2018

BLOCKCHAIN SCALABILITY AND SECURITY

Tuyet Duong

DISSERTATION BLOCKCHAIN SCALABILITY AND SECURITY

A Dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at Virginia Commonwealth University.

by

TUYET DUONG

Bachelor of Science, Vietnam National University, Vietnam - Dates

Director: Dissertation Hong-Sheng Zhou,

Assistant Professor, Department of Computer Science

Virginia Commonwealth University

Richmond, Virginia

July, 2018

PhD advisor (Committee Chairperson):

Dr. Hong-Sheng Zhou

Assistant Professor, Computer Science Department, Virginia Commonwealth University

Committee Members:

Dr. Foteini Baldimtsi

Assistant Professor, Computer Science Department, George Mason University

Dr. Thang N. Dinh

Assistant Professor, Computer Science Department, Virginia Commonwealth University

Dr. Carol Fung

Assistant Professor, Computer Science Department, Virginia Commonwealth University

Dr. Bingsheng Zhang

Assistant Professor, School of Computing and Communications, Lancaster University

Acknowledgements

First and foremost, I would like to express my immense appreciation to my advisor, Dr. Hong-Sheng Zhou, for his guidance and inspiration. I am very grateful for having Hong-Sheng as an advisor. I also appreciate all of the advices that he has given me over the years. This thesis would not have been possible without him.

I sincerely thank my committee members, Dr. Foteini Baldimtsi, Dr. Thang N. Dinh, Dr. Carol Fung and Dr. Bingsheng Zhang for all their valuable feedbacks and insightful advices in my research field.

I would like to thank Dr. Lei Fan and Alexander Chepurnoy for many years of collaboration. I always enjoyed working with them.

Finally, my sincere acknowledgment is to my husband, Dr. Hung T. Nguyen, for always being there for me and support me in every step of this journey.

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

LIST OF FIGURES

Abstract

DISSERTATION BLOCKCHAIN SCALABILITY AND SECURITY

By Tuyet Duong

A Dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at Virginia Commonwealth University.

Virginia Commonwealth University, 2018.

Director: Dissertation Hong-Sheng Zhou,

Assistant Professor, Department of Computer Science

Cryptocurrencies like Bitcoin have proven to be a phenomenal success. The underlying techniques hold huge promise to change the future of financial transactions, and eventually the way people and companies compute, collaborate, and interact. At the same time, the current Bitcoin-like proof-of-work based blockchain systems are facing many challenges. In more detail, a huge amount of energy/electricity is needed for maintaining the Bitcoin blockchain. In addition, their 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 will fail when this assumption is broken.

This research proposes novel blockchain designs to address the challenges. We first propose a novel blockchain protocol, called 2-*hop blockchain*, by combining proof-of-work and proof-of-stake mechanisms. That said, even if the adversary controls more than 50% computing power, the honest players still have the chance to defend the blockchain via honest stake. Then we revise and implement the design to obtain a practical cryptocurrency system called *Twinscoin*. In more detail, 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.

CHAPTER 1

INTRODUCTION

1.1 Motivation

Cryptocurrencies like Bitcoin [Nak08] have proven to be a phenomenal success. The underlying techniques hold a huge promise to change the future of financial transactions, and eventually our way of computation and collaboration. At the heart of the Bitcoin system is a global public distributed ledger, called *blockchain*, that records transactions between *users* in consecutive time windows. The blockchain is maintained by a peer-to-peer network of nodes called Bitcoin *miners* via the so-called proof-of-work (PoW) mechanism: in each time window, cryptographic puzzles (also called proof-of-work puzzles [DN93; Bac02]) are generated, and all miners are encouraged/incentivized to solve the puzzles; the first miner who finds a puzzle solution is allowed to extend the blockchain with a block of transactions, and at the same time he can collect a reward. It is easy to see that the more computing power a miner invests, the better his chances are at solving a puzzle first.

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 unique "easy to join/leave" feature along with a smart incentive strategy help the system "absorb"[1] a huge amount of computing resources over the past several years. Intuitively, the security of blockchain is backed up by a significant network of physical computing resources.

This intuition has recently been investigated in academia. For example, Garay et

[1] You may say the system *consumes/wastes* a huge amount of computing resources.

al. [GKL15] and then Pass et al. [PSS17a] looked into the "core" of the Bitcoin system, the *Nakamoto consensus protocol*; they showed that, assuming the majority of mining power in the Bitcoin system is controlled by the honest miners, then the Nakamoto consensus indeed satisfies several important security properties as defined in their cryptographic models. On the other hand, if this assumption does not hold, then the security of the Bitcoin system cannot be guaranteed.

This assumption has been seriously challenged: the mining power can be *dangerously distributed* in the system. For example, in 2014, the mining pool GHash.io exceeded 50% of the computational power in Bitcoin [Goo14]. Currently, top mining pools including F2Pool, AntPool, BTCC and BW, are all in China; they collectively control about 60% mining power. It is not clear if those mining pools collude. Very recently, one mining pool has had 50% mining power in Zcash [Tor]. Efforts have been made to address this crisis. In [Mil+15], novel ideas are introduced to discourage the formation of mining pools. However, it is not clear in practice how to utilize these ideas to protect the system if the adversary controls the majority of mining power. We here ask the following question:

> *Is that possible to strengthen Bitcoin-like system so that it can be secure even when the adversary controls more than 50% computing power in the system?*

1.2 Our Considerations

Before giving a proper solution to the above question, we need to understand Nakamoto's design more. Only then, we might be able to mimic Nakamoto's footprint and push this line of design further.

Leveraging the power of virtual resources in the system. Ideally, we would like to construct Bitcoin-like blockchain which is secure against a very strong adversary even from the beginning. However, it is easy to see that cryptocurrency systems are very frag-

ile in their early stage. It will be extremely difficult, if not impossible, to "grow" a stable Bitcoin-like blockchain if the adversary controls the majority of computing power at the very beginning. In this work, we consider how to make an already mature cryptocurrency system such as the current Bitcoin system to be more robust in the sense that the system remains stable even the adversary controls the majority of computing power. As mentioned, Bitcoin already "absorbed" a huge amount of honest computing power; note that these physical resources have been converted into "virtual resources", i.e., the coins. It is fairly reasonable to assume the coins are *nicely distributed* in the system and most of them are controlled by honest users. A natural way to go is to use this huge amount of honest *virtual* resources as a buffer to defend against the adversary who can dominate the network of computing power.

The difference between physical resources and virtual resources. It is definitely desirable to utilize the power of virtual resources to secure a blockchain. If successful, the new system will be "green" in the sense that it does not require a huge amount of *physical resources*, which cannot be recycled, to back up its security. Attempts have been made. For example, proof-of-stake (PoS) mechanisms have been widely discussed in the cryptocurrency community. In a nutshell, proof-of-stake mechanisms for consensus require protocol players to prove ownership of a certain amount of virtual resources. Only those that can provide such a proof can participate in maintaining the blockchain. However, it is not clear how to construct an open blockchain which can scale to a large number of network nodes (as in Bitcoin), via any proof-of-stake mechanism. At a very intuitive level, virtual resources, which proof-of-stake mechanisms are based on, are very difficult to manage in a practical protocol. SS This intuition has been confirmed recently by the concurrent works of [CM17; Kia+17; DPS17]); there, very interesting and non-trivial attempts have been made to construct provably secure, scalable blockchains via pure proof-of-stake. Unfortunately, these solutions cannot scale to a large number of network nodes in an *open*

setting where participants can freely join or leave the system any time they want. In more details, a secure bootstrapping mechanism (i.e., majority voting) is required for ensuring new participants can securely join the system. However, this "bootstrapping" cannot scale to a large network. See Section 1.3 (Related Work) for more discussion. In one word, it seems it is extremely difficult to mimic Nakamoto's footprint via virtual resources *only*.

On the other hand, physical resources are relatively easier to manage. Indeed, Nakamoto demonstrates to us an amazingly practical protocol via the proof-of-work mechanism to manage physical computing resource effectively. Alternative physical resources such as a publicly available random beacon or secure hardware can also allow us to construct fast protocols. See Section 1.3 (Related Work) for more discussion about open blockchain via alternative physical resources.

The practical elegance of using random oracle for cryptocurrency. Although fast blockchain protocols can be constructed via physical resources such as random beacon or trusted hardware, there is a significant drawback in these solutions. That is, the trapdoor information of the system is possessed by a single party. Currently, it is not clear how to eliminate such trapdoor information. Interestingly, blockchain via the proof-of-work mechanism can avoid such issue in practice: the underlying proof-of-work puzzles can be based on hash functions, and the security of the system can be argued in the so-called random oracle model. Theoretical cryptographers may criticize random oracle methodology since it is not sound [CGH98]. However, random oracles do enable an elegant solution to open blockchains in practice.[2]

Additional considerations. There are many reasons that Bitcoin has become a successful system. For example, proof-of-work puzzles Bitcoin is a *divisible* e-cash system [OO92;

[2] We note that Nakamoto's design is consistent with the folklore wisdom of a "nothing up my sleeve number" [Wikd] which has been widely used in practical cryptographic designs.

Oka95]. Besides the points we discussed above, to design a practical blockchain, we in general should avoid heavy cryptographic tools, and use only standard cryptographic primitives such as hash functions and digital signature schemes. Futhermore, the design should be simple. Finally, the provable security approach should be taken to develop blockchain techniques. We eventually should move these powerful blockchain techniques from an art to a science.

1.3 Related work

1.3.1 Closely related work on combining proof-of-work and proof-of-stake

Proof-of-activity. The idea of combining proof-of-work and proof-of-stake has been studied in [KN12; Cry14; Ben+14], and the latest one, proof-of-activity (PoA) [Ben+14] is closest to our work. Although these studies showed that their protocols are secure against some classes of attacks, they *do not provide any formal security model and analysis relying on precise definitions*. We note that the earlier proposal [Ben+14] is different from our result on both security analysis and the construction.

First, the leader election mechanism in PoA design is predictable. More specifically, miners generate empty block header and then the header will be mapped to N stakeholders, where $N \geq 1$ and typically N is set as 3. They use deterministic mapping function to elect exact N stake holders (see item 3 Figure 2 in [Ben+14]). That said, the *adaptive* adversary can easily predict future leaders and corrupt them in advance. As a result, the protocol will not be secure even if the adversary only controls the minority of stake or computing power. In contrast, our proposal naturally mimics Nakamoto's footprint, and it is the first *adaptively secure* proof-of-work / proof-of-stake hybrid consensus protocol.

In addition, we note that the PoA proposal in [Ben+14] will not immediately give us a divisible cryptocurrency [OO92; Oka95] unless extra efforts are paid.

Casper. Casper [BG17] is also a hybrid proof-of-work/proof-of-stake protocol. Informally, stakeholders can lock their stakes to be validators, and then vote for each block whose height is an exact multiple of 100 (i.e., checkpoint). However, this voting-based protocol only scales to a small number of validators. As the result, the protocol will eventually be centralized and the control is in the hand of a small number of participants.

In contrast, our proposal mimics Nakamoto's footprint so that anyone who has coins can join the proof-of-stake.

Decred. Another similar proposal is Decred [Jep15]. This proposal needs an additional trusted entity to produce tickets for participants. Informally, stakeholders will purchase tickets and check if their tickets are selected. Then the stakeholder with the selected ticket is qualified to produce a new block. Note that, he selection process is pseudo-randomized where the pseudo-randomness is from the proof-of-work chain.

This design is very similar to our proposal in the sense that the pseudo-randomness for the elections of stakeholders is from the proof-of-work chain. However, no trusted party is required in our protocol.

1.3.2 Proof-of-stake

Very recently, concurrent works (e.g., [CM17; Kia+17; PS17; DPS17; Dav+17; FZ17; Gil+17]) have made very interesting attempts to construct provably secure, scalable blockchains via PoS only. We note that, our work is fundamentally different from these pure PoS proposals in the following sense: ours is based on the majority of honest *collective* resources (including both stake and work), while these pure PoS proposals are based on the honest majority of stake. Combining stake with work will bring us several benefits. For example, in these pure PoS proposals, the stake holders must have their stakes registered much earlier before participating in the mining process. In addition, some of

these PoS protocols may suffer from rational attacks, including nothing at stake attacks and selfish mining attacks; see [Coh+] for discussions. Our design does not suffer from these attacks.

