf{head}(\mathcal{C})$, refers to the topmost block B_ℓ in chain $\mathcal{C}$; (II) BLOCKCHAIN LENGTH, denoted $\mathsf{len}(\mathcal{C})$, is the number of blocks in blockchain $\mathcal{C}$ *after the genesis block*, and here $\mathsf{len}(\mathcal{C}) = \ell$; (III) SUBCHAIN, refers to a segment of a blockchain; we use $\mathcal{C}[1, \ell]$ to denote an entire blockchain, and use $\mathcal{C}[j, m]$, with $j \geq 1$ and

$m \leq \ell$, to denote a subchain $B_j \| \cdots \| B_m$; in addition, we use $\mathcal{C}[i]$ to denote the i-th block B_i in blockchain $\mathcal{C}$; finally, if blockchain $\mathcal{C}$ is a prefix of another blockchain $\mathcal{C}'$, we write $\mathcal{C} \preceq \mathcal{C}'$. Similarly, we define a proof-of-stake (PoS) blockchain $\tilde{\mathcal{C}}$ by a sequence of concatenated PoS-blocks $\tilde{B}_\emptyset \| \tilde{B}_1 \| \tilde{B}_2 \| \cdots \| \tilde{B}_{\tilde{\ell}}$ for $\tilde{\ell} > 0$ that is maintained by a set of PoS-holders; here $\tilde{B}_\emptyset$ denotes the genesis block.

Chain growth, common prefix, and chain quality. Several important security properties have been considered for blockchain protocols. The *common prefix* and *chain quality* properties were originally formalized by Garay et al. [19], with the common prefix property later strengthened by Pass et al. [30]. The *chain growth* property was first formally defined by Kiayias et al. [22]. We provide corresponding definitions here, specialized to the case of a 2-hop blockchain protocol.

Definition 1 (Chain growth). *Consider 2-hop blockchain protocol Π. The chain growth property $\mathcal{Q}_{\text{cg}}$ with parameter $g \in \mathbb{R}$, states that for any honest player with the local chain-pair $\langle \mathcal{C}, \tilde{\mathcal{C}} \rangle$ in round r and $\langle \mathcal{C}', \tilde{\mathcal{C}}' \rangle$ in round r' where $r' - r > 0$, in $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$. It holds that $\mathsf{len}(\mathcal{C}') - \mathsf{len}(\mathcal{C}) \geq g \cdot (r' - r)$ and $\mathsf{len}(\tilde{\mathcal{C}}') - \mathsf{len}(\tilde{\mathcal{C}}) \geq g \cdot (r' - r)$.*

Definition 2 (Common prefix). *Consider 2-hop blockchain protocol Π. The common prefix property $\mathcal{Q}_{\text{cp}}$ with parameter $\kappa \in \mathbb{N}$ states that for any two honest players i in round r and j in round r' with the local chain-pairs $\langle \mathcal{C}_i, \tilde{\mathcal{C}}_i \rangle$ and $\langle \mathcal{C}_j, \tilde{\mathcal{C}}_j \rangle$, respectively, in $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$ where $r \leq r'$, it holds that $\mathcal{C}_i[\neg\kappa] \preceq \mathcal{C}_j$, and $\tilde{\mathcal{C}}_i[\neg\kappa] \preceq \tilde{\mathcal{C}}_j$.*

Definition 3 (Chain quality). *Consider 2-hop blockchain protocol Π. The chain quality property $\mathcal{Q}_{\text{cq}}$ with parameters $\mu \in \mathbb{R}$ and $\ell \in \mathbb{N}$ states that for any honest player with the local chain-pair $\langle \mathcal{C}, \tilde{\mathcal{C}} \rangle$ in $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$, it holds that for large enough ℓ consecutive block-pairs of chain-pair the ratio of honest block-pairs (an honest block-pair is a pair of an honest PoW-block and an honest PoS-block) is at least μ.*

Note that each block-pair consists of a PoW block and a PoS block. When the payload is attached only to PoS blocks (resp., PoW blocks), we can define the property of chain quality based on the ratio of honest PoS blocks (resp., PoW blocks).

3 Construction

3.1 The Main Protocol

Our protocol uses standard cryptographic building blocks: As in the original Bitcoin design, we use hash functions H and hash, and a regular digital signature scheme $\Sigma' = (\mathsf{Gen}', \mathsf{Sign}', \mathsf{Verify}')$; In addition, we need a *unique* signature scheme $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ and a third hash function $\tilde{\mathsf{H}}$.

Initialization. In the initialization phase, PoW players join the system by investing certain amount of computing power (as in the original Bitcoin design). To enable PoS players to register to the system, each PoS player first generates two pairs of signing-verification keys based on the signature schemes Σ and Σ'. More concretely, for all $i \in [\tilde{n}]$, PoS player S_i computes $(\mathsf{sk}_i, \mathsf{vk}_i) \leftarrow \mathsf{Gen}(1^\kappa)$ and $(\mathsf{sk}'_i, \mathsf{vk}'_i) \leftarrow \mathsf{Gen}'(1^\kappa)$. Then based on the identity $\mathsf{vk}_i, \mathsf{vk}'_i$, the PoS player S_i can join the system via investing certain amount of stake. We note that, the registration information including all PoS players' identities $\{\mathsf{vk}_i, \mathsf{vk}'_i\}_{i\in[\tilde{n}]}$ will be recorded in the genesis block $B_\emptyset$.

Blockchain extension. Our (logical) blockchain consists of a PoW-chain $\mathcal{C}$ and a PoS-chain $\tilde{\mathcal{C}}$. In order to extend the blockchain, for each block height, we will first extend PoW chain with one PoW-block, and then extend the PoS-chain with one PoS-block. Concretely, consider that we have a blockchain with height i; that is, PoW chain $\mathcal{C} = B_\emptyset \| B_1 \| \ldots \| B_i$ and PoS chain $\tilde{\mathcal{C}} = B_\emptyset \| \tilde{B}_1 \| \ldots \| \tilde{B}_i$. Next PoW block B_{i+1} and then PoS block $\tilde{B}_{i+1}$, will be generated, as follows.

In the first hop, PoW-miners attempt to extend $\mathcal{C}$ as usual, but building on both $\mathcal{C}$ and $\tilde{\mathcal{C}}$. That is, a miner first computes $h = \mathsf{hash}(B_i, \tilde{B}_i)$, and then attempts to find a suitable nonce ω such that $\mathsf{H}(h, \omega) < \mathtt{T}$ for some target $\mathtt{T}$. The new block on $\mathcal{C}$ takes the form $B_{i+1} = \langle h, \omega \rangle$.

The protocol now moves to the second hop. Intuitively, $\mathcal{C}$ serves as a (possibly biased) random beacon for choosing a stakeholder to extend $\tilde{\mathcal{C}}$. Once the new PoW-block B_{i+1} is published in the system, a PoS-holder is allowed to extend $\tilde{\mathcal{C}}$ if $\tilde{\mathsf{H}}(B_{i+1}, \tilde{\omega}, \mathsf{vk}) < \tilde{\mathtt{T}}$, where $\tilde{\omega} = \mathsf{Sign}_{\mathsf{sk}}(B_{i+1})$ and $\tilde{\mathtt{T}}$ is the current target. A new block takes the form $\tilde{B}_{i+1} = \langle (B_{i+1}, \tilde{\omega}, \mathsf{vk}), X, \sigma, \mathsf{vk}' \rangle$, where $X \in \{0,1\}^*$ is the payload of the block (also denoted as $\mathsf{payload}(\tilde{B}_{i+1})$); and $\sigma = \mathsf{Sign}'_{\mathsf{sk}'}((B_{i+1}, \tilde{\omega}, \mathsf{vk}), X)$.

At this moment, both PoW-chain $\mathcal{C}$ and PoS-chain $\tilde{\mathcal{C}}$ have been extended with one new block B_{i+1} and $\tilde{B}_{i+1}$, respectively, and the two-hop iterations continue. Please refer to Fig. 2 for the details of our main protocol. We note that all players collect blockchain information from the network, and wining players publish their generated blocks through the

PROTOCOL $\Pi = (\Pi^{\mathsf{w}}, \Pi^{\mathsf{s}})$

The protocol $\Pi = (\Pi^{\mathsf{w}}, \Pi^{\mathsf{s}})$ is executed by $(n + \tilde{n})$ players, including n PoW-miners and $\tilde{n}$ PoS-holders. Initially, each player holds local state $state_i := \{\langle \mathcal{C}_i, \tilde{\mathcal{C}}_i \rangle\}$ for $1 \leq i \leq n + \tilde{n}$, where $\mathcal{C}_i = \tilde{\mathcal{C}}_i = B_\emptyset$.

PoW-Miner Π^{w} by PoW-miner W_i, for $1 \leq i \leq n$, with local state $state_i$. Upon receiving an input message from $\mathcal{Z}$, and/or receiving messages of the form $(\text{BROADCAST}, \langle \mathcal{C}, \tilde{\mathcal{C}} \rangle)$ from the network, player W_i proceeds as follows:

1. *Select the best local chain-pair:*
   Set $\mathbb{C}$ to be the set of all chain-pairs collected from the network;
   Compute $\langle \mathcal{C}_{\text{best}}, \tilde{\mathcal{C}}_{\text{best}} \rangle := \mathsf{BestValid}(\mathbb{C} \cup \{\langle \mathcal{C}_i, \tilde{\mathcal{C}}_i \rangle\}, \textit{PoW-miner})$; // process $\mathsf{BestValid}()$ can be found in Fig. 3 below.
   Set $\mathcal{C}_i := \mathcal{C}_{\text{best}}$ and $\tilde{\mathcal{C}}_i := \tilde{\mathcal{C}}_{\text{best}}$.
2. *Attempt to extend PoW-chain:*
   - Compute $h := \mathsf{hash}(\mathsf{head}(\mathcal{C}_i), \mathsf{head}(\tilde{\mathcal{C}}_i))$;
   - Identify ω so that $\mathsf{H}(h, \omega) < \mathtt{T}$;
     If $\omega \neq \perp$, then set $B := \langle h, \omega \rangle$, $\mathcal{C}_i := \mathcal{C}_i \| B$, $state_i := state_i \cup \{\langle \mathcal{C}_i, \tilde{\mathcal{C}}_i \rangle\}$, and send $(\text{BROADCAST}, \langle \mathcal{C}_i, \tilde{\mathcal{C}}_i \rangle)$ to the network;

   Return an output message to the environment $\mathcal{Z}$.