We already discussed recent effort on provably secure pure proof-of-stake (PoS) proposals [CM17; Kia+17; DPS17] above. Before these recent rigorous efforts, using virtual resources (i.e., stake) to construct cryptocurrency has been intensively considered in the Bitcoin community. In a nutshell, the PoS mechanisms for consensus require a protocols' players to prove ownership of virtual resources. Only those that can provide such proofs can participate in maintaining the protocol's blockchain, their ability to do so is proportional to the stake owned. Since the inception of the idea in an online forum [Bit11], several variants of PoS that have been proposed and implemented in real cryptocurrencies including [NXT14; Kwo14; Vas14; BGM16; KN12; Cry14]. In general, a PoS proof system simulates random leader election, where each participants' chance of being elected is proportional to the amount of stake that they control in the system. The chosen leader proves that they were elected by providing a cryptographic proof (a digital signature) that they own a specific share of stake. We note that, these proposals lack of rigorous security analysis.

1.3.3 Cryptocurrency and security analysis

Anonymous digital currency was introduced by Chaum [Cha82] in the early 1980s. The first decentralized currency system, Bitcoin [Nak08], was launched about 30 years later, by incentivizing a set of players to solve moderately-hard cryptographic puzzles (also called proofs-of-work puzzles [DN93; Bac02]). Recently, the security of Bitcoin system has been analyzed in the rational setting, e.g., [ES14; Eya15; Nay+15; Kia+16; SSZ16; Sch+16], and also in cryptographic setting [GKL15; PSS17a; SZ15; KP15; KP16]. Several important security properties, *common prefix*, *chain growth*, and *chain quality*, have been

considered for secure blockchain protocols. The common prefix and chain quality properties were originally formalized by Garay et al. [GKL15]. The chain growth property was first formally defined by Kiayias et al. [KP15]. The common prefix property was later strengthened by Pass et al. [PSS17a]. In our study, we adopt the stronger variant of the common prefix property by Pass et al. [PSS17a] together with the chain quality and chain growth from [KP15; GKL15].

Very recently, Vassilis et al. [Bad+17] propose a simulation-based analysis for Bitcoin protocol. More specifically, they put forth a universally composable treatment of the Bitcoin protocol, formalizing ledger functionality and leader election (lottery) via proof-of-work, and show that the three security properties can be implied by the ledger functionality.

1.3.4 More alternative consensus techniques

Similar to PoW, alternative consensus techniques via different physical resources have been considered to replace computing power. For example, the physical storage resource is used in [Par+15; Mil+14; Abu+17]. Between the use of space/memory and the use of time, are *proofs of space time* introduced in [MO16]. This is a hybrid proof system utilizing both computational and space resources. Finally, trusted hardware has been used for constructing blockchain protocols in [Int16b; Zha+17].

CHAPTER 2

BACKGROUND

2.1 Cryptography

2.1.1 Notation

Throughout this thesis, we denote $\kappa \in \mathbb{N}$ as the security parameter. We write $\{0,1\}^\kappa$ as the set of binary strings of length κ and $\{0,1\}^*$ to indicate the set of all finite, binary strings.

Negligible function. We often need to show that cryptographic schemes can be broken with only very small probability that goes to 0 faster than any inverse polynomial. We call such probabilities, *negligible*. The formally definition of negligible function is as follows.

Definition 1. *We call a function* negl *negligible if for every polynomial* poly*, there exists an* N *such that for all* $\kappa > N$*,* $\mathsf{negl}(\kappa) < \frac{1}{\mathrm{poly}(\kappa)}$

2.1.2 Probabilistic inequalities

Theorem 1 (Bernoulli's inequality)**.** *For* $q \geq 1$ *and* $0 \leq p \leq 1$*,* $(1-p)^q \geq 1 - pq$

Theorem 2 (Chernoff bounds)**.** *Suppose* $\{X_i : i \in [n]\}$ *are mutually independent Boolean random variables, with* $Pr[X_i = 1] = p$*, for all* $i \in [n]$*. Let* $X = \sum_{i=1}^{n} X_i$ *and* $\mu = pn$*. Then, for any* $\delta \in (0,1]$*,*

$$\Pr[X \leq (1-\delta)\mu] \leq e^{\{}\delta^2\mu/2\} \text{ and } \Pr[X \geq (1+\delta)\mu] \leq e^{\{}\delta^2\mu/3\}$$

2.1.3 Cryptographic Primitives

2.1.3.1 Collision-resistant hash functions

Generally, hash functions are used to compress any arbitrary-length strings into shorter strings. More precisely, each hash function H of size κ has a table of length κ, then for each input string x, it will store x with the corresponding shorter string $\mathsf{H}(x)$ in the table. A hash function should be "good" in the sense that it only yields as few as possible pairs of the distinct input x and x' with the same hash strings, i.e., $\mathsf{H}(x) = \mathsf{H}(x')$ (called collision). Also, collision resistance does not mean that no collisions exist, simply that they are hard to find.

Informally, a function H is *collision resistant* if it is infeasible for any probabilistic polynomial-time algorithm to find a collision in H. In other words, collisions should be very hard to find. In [LK14], the formal definition of a hash function is as follows.

Note that, for formal presentation, we consider a family of hash functions indexed by a key s. However, different from cryptographic keys, the key s is 1) not kept secret, and 2) will be generated by an algorithm Gen.

Definition 2. *A* hash function *is a pair of probabilistic polynomial time algorithms* $(\mathsf{Gen}, \mathsf{H})$ *satisfying the following:*

- Gen *is a probabilistic algorithm which takes as input a security parameter* 1^κ *and outputs a key* s*. We assume that* 1^κ *is implicit in* s*.*
- *There exists a polynomial* ℓ *such that* H *takes as input a key* s *and a string* $x \in \{0,1\}^*$ *and outputs a string* $\mathsf{H}^s(x) \in \{0,1\}^{\ell(\kappa)}$ *(where* κ *is the value of the security parameter implicit in* s*)*

Definition 3. *A hash function* $\Pi = (\mathsf{Gen}, \mathsf{H})$ *is* collision-resistant *if for all polynomial-time adversaries* $\mathcal{A}$*, there exists a negligible function* negl *such that*

$$\Pr[\mathsf{Hash\text{-}coll}_{\mathcal{A},\Pi}(\kappa) = 1] \leq \mathsf{negl}(\kappa)$$

where the experiment $\mathsf{Hash\text{-}coll}_{\mathcal{A},\Pi}(\kappa)$ *is defined as follows.*

The collision-finding experiment $\mathsf{Hash\text{-}coll}_{\mathcal{A},\Pi}(\kappa)$**:**

- *A key* s *is generated by running* $\mathsf{Gen}(1^\kappa)$.
- *The adversary* $\mathcal{A}$ *is given* s *and outputs* x, x'*. (If* Π *is a fixed length hash function for inputs of length* $\ell'(\kappa)$ *then we require* $x, x' \in \{0,1\}^{\ell'(\kappa)}$*.)*
- *The output of the experiment is defined to be 1 if and only if* $x \neq x'$ *and* $\mathsf{H}^s(x) = \mathsf{H}^s(x')$*. In such a case we say that* $\mathcal{A}$ *has found a collision.*

We also emphasize that hash functions used in practice are often *unkeyed*. In other words, a fixed hash function H is defined, and there is no longer any notion of a Gen algorithm. However, it is important to include keys theoretically since it is not clear how to define a meaningful notion for the unkeyed hash functions [LK14],

2.1.3.2 Digital signature

Digital signature is a tool to preserve the integrity of messages. More concretely, any signer with an established public and private keys can sign message in such a way that other parties can use the public key to verify that the message is originated from the signer who has the keys. A signature scheme is "secure" if no party can force a valid signature without having the secret key. In [LK14], the formal definition of a digital scheme is as follows

Definition 4. *A signature scheme is a tuple of three probabilistic polynomial-time algorithms* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *satisfying the following:*

1. *The key-generation algorithm* Gen *takes as input a security parameter* 1^κ *and outputs*

a pair of keys $(\mathsf{vk}, \mathsf{sk})$*. These are called the public key and the private key, respectively. We assume for convenience that* vk *and* sk *each have length at least* κ*, and that* κ *can be determined from* vk, sk*.*

2. *The signing algorithm* Sign *takes as input a private key* sk *and a message* $m \in \{0,1\}^*$*. It outputs a signature* σ*, denoted as* $\sigma \leftarrow \mathsf{Sign}_{\mathsf{sk}}(m)$*.*

3. *The deterministic verification algorithm* Verify *takes as input a public key* vk*m a message* m*, and a signature* σ*. It outputs a bit* b*, with* $b = 1$ *meaning valid and* $b = 0$ *meaning invalid. We write this as* $b := \mathsf{Verify}_{\mathsf{vk}}(\sigma, m)$*.*

It is required that for every κ*, every* vk, sk *output by* $\mathsf{Gen}(1^\kappa)$*, and every* $m \in \{0,1\}^*$*, it holds that*

$$\mathsf{Verify}_{\mathsf{vk}}(m, \mathsf{Sign}_{\mathsf{sk}}(m)) = 1$$

If $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *is such that for every* $(\mathsf{vk}, \mathsf{sk})$ *output by* $\mathsf{Gen}(1^\kappa)$*, algorithm* $\mathsf{Sign}_{\mathsf{sk}}$ *is only defined for messages* $m \in \{0,1\}^{\ell(\kappa)}$ *(and* $\mathsf{Verify}_{\mathsf{vk}}$ *outputs 0 for* $m \notin \{0,1\}^{\ell(\kappa)}$*, then we say that* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *is a signature scheme for messages of length* $\ell(n)$*).*

A signature scheme is used as follows: a singer S runs $\mathsf{Gen}(1^\kappa)$ to get the public and private keys $(\mathsf{vk}, \mathsf{sk})$. Then publicly announce the public key vk belonging to the signer S. If S needs to send a message m to other party, it computes $\sigma \leftarrow \mathsf{Sign}_{\mathsf{sk}}(m)$ and gives (m, σ) to the receiver. The receiver knowing the public key vk can verify the authenticity of m by checking whether $\mathsf{Verify}_{\mathsf{vk}}(\sigma, m) = 1$.

Intuitively, we want to ensure that no malicious party can force a valid signature and message pair without knowing the corresponding secret key. In other words, only the owner of the key pair can produce the valid signatures. Let $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ be a signature scheme. The security of the signature scheme is formally defined as follows.

Definition 5. *A signature scheme* $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *is existentially unforgeable under an adaptive chosen-message attack if for all probabilistic polynomial-time adversaries* $\mathcal{A}$*, then there exists a negligible function* negl *such that:*

$$\Pr[\mathsf{Sig\text{-}force}_{\mathcal{A},\Pi}(\kappa) = 1] \leq \mathsf{negl}(\kappa)$$

where the signature experiment $\mathsf{Sig\text{-}force}_{\mathcal{A},\Pi}(\kappa)$ *is defined as follows:*

- $\mathsf{Gen}(1^\kappa)$ *is run to obtain keys* $(\mathsf{vk}, \mathsf{sk})$*.*
- *Adversary* $\mathcal{A}$ *is given* vk *and oracle access to* $\mathsf{Sign}_{\mathsf{sk}}(\cdot)$*. (This oracle returns a signature* $\mathsf{Sign}_{\mathsf{sk}}(m)$ *for any message* m *of the adversary's choice.) The adversary then outputs* (m, σ)*. Let* $\mathcal{Q}$ *denote the set of messages whose signatures were requested by* $\mathcal{A}$ *during its execution.*
- *The output of the experiment is defined to be 1 if and only if (1)* $\mathsf{Verify}_{\mathsf{vk}}(m, \sigma) = 1$*, and (2)* $m \notin \mathcal{Q}$*.*

Unique signature scheme. Unique signature scheme was introduced in [Lys02], which consists of four algorithms, a randomized key generation algorithm $\mathsf{uKeyGen}$, a deterministic key verification algorithm $\mathsf{uKeyVer}$, a deterministic signing algorithm uSign, and a deterministic verification algorithm $\mathsf{uVerify}$. 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 6. *We say* $(\mathsf{uKeyGen}, \mathsf{uKeyVer}, \mathsf{uSign}, \mathsf{uVerify})$ *is a strengthened unique signature scheme, if it satisfies:*

Correctness of key generation: Honestly generated key pair can always be verified. More for-

mally, it holds that

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

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

$$\Pr\left[\begin{array}{l}(PK, SK_1, SK_2) \leftarrow \mathcal{A}(1^\kappa)\\ :\ \mathsf{uKeyVer}(PK, SK_1) = 1 \wedge \mathsf{uKeyVer}(PK, SK_1) = 1 \wedge SK_1 \neq SK_2\end{array}\right] \leq \mathsf{negl}(\kappa)$$

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

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

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

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

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

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

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

Instantiations for the unique signature scheme. Efficient instantiations can be found in literature. For example, the well-known BLS signature [BLS01] can be a good candidate.