PoS-Holder Π^{s} by PoS-holder S_j, for $n + 1 \leq j \leq n + \tilde{n}$, with $state_j$.

Upon receiving an input message from $\mathcal{Z}$, and/or receiving messages of the form $(\text{MESSAGE}, \langle \mathcal{C}, \tilde{\mathcal{C}} \rangle)$ from the network, player S_j proceeds as follows:

1. *Select the best local chain-pair:*
   Set $\mathbb{C}$ as the set of all chain-pairs collected from the network;
   Compute $\langle \mathcal{C}_{\text{best}}, \tilde{\mathcal{C}}_{\text{best}} \rangle := \mathsf{BestValid}(\mathbb{C} \cup \{\langle \mathcal{C}_j, \tilde{\mathcal{C}}_j \rangle\}, \textit{PoS-holder})$; // process $\mathsf{BestValid}()$ can be found in Fig. 3 below.
   Set $\mathcal{C}_j := \mathcal{C}_{\text{best}}$ and $\tilde{\mathcal{C}}_j := \tilde{\mathcal{C}}_{\text{best}}$.
2. *Attempt to extend PoS-chain:*
   - Set B as the new PoW-block in $\mathcal{C}_{\text{best}}$; Compute $\tilde{\omega} := \mathsf{Sign}_{\mathsf{sk}}(B)$;
   - If $\tilde{\mathsf{H}}(B, \tilde{\omega}, \mathsf{vk}) < \tilde{\mathtt{T}}$, then compute $\sigma \leftarrow \mathsf{Sign}'_{\tilde{\mathsf{sk}}'}((B, \tilde{\omega}, \mathsf{vk}), X)$, and set $\tilde{B} := \langle (B, \tilde{\omega}, \mathsf{vk}), X, \sigma, \mathsf{vk}' \rangle$; Set $\tilde{\mathcal{C}}_j := \tilde{\mathcal{C}}_j \| \tilde{B}$, and $state_j := state_j \cup \{\langle \mathcal{C}_j, \tilde{\mathcal{C}}_j \rangle\}$. and send $(\text{BROADCAST}, \langle \mathcal{C}_j, \tilde{\mathcal{C}}_j \rangle)$ to the network.

   Return an output message to the environment $\mathcal{Z}$.

Fig. 2: Our main protocol $\Pi = (\Pi^{\mathsf{w}}, \Pi^{\mathsf{s}})$.

network; The protocol Π is parameterized by a content validation predicate $V(\cdot)$, which determines the proper structure of the information that is stored into the blockchain (as in [19, 30]).

Best chain-pair strategy. In the above protocol execution, players including Protocol players, PoW-miners and PoS-holders, need to be aware, which chain-pair is the best one during the protocol execution. We de-

scribe a strategy to decide the best chain-pair via $\mathsf{BestValid}$ process; please see Fig. 3 for details. The $\mathsf{BestValid}$ process is parameterized by a content validation predicate $V(\cdot)$ which determines the proper structure of the information that is stored into the blockchain as in [19], and takes as input a chain-pair set $\mathbb{C}'$. Intuitively, the process validates all chain-pair $\langle \mathcal{C}, \tilde{\mathcal{C}} \rangle$ in $\mathbb{C}'$, then finds the valid chain-pairs with the longest PoW-chain. It also ensures that, if $\mathit{Type} = \mathit{PoW\text{-}miner}$, every valid chain-pair should have its member chains $\mathcal{C}$ and $\tilde{\mathcal{C}}$ of the same length. On the other hand, if $\mathit{Type} = \mathit{PoS\text{-}holder}$, we allow the PoW-chain to be longer than the PoS-chain by one block since there may be a new PoW-block produced in the previous rounds. We emphasize that since each valid PoS-block is tied to a PoW-block, and each PoW-block or PoS-block is valid if their peers are valid. The strategy to deal with multiple chains with the same length is discussed in Remark 1 at the end of this section.