2.1.4 The standard simulation paradigm

We here provide a (tailored) review of the standard simulation paradigm [GMW87]. We remark that this section follows closely Canetti's elegant work [Can00b, Sections 3,4] but we restrict it in the standalone setting; furthermore, protocol composition has been excluded since this is not our focus.

2.1.4.1 ITM

For the readers' convenience, we use the following ITM related definitions from Canetti's work [Can00b].

Definition 7. *An interactive Turing machine (ITM) M is a Turing machine with the following augmentations:*
Special tapes *(i.e., data structures):*

- *An identity tape. The contents of this tape is interpreted as two strings. The first string contains a description, using some standard encoding, of the program of M (namely, its transition function). We call this description the code of M. The second string is called the identity of M. The identity of M together with its code is called the extended identity of M. This tape is "read only". That is, M cannot write to this tape.*
- *An outgoing message tape. Informally, this tape holds the current outgoing message generated by M, together with sufficient addressing information for delivery of the message.*
- *Three externally writable tapes for holding inputs coming form other computing devices (or, processes):*
  - *An input tape, representing inputs from the "calling program(s)" or external user.*
  - *An incoming communication tape, representing information coming from other*

programs within the same "protocol instance".

  - *A subroutine output tape, representing outputs coming from programs or modules that were created as "subroutines" of the present program.*

- *A one-bit activation tape. Informally, this tape represents whether the ITM is currently "in execution".*

New instructions:

- *An external write instruction. Roughly, the effect of this instruction is that the message currently written on the outgoing message tape is possibly written to the specified tape of the machine with the identity specified in the outgoing message tape.*
- *A read next message instruction. This instruction specifies a tape out of* { *input, incoming communication, subroutine output* }*. The effect is that the reading head jumps to the beginning of the next message.*

Executing Systems of ITMs. We specify the mechanics of executing a system that consists of multiple ITMs. However, some significant differences from that sketch do exist. First, here we make an explicit distinction between an ITM, which is a "static object", namely an algorithm or a program, and an ITM instance (ITI), which is a "run-time object", namely an instance of a program running on some specific data. In particular, the same program (ITM) may have multiple instances (ITIs) in an execution of a system.

Second, the model provides a concrete mechanism for addressing ITIs and exchanging information between them. The mechanism specifies how an addresee is determined, what information the recipient obtains on the sender, and the computational costs involved.

Third, the model allows the number of ITIs to grow dynamically in an unbounded way as a function of the initial parameters, by explicitly modeling the "generation" of new

ITIs. Furthermore, new ITIs may have dynamically determined programs. Here the fact that programs of ITMs can be represented as strings plays a central role.

Fourth, we augment the execution model with a control function, which regulates the transfer of information between ITIs. Specifically, the control function determines which "external write" instructions are "allowed". This added construct provides both clarity and flexibility to the execution model: All the model restrictions are expressed explicitly and in "one place." Furthermore, it is easy to define quite different execution models simply by changing the control function.

Writing to a tape of another ITI and invoking new ITIs. The mechanism that allows communication between ITIs is the external write instruction. The same instruction is used also for invoking new ITIs. More specifically, the effect of an external write instruction is the following. Let $\mu = (M, id)$ denote the ITI which performs the external write transition, The current contents of the outgoing message tape is interpreted (using some standard encoding) as consisting of μ, followed by an extended identity $\mu' = (M', id')$ of a "target ITI", a tape name out of {input, incoming communication, subroutine output }, and a string m called the message. Then:

1. If the control function *C*, applied to the current execution prefix, does not allow μ to write to the specified tape of μ' (i.e., it returns a disallow value) then the instruction is ignored.

2. If *C* allows the operation, and an ITI $\mu'' = (M''; id'')$ with identity $id'' = id'$ currently exists in the system (namely, one of the past configurations in the current execution prefix has identity id'), then:

   (a) If the target tape is the incoming communication tape, then the message m is written on the incoming communication tape of μ'', starting at the next blank

space. (That is, a new configuration of μ'' is generated. This configuration is the last configuration of μ'' in this execution, with the new information written on the incoming communication tape.) It is stressed that the code M'' of μ'' need not equal the code M' specified in the external write request.

This convention has the effect that an ITI does not necessarily know the code of the ITI it sends messages to using the communication tape. The recipient ITI learns neither the identity nor the code of the writing ITI. (Of course, this information may be included in the message itself, but the model provides no guarantees regarding the authenticity of this information.) The intuitive goal is to capture the effect of standard communication over an untrusted medium, such as a communication network.

(b) If the target tape is the input tape or subroutine output tape, and $M'' = M'$, then the specified message is copied to the specified tape of μ'', along with the code and identity of μ. If $M' \neq M''$ then the message is not copied and μ transitions to a special error state.

This convention has the effect that an ITI can verify the code of the ITI to whom it provides input or subroutine output. Furthermore, the recipient of an input or subroutine output knows both the identity and the code of the writing ITI. The intuitive goal here is to capture the effect of communication between processes within a trusted computing environment that allows verification of the code of receiver and sender.

3. If C allows the operation, and no ITI with identity id' exists in the system, then a new ITI μ' with code M' and identity id' is invoked. That is, a new configuration is generated, with code M', the value id' written on the identity tape, and a sufficiently long random string is written on the random input tape. Once the new ITI

is invoked, the external-write instruction is carried out as in Step 2. In this case, we say that μ invoked μ'.

2.1.4.2 Simulation paradigm

The basic model of execution. Following [Can01; Can00b], a protocol is represented as interactive Turing machines (ITMs), each of which represents the program to be run by a participant. Specifically, an ITM has several tapes that can be written to by other ITMs: the *input* and *output* tapes model the inputs from and the outputs to "father" programs or I/O callers, the *incoming communication* tapes and *outgoing communication* tapes model messages received from and to be sent to the network. It also has an *identity tape* that cannot be written to by the ITM itself. The identity tape contains the program of the ITM (in some standard encoding) plus additional identifying information specified below. Adversarial entities are also modeled as ITMs.

We distinguish between ITMs (which represent static objects, or programs) and *instances of ITMs*, or *ITI*s, that represent interacting processes in a running system. Specifically, an ITI is an ITM along with an identifier that distinguishes it from other ITIs in the same system. Since we consider a single session, the identifier only consists of a *party identifier (PID)* that distinguishes among the parties in a protocol instance. Typically the PID is also used to associate ITIs with "parties", or clusters, that represent some administrative domains or physical computers.

The model of computation consists of a number of ITIs that can write on each other's tapes in certain ways (specified in the model). The identifier PID is a unique identifier of the ITI in the system. With one exception (discussed within) we assume that all ITMs are probabilistic polynomial time.

Defining security of protocols. Protocols that securely carry out a given task are defined in three steps, as follows. First, the process of executing a protocol in an adversarial environment is formalized. Next, an "ideal process" for carrying out the task at hand is formalized. The parties have access to an "ideal functionality," which is essentially an incorruptible "trusted party" that is programmed to capture the desired functionality of the task at hand. A protocol is said to securely realize an ideal functionality if the process of running the protocol amounts to "emulating" the ideal process for that ideal functionality. Below we overview the model of protocol execution (called the *real-world model*), the ideal process, and the notion of protocol emulation.

The model for protocol execution. The model of computation consists of the parties running an instance of a protocol π, a network adversary $\mathcal{A}$ that controls the communication among the parties, and an *environment* $\mathcal{Z}$ that controls the inputs to the parties and sees their outputs. The execution consists of a sequence of *activations,* where in each activation a single participant (either $\mathcal{Z}$, $\mathcal{A}$, or some other ITM) is activated, and may write on a tape of at most *one* other participant, subject to the rules below. Once the activation of a participant is complete (i.e., once it enters a special waiting state), the participant whose tape was written on is activated next. (If no such party exists then the environment is activated next.)

The environment is given an external input z and is the first to be activated. In its first activation, the environment invokes the adversary $\mathcal{A}$, providing it with some arbitrary input. The environment can from now on invoke (namely, provide input to) only ITMs that consist of a single instance of protocol π. That is, all the ITMs invoked by the environment must have the same SID and the code of π.

Once the adversary is activated, it may read its own tapes and the outgoing communication tapes of all parties. It may either *deliver* a message to some party by writing

this message on the party's incoming communication tape or report information to $\mathcal{Z}$ by writing this information on the subroutine output tape of $\mathcal{Z}$. For simplicity of exposition, in the rest of this paper we assume authenticated communication; that is, the adversary may deliver only messages that were actually sent.

Once a protocol party (i.e., an ITI running π) is activated, either due to an input given by the environment or due to a message delivered by the adversary, it follows its code and possibly writes a local output on the subroutine output tape of the environment, or an outgoing message on the adversary's incoming communication tape. The protocol execution ends when the environment halts. The output of the protocol execution is the output of the environment. Without loss of generality we assume that this output consists of only a single bit.

Let $\mathsf{REAL}_{\mathcal{A},\Pi,\mathcal{Z}}(\kappa, z, r)$ denote the output of the environment $\mathcal{Z}$ when interacting with parties running protocol π on security parameter κ, input z and random input $r = r_{\mathcal{Z}}, r_{\mathcal{A}}, r_1, r_2, ...$ as described above (z and $r_{\mathcal{Z}}$ for $\mathcal{Z}$; $r_{\mathcal{A}}$ for $\mathcal{A}$, r_i for party P_i). Let $\mathsf{REAL}_{\mathcal{A},\Pi,\mathcal{Z}}(\kappa, z)$ denote the random variable describing $\mathsf{REAL}_{\mathcal{A},\Pi,\mathcal{Z}}(\kappa, z, r)$ when r is uniformly chosen. Let $\mathsf{REAL}_{\mathcal{A},\Pi,\mathcal{Z}}$ denote the ensemble $\{\mathsf{REAL}_{\mathcal{A},\Pi,\mathcal{Z}}(\kappa, z)\}_{\kappa\in\mathbb{N},z\in\{0,1\}^*}$.

Remark 1. *We note that in the rest of this research, we write* EXEC *for any execution including real or ideal execution.*

Ideal functionalities and ideal protocols. Security of protocols is defined via comparing the protocol execution to an *ideal protocol* for carrying out the task at hand. A key ingredient in the ideal protocol is the *ideal functionality* that captures the desired functionality, or the specification, of that task. The ideal functionality is modeled as another ITM (representing a "trusted party") that interacts with the parties and the adversary. More specifically, in the ideal protocol for functionality $\mathcal{F}$ all parties simply hand their inputs to an ITI running $\mathcal{F}$. (We will simply call this ITI $\mathcal{F}$. In addition, $\mathcal{F}$ can interact

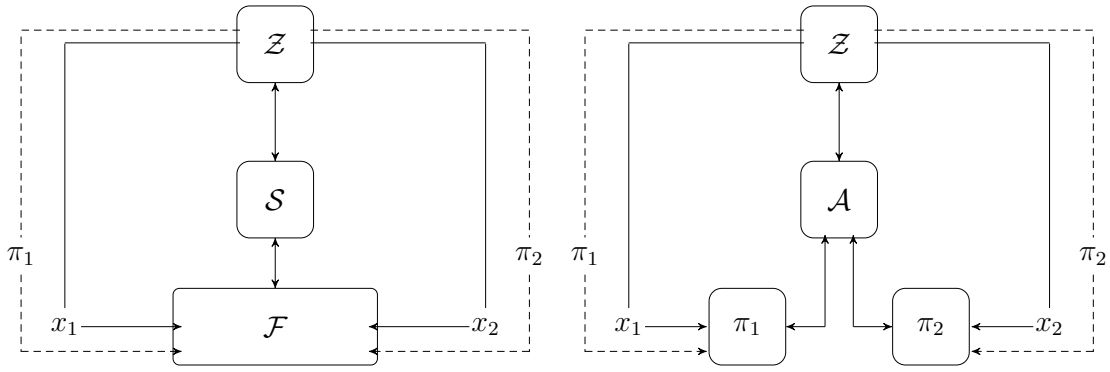

Fig. 1.: Ideal execution (left) and real execution (right) in the standard model

with the adversary according to its code. Whenever $\mathcal{F}$ outputs a value to a party, the party immediately copies this value to its own output tape. We call the parties in the ideal protocol *dummy parties*. Let ϕ denote the ideal protocol for functionality $\mathcal{F}$.

Securely realizing an ideal functionality. We say that a protocol π *emulates* protocol ϕ if for any network adversary $\mathcal{A}$ there exists an adversary (also known as simulator) $\mathcal{S}$ such that no environment $\mathcal{Z}$, on any input, can tell with non-negligible probability whether it is interacting with $\mathcal{A}$ and parties running π, or it is interacting with $\mathcal{S}$ and parties running ϕ. This means that, from the point of view of the environment, running protocol π is as good as interacting with ϕ. We say that π *securely realizes* an ideal functionality $\mathcal{F}$ if it emulates the corresponding ideal protocol ϕ. More precise definitions follow. A distribution ensemble is called *binary* if it consists of distributions over $\{0,1\}$.