PROCESS $\mathsf{BestValid}$

Upon receiving the input $(\mathbb{C}', \mathit{Type})$, the process $\mathsf{BestValid}$ proceeds as follows:

For every chain-pair $\langle \mathcal{C}, \tilde{\mathcal{C}} \rangle \in \mathbb{C}'$,

- If $\mathsf{len}(\mathcal{C}) - 1 = \mathsf{len}(\tilde{\mathcal{C}})$, then
  - Set $\ell := \mathsf{len}(\mathcal{C})$; Parse $\mathcal{C}[\ell]$ into $\langle h_\ell, \omega_\ell \rangle$.
  - If $\mathsf{hash}(\mathcal{C}[\ell-1], \tilde{\mathcal{C}}[\ell-1]) \neq h_\ell$, remove this chain-pair from $\mathbb{C}'$;
  - If $\mathsf{H}(\mathcal{C}[\ell]) \geq \mathtt{T}$, remove this chain-pair from $\mathbb{C}'$.
- Else if $\big(\mathsf{len}(\mathcal{C}) = \mathsf{len}(\tilde{\mathcal{C}})$ and $V(\mathsf{payload}(\tilde{\mathcal{C}})) = 1\big)$, or $\big(\mathsf{len}(\mathcal{C}) - 1 = \mathsf{len}(\tilde{\mathcal{C}})$ and $\mathit{Type} = \mathit{PoS\text{-}holder}\big)$, then for i from $\mathsf{len}(\tilde{\mathcal{C}})$ to 1, proceed as follows:
  - *Verify PoW-block* $\mathcal{C}[i]$*:* Parse $\mathcal{C}[i]$ into $\langle h_i, \omega_i \rangle$;
    If $\mathsf{hash}(\mathcal{C}[i-1], \tilde{\mathcal{C}}[i-1]) \neq h_i$, or $\mathsf{H}(\mathcal{C}[i]) \geq \mathtt{T}$, set $\mathrm{b}_1 := 0$; Else, set $\mathrm{b}_1 := 1$;
  - *Verify PoS-block* $\tilde{\mathcal{C}}[i]$*:* Parse $\tilde{\mathcal{C}}[i]$ into $\langle B, \tilde{\omega}, \mathsf{vk}, X, \sigma, \mathsf{vk}' \rangle$;
    If $\mathsf{Verify}'_{\mathsf{vk}'}((B, \tilde{\omega}, \mathsf{vk}, X), \sigma) = 1$, set $\mathrm{b}_2 := 1$; Else set $\mathrm{b}_2 := 0$;
    If $\mathsf{Verify}_{\mathsf{vk}}(B, \tilde{\omega}) = 0$, or $\tilde{\mathsf{H}}(B, \tilde{\omega}, \mathsf{vk}) \geq \tilde{\mathtt{T}}$, or $B \neq \mathcal{C}[i]$, set $\mathrm{b}_3 := 0$; Else, set $\mathrm{b}_3 := 1$;
  - If $\mathrm{b}_1 = 0$ or $\mathrm{b}_2 = 0$, or $\mathrm{b}_3 = 0$, remove this chain-pair from $\mathbb{C}'$.

Find the valid chain-pair $\langle \mathcal{C}_{\mathrm{best}}, \tilde{\mathcal{C}}_{\mathrm{best}} \rangle \in \mathbb{C}'$ with the longest PoW-chain. Then set $\langle \mathcal{C}_{\mathrm{best}}, \tilde{\mathcal{C}}_{\mathrm{best}} \rangle$ as the output.

Fig. 3: The chain set validation process $\mathsf{BestValid}$. The process is parameterized by a content validation predicate $V(\cdot)$.

Remark 1 (Tie breaking). Our protocol primarily deals with length so it makes sense to adopt a simple tie-breaking strategy to choose the best chain-pair from two chain-pairs of equal length. While there is work that show the advantages of choosing a chain randomly (viz. [17]), we follow the simple strategy considered in [19]; in which the best chain-pair is the one with the PoW-chain that is lexicographically the smallest. If two chain-pairs have same length, and the PoW-chains are same, we compare the PoS-chains with the same tie breaking mechanism for PoW-chains.

Remark 2 (Attaching transaction payloads to the PoW-chain). In the 2-hop protocol description above, the transaction payloads are attached to the PoS-chain. We note that, this is just a design choice; alternatively, the payloads can be attached to the PoW-chain. In the full version, we will provide the details.

4 Security Analysis

For simplicity, we analyze security assuming a fixed set of participants. Denote the total number of PoW-miners by n, and the portion of malicious computing power by ρ. Let p be the probability that a player can generate a PoW-block in a round. Let $\alpha = (1-\rho)np$ be the expected number of PoW-blocks that honest PoW-miners can find in a round. Let $\beta = \rho np$ be the expected number of PoW-blocks that malicious PoW-miners can find in a round. Thus $\frac{\alpha}{\beta} = \frac{1-\rho}{\rho}$. We assume $0 < \alpha \ll 1$, $0 < \beta \ll 1$ and $\alpha = \lambda\beta$ where $\lambda \in (0, \infty)$.