Definition 8. *Let π and ϕ be protocols. We say that π* emulates *ϕ if for any adversary $\mathcal{A}$ there exists an adversary $\mathcal{S}$ such that for any standalone environment $\mathcal{Z}$, we have* $\mathsf{REAL}_{\mathcal{A},\Pi,\mathcal{Z}} \stackrel{c}{\equiv} \mathsf{IDEAL}_{\mathcal{S},\phi,\mathcal{Z}}$.

Definition 9. *Let $\mathcal{F}$ be an ideal functionality and let π be a protocol. We say that π* realizes *$\mathcal{F}$ if π* emulates *the ideal protocol ϕ.*

2.1.4.3 Ideal functionalities

We here describe some functionalities which can be useful for our protocols in the body. We also discuss some of their implementations.

Random Oracle Functionality $\mathcal{F}_{\mathsf{RO}}$. The random oracle model (e.g., [BR93]) captures an idealization of a hash function. We here present the random oracle functionality $\mathcal{F}_{\mathsf{RO}}$ that has been defined in [HM04].

FUNCTIONALITY $\mathcal{F}_{\mathsf{RO}}$

The functionality $\mathcal{F}_{\mathsf{RO}}$ is parameterized by a security parameter κ, and interacts with a set $\mathcal{P}$ of parties, and an adversary. The functionality keeps a list L (which is initially empty) of pairs of bitstrings.

1. Upon receiving a value (m) (with $m \in \{0,1\}^*$) from some party $\mathsf{P} \in \mathcal{P}$ or from the adversary, proceed as follows.
   - If there is a pair $(m, \tilde{h})$ for some $\tilde{h} \in \{0,1\}^\kappa$ in the list L,set $h := \tilde{h}$.
   - if there is no such pair, choose uniformly $h \in \{0,1\}^\kappa$ and store the pair (m, h) in L.

   Once h is set, reply to the requesting party with (h).

Fig. 2.: Random oracle functionality $\mathcal{F}_{\mathsf{RO}}$.

Multi-Session Signature Functionality $\hat{\mathcal{F}}_{\mathsf{uSIG}}$. We present the multi-session version of the digital signature functionality in [Can03]. While the digital signature functionality can be realized by a signature protocol based on ordinary signature scheme, this functionality here can be realized a signature protocol based on unique signature scheme. The underlying part highlights the difference be between ours and that in [Can03]. The definition of unique signature scheme can be found below.

FUNCTIONALITY $\hat{\mathcal{F}}_{\mathsf{uSIG}}$

The functionality $\hat{\mathcal{F}}_{\mathsf{uSIG}}$ interacts with a set of signers $\{S_1, \ldots, S_k\}$, and a set of verifiers $\{V_1, \ldots, V_n\}$, and an adversary $\mathcal{S}$.

Key Gerneration: Upon receiving input $(\text{KEYGEN}, \mathsf{sid}, \mathsf{ssid})$ from a signer $P \in \{S_1, \ldots, S_k\}$, verify that $\mathsf{ssid} = (P, \mathsf{ssid}')$ for some ssid'. If not, ignore the request. Otherwise, hand $(\text{KEYGEN}, \mathsf{sid}, \mathsf{ssid})$ to the adversary. Upon receiving $(\text{VERIFICATION-KEY}, \mathsf{sid}, \mathsf{ssid}, \mathsf{vk})$ from the adversary, output $(\text{VERIFICATION-KEY}, \mathsf{sid}, \mathsf{ssid}, \mathsf{vk})$ to the party P.

Signature Generation: Upon receiving input $(\text{SIGN}, \mathsf{sid}, \mathsf{ssid}, m)$ from a signer $P \in \{S_1, \ldots, S_k\}$, verify that $\mathsf{ssid} = (P, \mathsf{ssid}')$ for some ssid'. If not, ignore the request. Otherwise check if and no $(\mathsf{ssid}, m, \sigma, \cdot, \cdot)$ has been recorded. If not, output $(\text{SIGNATURE}, \mathsf{sid}, \mathsf{ssid}, m, \sigma)$. Otherwise, send $(\text{SIGN}, \mathsf{sid}, \mathsf{ssid}, m)$ to the adversary.
Upon receiving $(\text{SIGNATURE}, \mathsf{sid}, \mathsf{ssid}, m, \sigma)$ from the adversary, verify that no entry $(\mathsf{ssid}, m, \sigma, \mathsf{vk}, 0)$ is recorded. If it is, then output an error message to P and halt. Otherwise, output $(\text{SIGNATURE}, \mathsf{sid}, \mathsf{ssid}, m, \sigma)$ to P, and record the entry $(\mathsf{ssid}, m, \sigma, \mathsf{vk}, 1)$.

Signature Verification: Upon receiving a message $(\text{VERIFY}, \mathsf{sid}, \mathsf{ssid}, m, \sigma, \mathsf{vk}')$ from some party $P \in \{V_1, \ldots, V_n\}$, hand $(\text{VERIFY}, \mathsf{sid}, \mathsf{ssid}, m, \sigma, \mathsf{vk}')$ to the adversary. Upon receiving $(\text{VERIFIED}, \mathsf{sid}, \mathsf{ssid}, m, \phi)$ from the adversary, do:

1. If $\mathsf{vk}' = \mathsf{vk}$ and the entry $(\mathsf{ssid}, m, \sigma, \mathsf{vk}, 1)$ is recorded, then set $f := 1$.
2. Else, if $\mathsf{vk}' = \mathsf{vk}$, the signer of subsession ssid is not corrupted, and no entry $(\mathsf{ssid}, m, \sigma', \mathsf{vk}, 1)$ for any σ' is recorded, then set $f := 0$.
3. Else, if there is an entry $(\mathsf{ssid}, m, \sigma, \mathsf{vk}', f')$ recorded, then let $f := f'$.
4. Else, let $f := \phi$ and record the entry $(\mathsf{ssid}, m, \sigma, \mathsf{vk}', \phi)$.

Output $(\text{VERIFIED}, \mathsf{sid}, \mathsf{ssid}, m, f)$ to P.

Fig. 3.: Multi-session signature functionality $\hat{\mathcal{F}}_{\mathsf{uSIG}}$.

2.2 Blockchain technology

2.2.1 Blockchain Notations

As in the original Bitcoin white paper [Nak08], a proof-of-work (PoW) blockchain is created and maintained by a set of players called *PoW-miners*. A PoW blockchain $\mathcal{C}$ consists of a sequence of ℓ concatenated PoW-blocks $B_1\|B_2\|\cdots\|B_\ell$, where $\ell \geq 0$. For each blockchain, we specify several notations such as head, length, and subchain:

- *blockchain head*, denoted $\mathsf{head}(\mathcal{C})$, refers to the topmost block B_ℓ in chain $\mathcal{C}$;
- *blockchain length*, denoted $\mathsf{len}(\mathcal{C})$, is the number of blocks in blockchain $\mathcal{C}$, and here $\mathsf{len}(\mathcal{C}) = \ell$;
- *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 $\mathcal{C} \preceq \mathcal{C}'$.

Similar notations can be defined for a proof-of-stake (PoS) blockchain $\tilde{\mathcal{C}}$ which consists of a sequence of $\tilde{\ell}$ concatenated PoS-blocks $\tilde{B}_1\|\tilde{B}_2\|\cdots\|\tilde{B}_{\tilde{\ell}}$ for $\tilde{\ell} > 0$, and is maintained by a set of players called *PoS-holders*.

2.2.2 Nakamoto's blockchain

We here briefly review Nakamoto's Bitcoin blockchain [Nak08]. Bitcoin blockchain is based on proof-of-work puzzles [DN93; Bac02], which can be abstractly described via the following hash inequality:

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

where $h_{\mathsf{w}} \in \{0,1\}^\kappa$ is the hash of the previous proof-of-work block (κ is a security parameter), ω is a suitable solution for this puzzle, X is the record component of the block,

and $\mathtt{T}$ denotes the current proof-of-work target. See [GKL15] 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 $\omega \in \{0,1\}^{\kappa}$ and checks whether ω 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. [GKL15] and Pass et al. [PSS17a] 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.

CHAPTER 3

MODEL

3.1 Modeling blockchain protocol execution

In order to study the security of Bitcoin-like protocol, Garay et al. [GKL15] and then Pass et al. [PSS17a] set up the first cryptographic models by following Canetti's formulation of the "real world" executions [Can00a; Can00b]. In this section, we borrow many ideas from their formulations. We further extend their models so that more blockchain protocols, e.g., 2-hop blockchains, are allowed.

Network communication. The underlying communication for blockchain protocols are formulated via a functionality $\mathcal{F}_{\mathsf{NET}}$ which captures the atomic unauthenticated "send-to-all" broadcast in an asynchronous communication setting. The functionality is parameterized by an upper bound Δ on the network latency, and interacts with players under the direction of the adversary. More concretely, the functionality proceeds as follows. Whenever it receives a message from a player, it would contact the adversary to ask the adversary to specify the delivery time for the message. Note that, if the specified delivery time exceeds the delay upper bound Δ, the functionality would not follow the adversary's instruction, and only delay the message to a maximum number of Δ rounds. That said, no messages are delayed more than Δ rounds. In addition, the adversary could read all messages sent by all honest players before deciding his strategy; the adversary may "spoof" the source of a message they transmit and impersonate the (honest) sender of the message. The functionality $\mathcal{F}_{\mathsf{NET}}$ is formally described in Figure 4.

PoW-miners and PoS-holders. We specify two types of players PoW-miner and

FUNCTIONALITY $\mathcal{F}_{\mathsf{NET}}$

The functionality is parametrized by Δ, and interacts with a set $\mathcal{P}$ of PoS-players, and the adversary.

- Upon receiving $(\textsc{Broadcast}, m)$ from a party P where $\mathsf{P} \in \mathcal{P}$, send $(\textsc{Broadcast}, \mathsf{P}, m)$ to $\mathcal{S}$ and record (P, m, D) where $D = 1$.
- Upon receiving $(\textsc{Delay}, m, \mathsf{P}', \mathsf{P}'', t)$ from $\mathcal{S}$ where $\mathsf{P}', \mathsf{P}'' \in \mathcal{P}$ (here, the adversary can "spoof" the source of the message), then
  - If there is a record (P, m, b) such that $D = 1$ and if $D + t \leq \Delta$, then set $D := D + t$, record $(\mathsf{delay}, \mathsf{P}', \mathsf{P}'', m, D)$
  - Otherwise, ignore the message.
- Upon receiving $(\textsc{Send})$ from the adversary.
  - For all record of form $(\mathsf{delay}, \cdot, \cdot, \cdot, D)$, set $D := D - 1$,
  - Check for all record of form $(\mathsf{delay}, \cdot, \cdot, \cdot, D)$ such that $D = 0$, then send the messages to the specified receivers in the form.

Fig. 4.: Network functionality $\mathcal{F}_{\mathsf{NET}}$.

PoS-holder which correspond to two types of chains, specifically, PoW-chain and PoS-chain; and two types of rounds that execute in turn: PoW-round and PoS-round. These two types of chains are tied and grow together (at the same rate.) That said, a chain-pair including a PoW-chain and a PoS-chain should have two member chains of the roughly similar length. If the PoW-chain or PoS-chain in this pair grows too fast, this chain-pair becomes invalid. Note that the PoW-miners and PoS-holders are playing different roles in our model; however, without the collaboration of these two types of players, our model cannot be secure.

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 hon-

est PoW-miners/PoS-holders are simply clusters of some arbitrary number of honest idealized-model PoW-miners/PoS-holders. We note that the protocol's players may never be certain about the number of participants in the protocol execution, given the unauthenticated nature of the communication model. Moreover, for simplicity, only a standalone static model is considered in this model, and the number of players is fixed during the course of the protocol execution.

In each PoW-round, PoW-miners have ability proportionally to their computing power to produce proof-of-work blocks. More concretely, upon receiving messages which include many chain-pairs from the network, each PoW-miner would choose the best valid chain-pair, and then utilize his computing power to solve the PoW puzzle in order to extend the best chain-pair in this round. On the other hand, in each PoS-round, the PoS-holder with the derived identity from the new PoW-block of the previous PoW-round is able to generate a new PoS-block, and then appends the new block to the best chain-pair on his local view. Note that, for each PoS-holder, the probability of being chosen is based on the amount of stake that party has. The detail of our blockchain execution is presented below.

The execution of proof-of-work blockchain protocol. Following Canetti's formulation of the "real world" executions [Can00a; Can00b], we present an abstract model for hybrid proof-of-work/proof-of-stake blockchain protocol $\Pi = (\Pi^{\mathsf{w}}, \Pi^{\mathsf{s}})$ in the hybrid model where Π^{w} and Π^{s} denote the code run by PoW-miners and PoS-holders, respectively. 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 κ is a security parameter), which activates an n bounded number of PoW-miners and $\tilde{n}$ bounded number of PoS-holders. The execution proceeds in *rounds*. Without loss of generality, we assume that odd rounds correspond to PoW-miners, and even rounds correspond to PoS-holders. The environment $\mathcal{Z}$ can "manage" protocol players through an adversary $\mathcal{A}$ that can adaptively corrupt honest parties.

PREPARATION PHASE. Any active PoW-miners and PoS-holder, before participating in the mining process, will obtain additional information from the system based on its initial state. More concretely, PoW-miners and PoS-holder will obtain the entire blockchain information from the system, and store the blockchain information in its storage.

EXECUTION PHASE. Any active PoW-miners or PoS-holder can join the mining process after the preparation phase. The mining process consists of multiple rounds. In each round, the environment $\mathcal{Z}$ provides inputs for all miners and receives outputs from these miners, and the miners communicate with each other. More concretely, in each round, each honest miner 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 storage; then based on the stored information, it carries out some mining operations; in the case that a new block is generated, the miner sends out the new block via BROADCAST() which will be guaranteed to be delivered to all miners in the beginning of the next round. Note that, at any point of the execution, the environment $\mathcal{Z}$ can communicate with the adversary $\mathcal{A}$ or access the local information of the miners. In addition have ideal access to an unbounded number of copies of some ideal functionality $\mathcal{F}$.

For simplicity, 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 consider adaptive adversaries who are allowed to take control of protocol players on the fly. At any point of the execution, $\mathcal{Z}$ can send message $(\text{CORRUPT}, P)$ to adversary $\mathcal{A}$. From that point, $\mathcal{A}$ has access to the party's local state and controls P.

Let $\mathsf{EXEC}^{\mathcal{F}}_{\Pi,\mathcal{A},\mathcal{Z}}$ be a random variable denoting the joint view of all parties (i.e., all their inputs, random coins and messages received, including those from the random oracle and signatures) in the above $\mathcal{F}$-hybrid execution; note that this joint view fully determines the execution. Whenever $\mathcal{F}$ are clear from context we often write $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$ or $\mathsf{EXEC}_{(\Pi^{\mathrm{w}},\Pi^{\mathrm{s}}),\mathcal{A},\mathcal{Z}}$.

3.2 Modeling proof-of-work

3.2.1 Functionality $\mathcal{F}^*_{\mathsf{PoW}}$

In our setting, the PoW-miners have limited ability to produce proofs of work. To capture this, all PoW-miners are assumed to have access to a physical resource setup $\mathcal{F}^*_{\mathsf{PoW}}$ which manages a huge "farm of computing devices" (represented by "nonces"), and these devices are provided by the environment $\mathcal{Z}$ through the adversary. In order to utilize the computing power of the functionality, each player needs to register the computing services of $\mathcal{F}^*_{\mathsf{PoW}}$ or disconnect the services at some later points. Indeed, this captures the dynamic computing power setting where different players could consume the computing resource for different windows of time. Functionality $\mathcal{F}^*_{\mathsf{PoW}}$ abstracts out the Bitcoin like mining process; this will simplify the design and analysis of protocols based on such mining ecosystem. Here, each PoW-miner is able to request one *search* query from $\mathcal{F}^*_{\mathsf{PoW}}$ that consumes one unit of computing power granted in each execution round. Then the player can only find it with a certain probability p. Besides the computing services, the setup also provides the *verification services* which allow any player to verify solutions in many times. Please refer to Figure 5 for more details.

As discussed above, this is an "idealized" interpretation of the setting where all miners have the same amount of computing power; nevertheless, this idealized model does not sacrifice generality. The adversary $\mathcal{A}$ is allowed to perform at most t queries per round, where t is the number of corrupted PoW-miners. Thus, the computing power is consumed by querying the functionality in a bounded number of times.

We remark that we are not the first to formulate the setup of computing resources. Earlier efforts can be found in [KMS14; PSS17a]. We argue that, our $\mathcal{F}^*_{\mathsf{PoW}}$ formulation is more rigorous than the previous efforts; we explicitly model how the computing resource is managed and distributed from the environment to parties. In our model, each party can

FUNCTIONALITY $\mathcal{F}^*_{\mathsf{PoW}}$

The functionality is parameterized by a PoW parameter p, a PoW security parameter κ, and interacts with PoW-miners $\{\mathcal{W}_1, \ldots, \mathcal{W}_n\}$, PoS-holders $\{\mathcal{S}_{n+1}, \ldots, \mathcal{S}_{n+\tilde{n}}\}$, as well as an adversary $\mathcal{S}$.

1. **Computing Resource Registration.** Upon receiving a message $(\text{WORK-REGISTER}, \mathcal{W}_i)$ from party $\mathcal{W}_i \in \{\mathcal{W}_1, \ldots, \mathcal{W}_n\}$, pass the message to the adversary. Upon receiving a message $(\text{WORK-REGISTERED}, \mathcal{W}_i, \omega)$ from the adversary, record $(\mathcal{W}_i, \omega)$, and pass ω to the party $\mathcal{W}_i$ (the party $\mathcal{W}_i$ registered.)

2. **Work Query.** Upon receiving $(\text{SEARCH}, \omega, \mathit{context})$ from a PoW-miner $\mathcal{W}_i$ or from the adversary where $\omega \in \{0,1\}^\kappa$, proceed as follows.

   (a) If there is record $(\cdot, \langle \mathit{context}, \omega \rangle, b_i^{\mathsf{w}})$, then send $(\text{SEARCHED}, \mathcal{W}_i, b_i^{\mathsf{w}})$ to the player.

   (b) Otherwise, check If $(\mathcal{W}_i, \omega)$ is not recorded, then ignore the message. Otherwise, if $(\mathcal{W}_i, \omega)$ is recorded (the party $\mathcal{W}_i$ registered and granted one unit of computational resource), then delete the record $(\mathcal{W}_i, \omega)$, and process the following.

   - with probability p, set $b_i^{\mathsf{w}} := 1$, (the party $\mathcal{W}_i$ discovers the solution,)
   - with probability $1 - p$, set $b_i^{\mathsf{w}} := 0$. (the party $\mathcal{W}_i$ does not discover the solution.)

   Then record $(\mathcal{W}_i, \langle \mathit{context}, \omega \rangle, b_i^{\mathsf{w}})$. Then send $(\text{SEARCHED}, \mathcal{W}_i, b_i^{\mathsf{w}})$ to the player $\mathcal{W}_i$

3. **Work Verification Query.** Upon receiving $(\text{RO-VERIFY}, \mathit{context}, \omega)$ from a party P where $P \in \{\mathcal{W}_1, \ldots, \mathcal{W}_n, \mathcal{S}_{n+1}, \ldots, \mathcal{S}_{n+\tilde{n}}\}$ or from the adversary, check if there exists a recorded entry $(\cdot, \langle \mathit{context}, \omega \rangle, b_i^{\mathsf{w}})$ then send $(\text{RO-VERIFIED}, b_i^{\mathsf{w}})$ to the party P. Otherwise, output error.

Fig. 5.: Proof-of-work functionality $\mathcal{F}^*_{\mathsf{PoW}}$.

register and receive the computing resource represented by nonce ω under the control of the environment. That said, this model naturally captures the joining of new players or the rejoining of old players.

Furthermore, our $\mathcal{F}^*_{\mathsf{PoW}}$ is closely related to, but different from $\mathcal{F}_{\mathsf{Tree}}$ in [PSS17a] and $\mathcal{F}_{\text{StX}}$ [Bad+17] . In [PSS17a], a "per protocol" approach is taken. That is, for different blockchain protocol, say GHOST protocol [SZ15], a different variant of $\mathcal{F}_{\mathsf{Tree}}$ should be defined. We take a different approach; we abstract the essence of the underlying resources, and our proof-of-work functionality $\mathcal{F}^*_{\mathsf{PoW}}$ can be used for different PoW-based blockchain protocols, and we do not need to revise the setup *per protocol*. Similar to $\mathcal{F}_{\mathsf{Tree}}$ and ours, In [Bad+17], the functionality $\mathcal{F}_{\text{StX}}$ abstracts out the election process where messages are accepted with a certain probability. However, $\mathcal{F}_{\text{StX}}$ additionally checks the validity of the

"input messages" ("context" in our functionality) with respect to the historical state. On the other hand, our $\mathcal{F}^*_{\mathsf{PoW}}$ only captures the proof-of-work mining mechanism, the validity of the context is not covered. This demonstrates the purpose of our model. Specifically, we want to abstract the proof-of-work mechanism or how players find the solutions for puzzles only. This $\mathcal{F}^*_{\mathsf{PoW}}$ can be used to analyze any PoW-based blockchain protocols.

3.2.2 Implementing $\mathcal{F}^*_{\mathsf{PoW}}$ in $\mathcal{F}_{\mathsf{RO}}$-hybrid model

As discussed in section 3.2.1, the functionality $\mathcal{F}^*_{\mathsf{PoW}}$ is implemented by a random oracle functionality $\mathcal{F}_{\mathsf{RO}}$ [HM04]. We denote ϕ^*_{PoW} as the ideal protocol for an ideal functionality $\mathcal{F}^*_{\mathsf{PoW}}$ and π_{PoW} as the protocol in the $\mathcal{F}_{\mathsf{RO}}$-hybrid model. In the ideal protocol ϕ^*_{PoW}, players are dummy since they just forward the messages received from the environment $\mathcal{Z}$ to the functionality $\mathcal{F}^*_{\mathsf{PoW}}$, and then forward the messages received from the functionality to the environment. On the other hand, upon receiving messages from the environment, the players in π_{PoW} execute the protocol and then pass the outputs to the environment. Note that, we allow each PoW-miner to receive only *one unit of computing power* (one chance of querying the random oracle) per round.

Essentially, protocol π_{PoW} captures the core of PoW-based blockchain (e.g., Bitcoin). Informally, π_{PoW} carries out the following two main steps: first, each PoW-miner is able to"mine" for a puzzle solution of a hash inequality; after that, any other players (PoW-miners or PoS-holders) can verify the found solution. The formal description of protocol π_{PoW} is given in Figure 6.

Let $\mathcal{S}$ be the adversary against the ideal protocol ϕ^*_{PoW}, and $\mathcal{A}$ be the adversary against the hybrid protocol π_{PoW}. We now show that π_{PoW} is as " secure" as ϕ^*_{PoW} with respect to the adversary $\mathcal{S}$. Let $\mathsf{EXEC}^{\mathcal{F}_{\mathsf{RO}}}_{\pi_{\mathsf{PoW}},\mathcal{A},\mathcal{Z}}$ be the random variable denoting the joint view of all parties in the execution of π_{PoW} with the adversary $\mathcal{A}$ and an environment $\mathcal{Z}$. In addition, let $\mathsf{EXEC}^{\mathcal{F}^*_{\mathsf{PoW}}}_{\phi^*_{\mathsf{PoW}},\mathcal{S},\mathcal{Z}}$ be the random variable denoting the joint view of all parties in

PROTOCOL π_{PoW}

The protocol is parameterized by a PoW parameter p and a security parameter κ.

Each PoW-miner $\mathcal{W}_i$, where $1 \leq i \leq n$, proceeds as follows.

1. Upon receiving $(\text{WORK-REGISTER}, \mathcal{W}_i)$ from the environment $\mathcal{Z}$, if $(\mathcal{W}_i, \omega)$ is already recorded, then ignore the message. Otherwise, choose $\omega \in \{0,1\}^\kappa$, record $(\mathcal{W}_i, \omega)$, and pass the message $(\text{WORK-REGISTERED}, \mathcal{W}_i, \omega)$ to the environment.

2. Upon receiving $(\text{SEARCH}, \mathcal{W}_i, \mathit{context})$ from the environment $\mathcal{Z}$, if recored ω, then query the functionality $\mathcal{F}_{\mathsf{RO}}$ on input $B = (\mathit{context}, \omega)$ and then obtain output h. If $h \leq \mathrm{T}$ where $\mathrm{T} = p \cdot 2^\kappa$, set $b_i^{\mathsf{w}} := 1$. Otherwise, if $h > \mathrm{T}$, set $b_i^{\mathsf{w}} := 0$ send $(\text{SEARCHED}, \mathcal{W}_i, b_i^{\mathsf{w}})$ to the environment.