We then describe the important parameters in the second hop (i.e., proof-of-stake blockchain). Similarly, denote the total number of PoS-holders by $\tilde{n}$, and the portion of malicious stakes by $\tilde{\rho}$. Let $\tilde{p}$ be the probability that a PoW-block is mapped to a PoS-holder. We assume $\tilde{p}\tilde{n} \ll 1$. Let $\tilde{\alpha} = 1-(1-\tilde{p})^{(1-\tilde{\rho})\tilde{n}} \approx (1-\tilde{\rho})\tilde{n}\tilde{p}$ be the probability that a PoW-block is mapped to at least one honest PoS-holder. Let $\tilde{\beta} = 1-(1-\tilde{p})^{\tilde{\rho}\tilde{n}} \approx \tilde{\rho}\tilde{n}\tilde{p}$ be the probability that a PoW-block is mapped to at least one malicious PoS-holder.

Now, we have a parameter $\hat{\alpha} = \alpha\tilde{\alpha}$ which is the probability that honest parties find a new PoW-block and is mapped to an honest PoS-holder in a round. We also have $\hat{\beta} = \beta\tilde{\beta}$, the expected number that malicious parties find new PoW-blocks and are mapped to malicious PoS-holders in a round. We say $\hat{\alpha}$ and $\hat{\beta}$ are **collective** resources for honest parties and malicious parties respectively. Note that $\hat{\gamma} = \frac{\hat{\alpha}}{1+2\Delta\hat{\alpha}}$ can be viewed as a "discounted" version of $\hat{\alpha}$ due to the fact that the messages sent

by honest parties can be delayed by Δ rounds; $\hat{\gamma}$ corresponds to the "effective" honest collective resource.

As shown in the analysis of PoW protocol [19, 30], due to the network delay, the block time (i.e., the time period between two consecutive blocks) has to be set very long; in other words, the probability to generate new blocks is very small. We note however in our 2-hop protocol, the block time of generating PoW-blocks can be much shorter. As long as the block time of generating new *block-pairs* is long, the security properties of our 2-hop protocol can be achieved.

Note that, the expected number of PoW-blocks that are generated in a round is $\alpha + \beta = pn$; the expected number of PoS-blocks that map to a PoW-block is $\tilde{\alpha}+\tilde{\beta} \approx \tilde{p}\tilde{n}$. In our analysis, we assume $(\alpha+\beta)(\tilde{\alpha}+\tilde{\beta})\Delta \ll 1$; that is, most of the time, no block-pair is generated. We are now ready to state our main theorems.

Theorem 1 (Chain growth). *Consider 2-hop blockchain protocol $\Pi = (\Pi^{\mathsf{w}}, \Pi^{\mathsf{s}})$ in Section 3.1. For any honest player with the local chain-pair $\langle \mathcal{C}, \tilde{\mathcal{C}} \rangle$ in round r and $\langle \mathcal{C}', \tilde{\mathcal{C}}' \rangle$ in round $r' = r + t$ where $t > 0$. In $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$, the probability that*

$$\mathsf{len}(\mathcal{C}') - \mathsf{len}(\mathcal{C}) \geq g \cdot t, \qquad \mathsf{len}(\tilde{\mathcal{C}}') - \mathsf{len}(\tilde{\mathcal{C}}) \geq g \cdot t,$$

is at least $1 - e^{-\Omega(t)}$ where $g = (1-\delta)\hat{\gamma}$, for any $\delta > 0$.

Theorem 2 (Chain quality). *We assume $\hat{\gamma} = \hat{\lambda}(\alpha+\beta)\tilde{\beta}$ and $\hat{\lambda} > 1$. Consider protocol $\Pi = (\Pi^{\mathsf{w}}, \Pi^{\mathsf{s}})$ in Section 3.1. For any honest player with the local chain-pair $\langle \mathcal{C}, \tilde{\mathcal{C}} \rangle$. In $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$, the probability that for large enough ℓ consecutive block-pairs of chain-pair the ratio of honest block-pairs is no less than*

$$\mu = 1 - (1+\delta)\frac{(\alpha+\beta)\tilde{\beta} + \beta\tilde{\alpha}}{\hat{\gamma} + \beta\tilde{\alpha}}$$

is at least $1 - e^{-\Omega(\ell)}$, for any $\delta > 0$.

Theorem 3 (Common prefix). *We assume $\hat{\alpha} = \hat{\lambda}(\alpha+\beta)\tilde{\beta}$ and $\hat{\lambda} > 1$. Consider protocol $\Pi = (\Pi^{\mathsf{w}}, \Pi^{\mathsf{s}})$ in Section 3.1. Let κ be the security parameter. For any two honest players P_i in round r, and P_j in round r', with the local best chain-pairs $\langle \mathcal{C}_i, \tilde{\mathcal{C}}_i \rangle$, $\langle \mathcal{C}_j, \tilde{\mathcal{C}}_j \rangle$, respectively, in $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$ where $r \leq r'$, the probability that,*

$$\mathcal{C}_i[\neg\kappa] \preceq \mathcal{C}_j, \quad \mathcal{C}_j[\neg\kappa] \preceq \mathcal{C}_i, \quad \tilde{\mathcal{C}}_i[\neg\kappa] \preceq \tilde{\mathcal{C}}_j, \quad \tilde{\mathcal{C}}_j[\neg\kappa] \preceq \tilde{\mathcal{C}}_i,$$

is at least $1 - e^{-\Omega(\kappa)}$.