Each player $P \in \{\mathcal{W}_1, \ldots, \mathcal{W}_n, \mathcal{S}_{n+1}, \ldots, \mathcal{S}_{n+\tilde{n}}\}$ proceeds as follows. Upon receiving $(\text{RO-VERIFY}, \mathit{context}, \omega)$ from the environment $\mathcal{Z}$, send $(\mathit{context}, \omega)$ to the functionality $\mathcal{F}_{\mathsf{RO}}$ and receive h. If $h \leq \mathrm{T}$, set $b_i^{\mathsf{w}} =:= 1$. Otherwise, if $h > \mathrm{T}$, set $b_i^{\mathsf{w}} := 0$. Then send $(\text{RO-VERIFIED}, b_i^{\mathsf{w}})$ to the environment.

Fig. 6.: Proof-of-work protocol π_{PoW}.

the execution of ϕ^*_{PoW} with the adversary $\mathcal{S}$ and an environment $\mathcal{Z}$.

Lemma 1. *Consider protocol π_{PoW} in Figure 6 and the ideal protocol ϕ^*_{PoW} described above. Assume $pn \ll 1$, where p is a PoW parameter and n is the total number of PoW-miners. It holds that the two ensembles $\mathsf{EXEC}^{\mathcal{F}_{\mathsf{RO}}}_{\pi_{\mathsf{PoW}},\mathcal{A},\mathcal{Z}}$ and $\mathsf{EXEC}^{\mathcal{F}^*_{\mathsf{PoW}}}_{\phi^*_{\mathsf{PoW}},\mathcal{S},\mathcal{Z}}$ are perfectly indistinguishable.*

Proof. We show that the two executions are perfectly close by the following simulation. Consider the adversary $\mathcal{A}$ for π_{PoW}, we now construct an adversary $\mathcal{S}$ on input 1^κ and a PoW parameter p for ϕ^*_{PoW} as follows. Note that, the adversary $\mathcal{S}$ stores a table T.

1. Upon receiving $(\text{WORK-REGISTER}, \mathcal{W}_i)$ from the functionality $\mathcal{F}^*_{\mathsf{PoW}}$ for honest parties, choose random $\omega \leftarrow \{0,1\}^\kappa$, then send $(\text{WORK-REGISTERED}, \mathcal{W}_i, \omega)$ to $\mathcal{F}^*_{\mathsf{PoW}}$. For corrupted parties, wait to receive $(\mathit{context}, \omega)$ when the corrupted parties (controlled by $\mathcal{A}$) query $\mathcal{F}_{\mathsf{RO}}$ and set the nonce.

2. Upon receiving $(\mathit{context}, \omega)$ from any party $\mathcal{W}_i$ (corrupted or uncorrupted), if there is a record $((\mathit{context}, \omega), b_i^{\mathsf{w}}, h)$ in T, then send h to $\mathcal{W}_i$. Otherwise, if there is no

record $((context, \omega), b_i^{\mathsf{w}})$, send $(\textsc{Search}, \omega, context)$ to $\mathcal{F}^*_{\mathsf{PoW}}$ and

Upon receiving $(\textsc{Searched}, \mathcal{W}_i, b_i^{\mathsf{w}})$ from $\mathcal{F}^*_{\mathsf{PoW}}$. if $b_i^{\mathsf{w}} = 1$ choose random $h \in \{0,1\}^\kappa$ such that $h \leq \mathrm{T}$. Otherwise if $b_i^{\mathsf{w}} = 0$ choose random $h \in \{0,1\}^\kappa$ such that $h > \mathrm{T}$ where $\mathrm{T} = p \cdot 2^\kappa$. Then record $((context, \omega), b_i^{\mathsf{w}}, h)$. and then send h to $\mathcal{W}_i$

Since there are no interactions with the adversary $\mathcal{A}$ or corrupted parties in registration phase and verification query phase, the simulator does not need to simulate these phases.

We demonstrate that $\mathsf{EXEC}^{\mathcal{F}_{\mathrm{RO}}}_{\pi_{\mathsf{PoW}},\mathcal{A},\mathcal{Z}}$ and $\mathsf{EXEC}^{\mathcal{F}^*_{\mathsf{PoW}}}_{\phi^*_{\mathsf{PoW}},\mathcal{S},\mathcal{Z}}$ are perfectly indistinguishable. This is done by showing that the joint view of all parties in the execution of π_{PoW} with adversary $\mathcal{A}$ and environment $\mathcal{Z}$ is perfectly indistinguishable from the joint view of all parties in the execution of ϕ^*_{PoW} with $\mathcal{S}$ and $\mathcal{Z}$. We can easily see that (1) each random oracle query from $\mathcal{A}$ is sampled uniformly at random from a set $\{0,1\}^\kappa$, and (2) each work query, or verification query to $\mathcal{F}^*_{\mathsf{PoW}}$ is also sampled uniformly at random from the set $\{0,1\}^\kappa$. Therefore, the two ensembles $\mathsf{EXEC}^{\mathcal{F}_{\mathrm{RO}}}_{\pi_{\mathsf{PoW}},\mathcal{A},\mathcal{Z}}$ and $\mathsf{EXEC}^{\mathcal{F}^*_{\mathsf{PoW}}}_{\phi^*_{\mathsf{PoW}},\mathcal{S},\mathcal{Z}}$ are perfectly close.

□

3.3 Modeling proof-of-stake

3.3.1 Unpredictable unique signature functionality $\mathcal{F}_{\mathsf{uuSIG}}$

The usual unique signature functionality is too weak for proof-of-stake related proposals since the adversary can see the distribution of signatures and predict future election. Thus, we need the unpredictability of the signature scheme. This is formalized via the unpredictable unique signature scheme in Figure 7

FUNCTIONALITY $\mathcal{F}_{\mathsf{uuSIG}}$

The functionality $\mathcal{F}_{\mathsf{uuSIG}}$ interacts with a set of signers $\{S_1, \ldots, S_k\}$, and a set of verifiers $\{V_1, \ldots, V_n\}$, and an adversary $\mathcal{S}$.

1. *Key Gerneration:* Upon receiving input $(\text{KEYGEN}, \mathsf{sid}, \mathsf{ssid})$ from a signer $P \in \{S_1, \ldots, S_k\}$, verify that $\mathsf{ssid} = (P, \mathsf{ssid}')$ for some ssid'. If not, ignore the request. Otherwise, hand $(\text{KEYGEN}, \mathsf{sid}, \mathsf{ssid})$ to the adversary. Upon receiving $(\text{VERIFICATION-KEY}, \mathsf{sid}, \mathsf{ssid}, \mathsf{vk})$ from the adversary, output $(\text{VERIFICATION-KEY}, \mathsf{sid}, \mathsf{ssid}, \mathsf{vk})$ to the party P.

2. *Signature Generation:* Upon receiving input $(\text{SIGN}, \mathsf{sid}, \mathsf{ssid}, m)$ from a signer $P \in \{S_1, \ldots, S_k\}$, verify that $\mathsf{ssid} = (P, \mathsf{ssid}')$ for some ssid'. If not, ignore the request. Otherwise check if and no $(\mathsf{ssid}, m, \sigma, \cdot, \cdot, \cdot)$ has been recorded. If not, output $(\text{SIGNATURE}, \mathsf{sid}, \mathsf{ssid}, m, \sigma, h)$. Otherwise, send $(\text{SIGN}, \mathsf{sid}, \mathsf{ssid}, m)$ to the adversary.

   Upon receiving $(\text{SIGNATURE}, \mathsf{sid}, \mathsf{ssid}, m, \sigma)$ from the adversary, verify that no entry $(\mathsf{ssid}, m, \sigma, \cdot, \mathsf{vk}, 0)$ is recorded. If it is, then output an error message to P and halt. Otherwise, choose random $h \in \{0,1\}^\kappa$ and output $(\text{SIGNATURE}, \mathsf{sid}, \mathsf{ssid}, m, \sigma, h)$ to P, and record the entry $(\mathsf{ssid}, m, \sigma, h, \mathsf{vk}, 1)$.

3. *Signature Verification:* Upon receiving a message $(\text{VERIFY}, \mathsf{sid}, \mathsf{ssid}, m, \sigma, h, \mathsf{vk}')$ from some party $P \in \{V_1, \ldots, V_n\}$, hand $(\text{VERIFY}, \mathsf{sid}, \mathsf{ssid}, m, \sigma, \mathsf{vk}')$ to the adversary. Upon receiving $(\text{VERIFIED}, \mathsf{sid}, \mathsf{ssid}, m, \phi)$ from the adversary, do:

   (a) If $\mathsf{vk}' = \mathsf{vk}$ and the entry $(\mathsf{ssid}, m, \sigma, h, \mathsf{vk}, 1)$ is recorded, then set $f := 1$.

   (b) Else, if $\mathsf{vk}' = \mathsf{vk}$, the signer of subsession ssid is not corrupted, and no entry $(\mathsf{ssid}, m, \sigma', \mathsf{vk}, 1)$ for any σ' is recorded, then set $f := 0$.

   (c) Else, if there is an entry $(\mathsf{ssid}, m, \sigma, \mathsf{vk}', f')$ recorded, then let $f := f'$.

   (d) Else, let $f := \phi$ and record the entry $(\mathsf{ssid}, m, \sigma, \mathsf{vk}', \phi)$.

   Output $(\text{VERIFIED}, \mathsf{sid}, \mathsf{ssid}, m, f)$ to P.

Fig. 7.: Unpredictable unique signature functionality $\mathcal{F}_{\mathsf{uuSIG}}$.

3.3.2 Functionality $\mathcal{F}^*_{\mathsf{PoS}}$

In this subsection, we introduce our proof-of-stake functionality $\mathcal{F}^*_{\mathsf{PoS}}$ describing the usage of virtual resource in our system. Please refer to Figure 8 for more details.

Similar to the proof-of-work functionality $\mathcal{F}^*_{\mathsf{PoW}}$, the proof-of-stake functionality are introduced to capture the leader election process via the virtual resource, i.e., stakes. Intuitively, players having stake will have a certain chance (with probability $\tilde{p}$) of being elected as the leader. This guarantees that the more stakes players have, the better chances of being elected.

3.3.3 Implementing $\mathcal{F}^*_{\mathsf{PoS}}$ in $\{\mathcal{F}_{\mathsf{uuSIG}}, \mathcal{F}_{\mathsf{RO}}\}$-hybrid model

We denote ϕ^*_{PoS} as the ideal protocol for an ideal functionality $\mathcal{F}^*_{\mathsf{PoS}}$, and π_{PoS} as protocol in the $\{\mathcal{F}_{\mathsf{uuSIG}}, \mathcal{F}_{\mathsf{RO}}\}$-hybrid model. In the ideal protocol ϕ^*_{PoS}, the dummy players only forward the messages received from the environment to the functionality $\mathcal{F}^*_{\mathsf{PoS}}$, and then forward the messages received from the functionality to the environment. Informally, each PoS-holder through his stake determines whether he is the elected leader in the current round or not; then he is able to generate a valid signature, which can later be verified by any other players. The protocol π_{PoS} is formally described in Figure 9.

Let $\mathcal{S}$ be the adversary against the ideal protocol ϕ^*_{PoS}, and $\mathcal{A}$ be the adversary against protocol π_{PoS}. Let $\mathsf{EXEC}^{\mathcal{F}^*_{\mathsf{PoS}}}_{\phi^*_{\mathsf{PoS}},\mathcal{S},\mathcal{Z}}$ be the random variable denoting the joint view of all parties in the execution of ϕ^*_{PoS} with the adversary $\mathcal{S}$ and an environment $\mathcal{Z}$. Let $\mathsf{EXEC}^{\mathcal{F}_{\mathsf{uuSIG}},\mathcal{F}_{\mathsf{RO}}}_{\pi_{\mathsf{PoS}},\mathcal{A},\mathcal{Z}}$ be the random variable denoting the joint view of all parties in the execution of π_{PoS} with the adversary $\mathcal{A}$ and an environment $\mathcal{Z}$.