4.1 Proof Intuition

Due to space limitations, we defer the full security analysis to the full version of our paper. Here, we present the main ideas underlying the security analysis.

In our protocol, there are two types of players, PoW-miners and PoS-holders. Both PoW-miners and PoS-holders can be honest or malicious. In order to extend the pair of blockchains, a PoW-miner needs to generate a PoW-block first, and then the corresponding stakeholder will sign this block. We note that, our security analysis mainly focuses on PoS-chain, and the analysis for PoW-chain is followed from PoS-chain's. Consider that players may be honest or malicious, we have

- **Case 1:** An honest PoW-miner finds a new PoW-block which is mapped to an honest PoS-holder. The honest PoS-holder will generate the corresponding PoS-block faithfully.
- **Case 2:** A malicious PoW-miner finds a new PoW-block which is mapped to a malicious PoS-holder. The malicious PoS-holder may generate the corresponding PoS-block faithfully, or just discard it.
- **Case 3:** An honest PoW-miner finds a new PoW-block which is mapped to a malicious PoS-holder. Again, as in Case 2, the malicious PoS-holder may generate the corresponding PoS-block faithfully, or just discard it.
- **Case 4:** When a malicious PoW-miner finds a new PoW-block which is mapped to an honest PoS-holder. The malicious PoW-miner may publish the new PoW-block (so that the corresponding honest PoS-holder can generate the PoS-block), or withhold the PoW-block and discard it.

We note that, intuitively in Case 1, the malicious players cannot stop honest players from extending the chain-pairs; thus the chain growth property can be ensured. Now let's consider the total number of PoS-blocks from malicious players in Case 2 and in Case 3. If the number of PoS-blocks from honest players in Case 1 is larger than that from the malicious players in Case 2 and Case 3, we can also see that the common prefix property is ensured.

In Case 2 or Case 3, the malicious PoS-player may generate multiple PoS-blocks based on a single PoW-block. We remark here that this malicious strategy will bring no advantage to the attacker, since only *one* of the multiple PoS-blocks will be extended by honest PoW-miners.

As discussed above, $\hat{\alpha} = \alpha\tilde{\alpha}$ and $\hat{\beta} = \beta\tilde{\beta}$ are the collective probabilities of Case 1 and Case 2, respectively. We define them as the collective resources of the honest and malicious parties, respectively.

In our protocol, the malicious players are allowed to delay communication messages for at most Δ rounds. When the malicious players delay the communication messages, each honest player might not be able obtain its best chain-pair on time. As a consequence, honest miners may work on a wrong chain-pair during the delayed communication rounds. In our analysis, we thus use the *discounted version* of the computing/stake resource to calculate the probability that the honest players can generate a block in a round.

Chain growth. The malicious players may delay all of the communication messages from the honest players up to Δ rounds. Consider that to generate a block-pair, two hops are needed; The adversary can delay at most 2Δ rounds for a PoS-block generation. We use $\hat{\gamma}$ to denote the discounted collective honest resources where $\hat{\gamma} = \frac{\hat{\alpha}}{1+2\Delta\hat{\alpha}}$.

In the formal proof, we introduce a hybrid execution, formalizing the worst case communication delay. In the hybrid execution, the malicious players will not contribute to the chain growth; furthermore, the adversary will delay all communication messages from the honest players with the goal of stopping the chain growth as much as possible. When Case 1 occurs, the longest chain-pair that can be observed by all honest players, will increase by 1 block-pair (one PoW-block and one PoS-block). Note that the probability that Case 1 occurs in a round is $\hat{\gamma}$. Also note that the probability that Case 1 occurs in our protocol execution, will not be smaller than that in the worst case hybrid execution. Thus the chain growth rate is guaranteed by $\hat{\gamma}$.

Chain quality. Assume $\tilde{p}\tilde{n} \ll 1$. With high probability, at the same block height, there is at most one block-pair in Case 1 or Case 4. During any t consecutive rounds, the expected number of block-pairs generated in Case 1, is at least $\hat{\gamma}t$. Let θt denote the number of block-pairs generated in Case 4 during the t rounds, for some θ. Then we can have $0 \leq \theta \leq \beta\tilde{\alpha}$. The chain growth in the t round is $(\hat{\gamma} + \theta)t$. In addition, the expected number of block-pair that generated in Case 2 or Case 3 during t rounds, is at most $(\alpha+\beta)\tilde{\beta}t$. Therefore, the chain quality is at least $1-\frac{(\alpha+\beta)\tilde{\beta}+\theta}{\hat{\gamma}+\theta} \geq 1-\frac{(\alpha+\beta)\tilde{\beta}+\beta\tilde{\alpha}}{\hat{\gamma}+\beta\tilde{\alpha}}$.

Common prefix. Assume $2(\alpha+\beta)(\tilde{\alpha}+\tilde{\beta})\Delta \ll 1$. We can compute that, the probability that no new block-pair is generated in a round, is

$1-2(\alpha+\beta)(\tilde{\alpha}+\tilde{\beta})\Delta$. We can also compute that, the probability for honest players to generate *at least one* new block-pair in a round, is at least $\hat{\alpha}$. We can further argue that, the probability for honest players to generate *exactly one* new block-pair in a round, is $\hat{\alpha}(1-\hat{\alpha})$, which approximates to $\hat{\alpha}$, given that we assume $\hat{\alpha} \ll 1$.

After the publication of one block-pair in the system, if in the upcoming 2Δ rounds, there is no block-pair published, then all honest players will have the same best chain-pair, and their views will be convergent. The malicious players may generate blocks to achieve their goal of stopping the honest players to develop convergent views of the best chain-pair. However, based on our assumption, the malicious players cannot generate enough number of block-pairs to achieve this goal.

On adaptive corruption. In our protocol, the adversary can corrupt any player adaptively at any time. We note that in the first hop the adversary cannot predict which PoW-player will be able to find a solution to the PoW puzzle before the solution is published. Thus, adaptively corrupting PoW miners will not bring the adversary any extra advantage. Then in the second hop, the solution to the PoS puzzle consists of the (unique) signature from a PoS-player. Again, the adversary cannot predict which PoS-player will be elected before the the solution to the PoS puzzle is published. Similarly, the adaptive corruption strategy will not bring extra advantage.

References

1. A. Back. Hashcash—A denial of service counter-measure, 2002. Available at http://hashcash.org/papers/hashcash.pdf.
2. C. Badertscher, P. Gazi, A. Kiayias, A. Russell, and V. Zikas. Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability. In D. Lie, M. Mannan, M. Backes, and X. Wang, editors, *ACM CCS 2018*, pages 913–930. ACM Press, Oct. 2018.
3. I. Bentov, A. Gabizon, and A. Mizrahi. Currencies without proof of work. In *Bitcoin Workshop*, 2016.
4. I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld. Proof of activity: Extending bitcoin's proof of work via proof of stake. In *SIGMETRICS Perform. Eval. Rev.*, pages 34–37. ACM, 2014.
5. Bitcointalk. Proof of stake instead of proof of work, 2011. Online post by QuantumMechanic, available at https://bitcointalk.org/index.php?topic=27787.0.
6. R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, Jan. 2000.
7. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000. http://eprint.iacr.org/2000/067.

8. J. Chen, S. Gorbunov, S. Micali, and G. Vlachos. Algorand agreement: Super fast and partition resilient Byzantine agreement, 2018. Available at https://eprint.iacr.org/2018/377.
9. J. Chen and S. Micali. Algorand, 2017. Available at http://arxiv.org/abs/1607.01341.
10. A. Chepurnoy, T. Duong, L. Fan, and H.-S. Zhou. Twinscoin: A cryptocurrency via proof-of-work and proof-of-stake. In *Proceedings of the 2nd ACM Workshop on Blockchains, Cryptocurrencies, and Contracts*, pages 1–13. ACM, 2018.
11. CryptoManiac. Proof of stake, 2014. NovaCoin wiki. Available at https://github.com/novacoin-project/novacoin/wiki/Proof-of-stake/.
12. P. Daian, R. Pass, and E. Shi. Snow white: Robustly reconfigurable consensus and applications to provably secure proof of stake. In I. Goldberg and T. Moore, editors, *FC 2019*, volume 11598 of *LNCS*, pages 23–41. Springer, Heidelberg, Feb. 2019.
13. B. David, P. Gazi, A. Kiayias, and A. Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In J. B. Nielsen and V. Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 66–98. Springer, Heidelberg, Apr. / May 2018.
14. T. Duong, L. Fan, and H.-S. Zhou. 2-hop blockchain: Combining proof-of-work and proof-of-stake securely, 2016. Available at 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. L. Fan and H.-S. Zhou. A scalable proof-of-stake blockchain in the open setting (or, how to mimic Nakamoto's design via proof-of-stake), July 2017. Available at https://eprint.iacr.org/2017/656/.
19. 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.
20. Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 51–68. ACM, 2017.
21. D. Goodin. Bitcoin security guarantee shattered by anonymous miner with 51% network power, 2014. Available at http://arstechnica.com/.
22. A. Kiayias and G. Panagiotakos. Speed-security tradeoffs in blockchain protocols. Cryptology ePrint Archive, Report 2015/1019, 2015. http://eprint.iacr.org/2015/1019.
23. A. Kiayias and G. Panagiotakos. On trees, chains and fast transactions in the blockchain. In *International Conference on Cryptology and Information Security in Latin America*, pages 327–351. Springer, 2017.
24. A. Kiayias, A. Russell, B. David, and R. Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In J. Katz and H. Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 357–388. Springer, Heidelberg, Aug. 2017.