Lemma 2. *Consider ϕ^*_{PoS} described above and π_{PoS} in Figure 9. Assume $\tilde{p}\tilde{n} \ll 1$, where $\tilde{p}$ is a PoS parameter and $\tilde{n}$ is the total number of PoS-holders. It holds that the two ensembles $\mathsf{EXEC}^{\mathcal{F}^*_{\mathsf{PoS}}}_{\phi^*_{\mathsf{PoS}},\mathcal{S},\mathcal{Z}}$ and $\mathsf{EXEC}^{\mathcal{F}_{\mathsf{uuSIG}},\mathcal{F}_{\mathsf{RO}}}_{\pi_{\mathsf{PoS}},\mathcal{A},\mathcal{Z}}$ are perfectly indistinguishable.*

FUNCTIONALITY $\mathcal{F}^*_{\mathsf{PoS}}$

The functionality is parameterized by a PoS parameter $\tilde{p}$, a security parameter κ, and interacts with PoW-miners $\{\mathcal{W}_1, \ldots, \mathcal{W}_n\}$, PoS-holders $\{\mathcal{S}_{n+1}, \ldots, \mathcal{S}_{n+\tilde{n}}\}$, as well as an adversary $\mathcal{S}$.

1. **Stake Resource Registration.**

   Upon receiving a message $(\textsc{Stake-Register}, \mathcal{S}_j)$ from party $\mathcal{S}_j \in \{\mathcal{S}_{n+1}, \ldots, \mathcal{S}_{n+\tilde{n}}\}$, if there is an entry $(\mathcal{S}_j, 1)$, then ignore the message. Otherwise, pass the message to the adversary. Upon receiving a message $(\textsc{Stake-Registered}, \mathcal{S}_j, \mathsf{vk})$ from the adversary, record $(\mathcal{S}_j, \mathsf{vk})$, and pass vk to the party $\mathcal{S}_j$ (the party $\mathcal{S}_j$ registered.)

2. **Stake Election:** Upon receiving $(\textsc{Elect}, \mathsf{vk}, \mathit{context})$ from a PoS-holder $\mathcal{S}_j$, proceed as follows.

   If $(\mathcal{S}_j, \mathsf{vk})$ is not recorded, then ignore the message. Otherwise, proceed the following.

   - If $(\textsc{Elect}, \mathit{context}, \mathcal{S}_j, \tilde{\omega}, \mathsf{vk}, b^{\mathsf{s}}_j)$ has been recored, send $(\textsc{Elected}, \mathcal{S}_j, \tilde{\omega}, b^{\mathsf{s}}_j)$ to $\mathcal{S}_j$
   - Otherwise,
     - if $(\textsc{Sign}, \mathcal{S}_j, \mathit{context}, \tilde{\omega}, \cdot)$ has been recorded, then ignore the input.
     - Otherwise, send $(\textsc{Elect-Sign}, \mathcal{S}_j, \mathit{context})$ to the adversary. Upon receiving $(\textsc{Elect-Signed}, \mathcal{S}_j, \mathit{context}, \tilde{\omega})$ from the adversary, verify that no entry $(\textsc{Sign}, \mathcal{S}_j, \mathit{context}, \tilde{\omega}, 0)$ is recorded. If it is, then output an error message $(\textsc{Error})$ to $\mathcal{S}_j$ and halt. Else, record the entry $(\textsc{Sign}, \mathcal{S}_j, \mathit{context}, \tilde{\omega}, 1)$.

     Then

     - with probability $\tilde{p}$, set $b^{\mathsf{s}}_j := 1$ and send $(\textsc{Elected}, \mathcal{S}_j, \tilde{\omega}, b^{\mathsf{s}}_j)$ to $\mathcal{S}_j$. (the party $\mathcal{S}_j$ is elected.)
     - with probability $1 - \tilde{p}$, set $b^{\mathsf{s}}_j := 0$ and send $(\textsc{Elected}, \mathcal{S}_j, \tilde{\omega}, b^{\mathsf{s}}_j)$ to $\mathcal{S}_j$ (the party $\mathcal{S}_j$ is not elected.)

     and record the entry $(\textsc{Elect}, \mathit{context}, \mathcal{S}_j, \tilde{\omega}, \mathsf{vk}, b^{\mathsf{s}}_j)$.

3. **Stake Verification:** Upon receiving $(\textsc{Stake-Verify}, (\mathcal{S}_j, \mathit{context}), \tilde{\omega}, \mathsf{vk})$ from a party $P \in \{\mathcal{W}_1, \ldots, \mathcal{W}_n, \mathcal{S}_{n+1}, \ldots, \mathcal{S}_{n+\tilde{n}}\}$ or from the adversary,

   (a) If there exists a record of the form $(\textsc{Elect}, \mathit{context}, \mathcal{S}_j, \tilde{\omega}, \mathsf{vk}, b^{\mathsf{s}}_j)$ where $b^{\mathsf{s}}_j = 1$ (the party $\mathcal{S}_j$ is elected), hand $(\textsc{Stake-Verify}, (\mathcal{S}_j, \mathit{context}), \tilde{\omega}, \mathsf{vk})$ to the adversary. Upon receiving $(\textsc{Stake-Verified}, (\mathcal{S}_j, \mathit{context}, \tilde{\omega}), \phi)$ from the adversary, do:

   - If $(\textsc{Sign}, \mathcal{S}_j, \mathit{context}, \tilde{\omega}, \mathsf{vk}, 1)$ is recorded, then set $f := 1$.
   - Else, if $\mathcal{S}_j$ is not corrupted, and no entry $(\textsc{Sign}, \mathcal{S}_j, \mathit{context}, \tilde{\omega}', \mathsf{vk}, 1)$ for any $\tilde{\omega}'$ is recorded, then set $f := 0$ and record the entry $(\textsc{Sign}, \mathcal{S}_j, \mathit{context}, \tilde{\omega}, \mathsf{vk}, f)$
   - Else, if there is an entry $(\textsc{Sign}, \mathcal{S}_j, \mathit{context}, \tilde{\omega}, \mathsf{vk}, f')$, then set $f := f'$.
   - Else, set $f := \phi$, and record the entry $(\textsc{Sign}, \mathcal{S}_j, \mathit{context}, \tilde{\omega}, \mathsf{vk}, f)$.

   Output $(\textsc{Stake-Verified}, (\mathcal{S}_j, \mathit{context}), f)$ to the party P.

   (b) Otherwise, if there is no record of the form $(\textsc{Elect}, \mathit{context}, \mathcal{S}_j, \tilde{\omega}, \mathsf{vk})$ (the party $\mathcal{S}_j$ is not elected), set $f := 0$, and output $(\textsc{Stake-Verified}, (\mathcal{S}_j, \mathit{context}), f)$ to the party P.

Fig. 8.: Proof-of-stake functionality $\mathcal{F}^*_{\mathsf{PoS}}$.

PROTOCOL π_{PoS}

The protocol is parameterized by a PoS parameter $\tilde{p}$ and a security parameter κ.

Each $\mathcal{S} \in \mathcal{P}_0$, proceeds as follows.

1. Upon receiving $(\text{STAKE-REGISTER}, \mathcal{S})$ from the environment, pass $(\text{KEYGEN}, \mathsf{sid}, \mathsf{ssid})$ for some $\mathsf{sid}, \mathsf{ssid}$ where $\mathsf{ssid} = (\mathcal{S}, \mathsf{ssid}')$ for some ssid' to the functionality $\mathcal{F}_{\mathsf{uuSIG}}$. Upon receiving $(\text{VERIFICATION-KEY}, \mathsf{sid}, \mathsf{ssid}, \mathsf{vk})$ from $\hat{\mathcal{F}}_{\mathsf{uSIG}}$, record $(\mathcal{S}, \mathsf{vk})$. Then send vk to the environment.

2. Upon receiving $(\text{ELECT}, \mathcal{S}, \mathit{context})$ from the environment $\mathcal{Z}$, send $(\text{SIGN}, \mathsf{sid}, \mathsf{ssid}, \mathcal{S}, \mathit{context})$ to the functionality $\mathcal{F}_{\mathsf{uuSIG}}$.

   Upon receiving $(\text{SIGNATURE}, \mathsf{sid}, \mathsf{ssid}, (\mathcal{S}, \mathit{context}), \tilde{\omega}, h)$ from $\mathcal{F}_{\mathsf{uuSIG}}$, send $(\mathit{context}, \mathsf{vk}, \tilde{\omega}, h)$ to the functionality $\mathcal{F}_{\text{RO}}$ and receives $\tilde{h}^{\text{puzz}}$.

   - If $\tilde{h}^{\text{puzz}} > \mathtt{T}$ where $\mathtt{T} = \tilde{p} \cdot 2^{\kappa}$, then set $b^{\mathsf{s}} := 0$.
   - Else, set $b^{\mathsf{s}} := 1$.

   Send $(\text{ELECTED}, \mathcal{S}, \tilde{\omega}, b^{\mathsf{s}})$ to the environment.

3. Upon receiving $(\text{STAKE-VERIFY}, \mathcal{S}, \mathit{context}, \tilde{\omega}, h, \mathsf{vk})$ from the environment $\mathcal{Z}$, then send $(\text{VERIFY}, \mathsf{sid}, \mathsf{ssid}, \mathcal{S}, \mathit{context}, \tilde{\omega}, h, \mathsf{vk})$ to the functionality $\mathcal{F}_{\mathsf{uuSIG}}$. Upon receiving $(\text{VERIFIED}, \mathsf{sid}, \mathsf{ssid}, (\mathcal{S}, \mathit{context}), f)$ from the functionality $\mathcal{F}_{\mathsf{uuSIG}}$. If $f = 1$, send $(\mathit{context}, \mathsf{vk}, \tilde{\omega}, h)$ to the functionality $\mathcal{F}_{\text{RO}}$ and receives $\tilde{h}^{\text{puzz}}$.

   - If $\tilde{h}^{\text{puzz}} > \mathtt{T}$ where $\mathtt{T} = \tilde{p} \cdot 2^{\kappa}$, then set $f := 0$.
   - Else, set $f := 1$.

   send $(\text{STAKE-VERIFIED}, (\mathcal{S}, \mathit{context}), f)$ to the environment.

Fig. 9.: Proof-of-stake protocol π_{PoS}.

Proof. We show that the two executions are perfectly indistinguishable by the following simulation. Consider the adversary $\mathcal{A}$ for π_{PoS}, we now construct an adversary $\mathcal{S}$ on input 1^{κ} and a PoS parameter $\tilde{p}$ for ϕ^{*}_{PoS} as follows. $\mathcal{S}$ stores a table T

Initialization and Stake Election:

1. Simulating the execution with an uncorrupted party $\mathcal{S}$ as follows. When $\mathcal{S}$ receives in the ideal process a message $(\text{STAKE-VERIFIED}, \mathsf{P}, \mathit{context}, \tilde{\omega})$ from $\mathcal{F}^{*}_{\mathsf{PoS}}$, where $\mathcal{S}$ is uncorrupted, it proceeds as follows:

   If this is the first time that $\mathcal{S}$ generates a signature, then simulate for $\mathcal{A}$ the process

of key generation . That is, send to $\mathcal{A}$ (in the name of $\mathcal{F}_{\mathsf{uuSIG}}$) $(\textsc{Keygen}, \mathsf{sid}, \mathsf{ssid})$ to the adversary, and receive $(\textsc{Verification-Key}, \mathsf{sid}, \mathsf{ssid}, \mathsf{vk})$ from the adversary $\mathcal{A}$, then send $(\textsc{Register}, \mathsf{sid}, \mathsf{ssid}, \mathsf{vk})$ to the adversary $\mathcal{A}$; upon receiving $(\textsc{Registered}, \mathsf{sid}, \mathsf{ssid})$ from the adversary, then record the pair $(\mathsf{ssid}, \mathsf{vk})$.

2. Simulating the execution with a corrupted party $\mathcal{S}$ with $\mathcal{F}_{\mathsf{uuSIG}}$ as follows.

   (a) Upon receiving $(\textsc{Keygen}, \mathsf{sid}, \mathsf{ssid})$ from party $\mathcal{S}$, send to $\mathcal{A}$ (in the name of $\mathcal{F}_{\mathsf{uuSIG}}$) $(\textsc{Keygen}, \mathsf{sid}, \mathsf{ssid})$ to the adversary, and receive $(\textsc{Verification-Key}, \mathsf{sid}, \mathsf{ssid}, \mathsf{vk})$ from the adversary $\mathcal{A}$, then send $(\textsc{Verification-Key}, \mathsf{sid}, \mathsf{ssid}, \mathsf{vk})$ (in the name of $\mathcal{F}_{\mathsf{uuSIG}}$) to $\mathcal{S}$ and record $(\mathsf{ssid}, \mathcal{S}, \mathsf{vk})$.

   (b) Upon receiving $(\textsc{Sign}, \mathsf{sid}, \mathsf{ssid}, \mathcal{S}, \mathit{context})$ from party $\mathcal{S}$, check if no $(\mathsf{ssid}, \mathit{context}, \cdot, \cdot)$ has been recorded. If not, ignore the request. Otherwise, send $(\textsc{Sign}, \mathsf{sid}, \mathsf{ssid}, \mathit{context})$ to the adversary.