25. S. King and S. Nadal. PPCoin: Peer-to-peer crypto-currency with proof-of-stake, 2012. https://peercoin.net/assets/paper/peercoin-paper.pdf.
26. A. Lysyanskaya. Unique signatures and verifiable random functions from the DH-DDH separation. In M. Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 597–612. Springer, Heidelberg, Aug. 2002.
27. 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 2015*, pages 680–691. ACM Press, Oct. 2015.
28. S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. Available at https://bitcoin.org/bitcoin.pdf.
29. NXT whitepaper, 2014. https://www.dropbox.com/s/cbuwrorf672c0yy/NxtWhitepaper_v122_rev4.pdf.
30. R. Pass, L. Seeman, and a. shelat. Analysis of the blockchain protocol in asynchronous networks. In J. Coron and J. B. Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 643–673. Springer, Heidelberg, Apr. / May 2017.
31. R. Pass and E. Shi. The sleepy model of consensus. In T. Takagi and T. Peyrin, editors, *ASIACRYPT 2017, Part II*, volume 10625 of *LNCS*, pages 380–409. Springer, Heidelberg, Dec. 2017.
32. A. Sapirshtein, Y. Sompolinsky, and A. Zohar. Optimal selfish mining strategies in bitcoin. In J. Grossklags and B. Preneel, editors, *FC 2016*, volume 9603 of *LNCS*, pages 515–532. Springer, Heidelberg, Feb. 2016.
33. O. Schrijvers, J. Bonneau, D. Boneh, and T. Roughgarden. Incentive compatibility of bitcoin mining pool reward functions. In J. Grossklags and B. Preneel, editors, *FC 2016*, volume 9603 of *LNCS*, pages 477–498. Springer, Heidelberg, Feb. 2016.
34. 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.
35. P. Vasin. Blackcoin's proof-of-stake protocol v2, 2014. Available at http://blackcoin.co/blackcoin-pos-protocol-v2-whitepaper.pdf.

A Unique Signature Schemes

Unique signature schemes were introduced in [26], which consists of four algorithms, a randomized key generation algorithm KeyGen, a deterministic key verification algorithm KeyVer, a deterministic signing algorithm Sign, and a deterministic verification algorithm Verify. We expect for each verification key there exists only one signing key. We also expect for each pair of message and verification key, there exists only one signature. We have the following definition.

Definition 4. *We say* $(\mathsf{KeyGen}, \mathsf{KeyVer}, \mathsf{Sign}, \mathsf{Verify})$ *is a unique signature scheme, if it satisfies:*

Correctness of key generation: Honestly generated key pair can always be verified. More formally, it holds that

$$\Pr\left[(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{KeyGen}(1^\kappa) \ : \ \mathsf{KeyVer}(\mathrm{PK}, \mathrm{SK}) = 1\right] \geq 1 - \mathsf{negl}(\kappa)$$

Uniqueness of signing key: There does not exist two different valid signing keys for a verification key. More formally, for all PPT *adversary* $\mathcal{A}$*, it holds that*

$$\Pr\left[\begin{array}{l}(\mathrm{PK},\mathrm{SK}_1,\mathrm{SK}_2)\leftarrow\mathcal{A}(1^\kappa)\\ :\ \mathsf{KeyVer}(\mathrm{PK},\mathrm{SK}_1)=1\wedge\mathsf{KeyVer}(\mathrm{PK},\mathrm{SK}_1)=1\wedge\mathrm{SK}_1\neq\mathrm{SK}_2\end{array}\right]\leq\mathsf{negl}(\kappa)$$

Correctness of signature generation: For any message x*, it holds that*

$$\Pr\left[\begin{array}{l}(\mathrm{PK},\mathrm{SK})\leftarrow\mathsf{KeyGen}(1^\kappa);\sigma:=\mathsf{Sign}(\mathrm{SK},x)\\ :\ \mathsf{Verify}(\mathrm{PK},x,\sigma)=1\end{array}\right]\geq 1-\mathsf{negl}(\kappa)$$

Uniqueness of signature generation: For all PPT *adversary* $\mathcal{A}$*,*

$$\Pr\left[\begin{array}{l}(\mathrm{PK},x,\sigma_1,\sigma_2)\leftarrow\mathcal{A}(1^\kappa)\\ :\ \mathsf{Verify}(\mathrm{PK},x,\sigma_1)=1\wedge\mathsf{Verify}(\mathrm{PK},x,\sigma_2)=1\wedge\sigma_1\neq\sigma_2\end{array}\right]\leq\mathsf{negl}(\kappa)$$

Unforgeability of signature generation: For all PPT *adversary* $\mathcal{A}$*,*

$$\Pr\left[\begin{array}{l}(\mathrm{PK},\mathrm{SK})\leftarrow\mathsf{KeyGen}(1^\kappa);(x,\sigma)\leftarrow\mathcal{A}^{\mathsf{Sign}(\mathrm{SK},\cdot)}(1^\kappa)\\ :\ \mathsf{Verify}(\mathrm{PK},x,\sigma)=1\wedge(x,\sigma)\notin Q\end{array}\right]\leq\mathsf{negl}(\kappa)$$

where Q *is the history of queries that the adversary* $\mathcal{A}$ *made to signing oracle* $\mathsf{Sign}(\mathrm{SK},\cdot)$*.*