   Upon receiving $(\textsc{Signature}, \mathsf{sid}, \mathsf{ssid}, \mathit{context}, \tilde{\omega})$ from the adversary, choose random $h \in \{0,1\}^{\kappa}$, output $(\textsc{Signature}, \mathsf{sid}, \mathsf{ssid}, \mathit{context}, \tilde{\omega}, h)$ to $\mathcal{S}$, and record the entry $(\mathsf{ssid}, \mathit{context}, \tilde{\omega}, h, \mathsf{vk}, 1)$.

   (c) Then, instruct the corrupted party $\mathcal{S}$ send the message $(\textsc{Elect}, \mathcal{S}, \mathit{context}, \mathsf{vk})$ to $\mathcal{F}^{*}_{\mathsf{PoS}}$.

   Upon receiving $(\textsc{Sign}, \mathsf{sid}, \mathsf{ssid}, \mathit{context})$ from $\mathcal{F}^{*}_{\mathsf{PoS}}$, send $(\textsc{Signature}, \mathsf{sid}, \mathsf{ssid}, \mathit{context}, (\tilde{\omega}, h))$ to $\mathcal{F}^{*}_{\mathsf{PoS}}$.

3. Simulate the interaction of any party $\mathcal{S}$ with $\mathcal{F}_{\mathsf{RO}}$ as follows.

   For query $(\mathit{context}, \mathsf{vk}, \tilde{\omega}, h)$ from party $\mathcal{S}$, send $(\textsc{Stake-Verify}, \mathcal{S}, \mathit{context}, \tilde{\omega})$ to $\mathcal{F}^{*}_{\mathsf{PoS}}$ and receive $(\textsc{Stake-Verified}, (\mathcal{S}, \mathit{context}), \tilde{\omega}, f)$. If $f = 0$, choose random $\tilde{h}^{\text{puzz}} \in \{0,1\}^{\kappa}$ such that $\tilde{h}^{\text{puzz}} > \mathsf{T}$ where $\mathsf{T} = \tilde{p} \cdot 2^{\kappa}$. If $f = 1$, choose $\tilde{h}^{\text{puzz}} \in \{0,1\}^{\kappa}$

such that $\tilde{h}^{\text{puzz}} \leq \mathtt{T}$. Then store $((context, \mathsf{vk}, \tilde{\omega}, h), \tilde{h}^{\text{puzz}}))$ in the table T and send $\tilde{h}^{\text{puzz}}$ to $\mathcal{S}$.

Block Verification:

1. Simulating the execution with an uncorrupted party $\mathcal{S}$ as follows. When notified by $\mathcal{F}^*_{\mathsf{PoS}}$ that some uncorrupted party $\mathcal{S}$ made a verification request, proceed as follows. Upon receiving message $(\textsc{Stake-Verify}, \mathcal{S}, context, \tilde{\omega}, \mathsf{vk})$ from $\mathcal{F}^*_{\mathsf{PoS}}$, then forward message $(\textsc{Stake-Verify}, \mathcal{S}, context, \tilde{\omega}, h, \mathsf{vk})$ (h is recored with $context$ and $\tilde{\omega}$ by $\mathcal{S}$ above) to $\mathcal{A}$ (in the name of $\mathcal{F}_{\mathsf{uuSIG}}$). Forward $\mathcal{A}$'s response back to $\mathcal{F}^*_{\mathsf{PoS}}$.
2. Simulating the execution with a corrupted party $\mathcal{S}$ with $\hat{\mathcal{F}}_{\mathsf{uSIG}}$ as follows.
   (a) Instruct the corrupted party $\mathcal{S}$ send the message $(\textsc{Stake-Verify}, \mathcal{S}, context, \tilde{\omega}, \mathsf{vk})$ to $\mathcal{F}^*_{\mathsf{PoS}}$.
   (b) Upon receiving $(\textsc{Verify}, \mathsf{sid}, \mathsf{ssid}, \mathcal{S}, context, \tilde{\omega}, h, \mathsf{vk})$ from a corrupted party $\mathcal{S}$, generate a response following the instructions of $\mathcal{F}_{\mathsf{uuSIG}}$.
3. Simulate the interaction of any party $\mathcal{S}$ with $\mathcal{F}_{\mathsf{RO}}$ as follows.
   For query $(context, \mathsf{vk}, \tilde{\omega}, h)$ from party $\mathcal{S}$ check if $((context, \mathsf{vk}, \tilde{\omega}, h), \tilde{h}^{\text{puzz}}))$ in the table T and send $\tilde{h}^{\text{puzz}}$ to $\mathcal{S}$. otherwise, send $(context, \mathsf{vk}, \tilde{\omega})$, send $(\textsc{Stake-Verify}, \mathcal{S}, context, \tilde{\omega})$ to $\mathcal{F}^*_{\mathsf{PoS}}$ and receive $(\textsc{Stake-Verified}(\mathcal{S}, context), \tilde{\omega}, f)$. If $f = 0$, choose random $\tilde{h}^{\text{puzz}} \in \{0,1\}^\kappa$ such that $\tilde{h}^{\text{puzz}} > \mathtt{T}$ where $\mathtt{T} = \tilde{p} \cdot 2^\kappa$. If $f = 1$, choose $\tilde{h}^{\text{puzz}} \in \{0,1\}^\kappa$ such that $\tilde{h}^{\text{puzz}} \leq \mathtt{T}$. Then store $((h^{\text{prev}}, r, \mathsf{vk}, \tilde{\omega}), \tilde{h}^{\text{puzz}}))$ in the table T and send $\tilde{h}^{\text{puzz}}$ to $\mathcal{S}$.

We now show that the two ensembles $\mathsf{EXEC}^{\mathcal{F}^*_{\mathsf{PoS}}}_{\phi^*_{\mathsf{PoS}}, \mathcal{S}, \mathcal{Z}}$ and $\mathsf{EXEC}^{\hat{\mathcal{F}}_{\mathsf{uSIG}}, \mathcal{F}_{\mathsf{RO}}}_{\pi_{\mathsf{PoS}}, \mathcal{A}, \mathcal{Z}}$ are perfectly close. Notice that for each election query, the adversary $\mathcal{S}$ is noticed by the functionality $\mathcal{F}^*_{\mathsf{PoS}}$ whether this query is successful or not, then it samples the output randomly from a set $\{0,1\}^\kappa$ that satisfied inequality $\mathsf{H}(context, \mathsf{vk}, \tilde{\omega}, h) \leq \tilde{\mathtt{T}}$ if the query is successful.

Putting them together, the views of players in the two executions are perfectly indistinguishable.

□

3.3.4 Blockchain security properties

Previously, several fundamental security properties for 1-hop blockchain protocols have been defined: *common prefix property* [GKL15; PSS17a], *chain quality property* [GKL15], and *chain growth property* [KP15]. Intuitively, the chain growth property states that the chains of honest players should grow linearly to the number of rounds. The common prefix property indicates the consistency of any two honest chains except the last κ blocks. The chain quality property, aims at expressing the number of honest blocks' contributions that are contained in a sufficiently long and continuous part of an honest chain. Specifically, for parameters $\ell \in \mathbb{N}$ and $\mu \in (0, 1)$, the ratio of honest input contributions in a continuous part of an honest chain has a lower bounded μ.

We follow the same spirit to define the security properties for our 2-hop blockchain protocol. Since each valid PoS-chain and PoW-chain exist in our system as a pair having the same structure and grow at the same rate, we focus on the common prefix, chain quality, and chain growth properties for the PoS-chain. The definitions for these properties are formally given as follows.

Definition 10 (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 PoS-holder* $\mathcal{S} \in \{\mathcal{S}_{n+1}, \ldots, \mathcal{S}_{n+\tilde{n}}\}$ *with the local PoS-chain* $\tilde{\mathcal{C}}$ *in round* r *and* $\tilde{\mathcal{C}}'$ *in round* r' *where* $r' - r > 0$*, in* $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$*. It holds that* $\mathsf{len}(\tilde{\mathcal{C}}') - \mathsf{len}(\tilde{\mathcal{C}}) \geq g \cdot (r' - r)$

Definition 11 (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 PoS-holders* $\mathcal{S}_i$ *in round*

r and $\mathcal{S}_j$ in round r' with the local PoS-chains $\tilde{\mathcal{C}}_i$, $\tilde{\mathcal{C}}_j$, respectively, in $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$ *where $i, j \in \{n+1, \ldots, n+\tilde{n}\}, r \leq r'$, it holds that $\tilde{\mathcal{C}}_i[1, \ell_i] \preceq \tilde{\mathcal{C}}_j$ where $\ell_i = \mathsf{len}(\tilde{\mathcal{C}}_i) - \Theta(\kappa)$.*

Definition 12 (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 PoS-holder $\mathcal{S} \in \{\mathcal{S}_{n+1}, \ldots, \mathcal{S}_{n+\tilde{n}}\}$ with PoS-chain $\tilde{\mathcal{C}}$ in* $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$*, it holds that for large enough ℓ consecutive PoS-blocks of $\tilde{\mathcal{C}}$ the ratio of honest blocks is at least μ.*

We note that, the chain growth and common prefix for the PoW-chains would be implied from the PoS-chain except the chain quality property since the adversary could be able to attach more malicious PoW-blocks to the PoW-chain in case he controls the majority of computing power.

New property: Chain soundness. We here introduce a new security property, *chain soundness*, which is critical for blockchain protocols in the open setting. A good protocol in the open network environment, should ensure honest new players to join the system securely. Intuitively, the protocol can help the new players to obtain a blockchain which is compatible with the local chain of an existing honest player in some recent rounds. While this property is not needed for protocols in the *closed* setting where new players are not allowed, it is important for blockchains in the open network environments. Without this security requirement, unsatisfactory protocols could be allowed. The chain soundness property can be described as follows.

Definition 13 (Chain soundness)**.** *Consider 2-hop blockchain protocol Π with a set $\mathcal{P}$ of players. Consider a new player $\mathsf{P}_i \in \mathcal{P}$ with best local chain $\tilde{\mathcal{C}}_i$ in round r, in* $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$*. The* chain soundness *property $\mathcal{Q}_{\text{cs}}$ states the following: for the new player P_i and any existing players P_j with best local chain $\tilde{\mathcal{C}}_j$ at round r we have that $\tilde{\mathcal{C}}_i[\neg\kappa] \preceq \tilde{\mathcal{C}}_j$ and $\tilde{\mathcal{C}}_j[\neg\kappa] \preceq \tilde{\mathcal{C}}_i$.*

CHAPTER 4

INITIAL DESIGN AND PROVABLE SECURITY: 2-HOP BLOCKCHAIN

In this chapter, we propose a hybrid proof-of-work/proof-of-stake protocol. In our design, as argued above, we (intend to) use both physical resources and virtual resources. That means, in addition to PoW-miners, a new type of players — PoS-holder (stakeholder) — is introduced in our system. Now a winning PoW-miner cannot extend the blockchain immediately. Instead, the winning PoW-miner provides a base which enables a PoS-holder to be "selected" to extend the blockchain. In short, in our system, a PoW-miner and then a PoS-holder jointly extend the blockchain with a new block. If Nakamoto's consensus can be viewed as a *1-hop protocol*, then ours is a *2-hop protocol*.

4.1 2-hop design

We first give the high-level description of our 2-hop blockchain design in Section 4.1.1; then in Section 4.1.2, we formally present our main protocol; the process of choosing the best valid chain-pair among a set of chain-pairs can be found in Section 4.1.3.

4.1.1 High-level description

Nakamoto's system is powered by physical computing resources and secure against fully adaptive adversary who can corrupt participants at any moment. The blockchain is maintained by PoW-miners; there, each winning PoW-miner can extend the blockchain with a new block. In our design, as argued above, we (intend to) use both physical resources and virtual resources to achieve the same level of security— adaptive security. That means, in addition to PoW-miners, a new type of players — PoS-holder (stakeholder)

— is introduced in our system. Now a winning PoW-miner cannot extend the blockchain immediately. Instead, the winning PoW-miner provides a base which enables a PoS-holder to be "selected" to extend the blockchain. In short, in our system, a PoW-miner and then a PoS-holder jointly extend the blockchain with a new block. If Nakamoto's consensus can be viewed as a *1-hop protocol*, then ours is a *2-hop protocol*.

A pictorial illustration of our 2-hop blockchain structure can be found in Figure 10: red blocks are generated by PoW-miners in the first hops, while green blocks are produced by PoS-holders in the second hops; now naturally a PoW-chain consists the sequence of red blocks $B_1, B_2, B_3, \cdots$, and a PoS-chain consists the sequence of green blocks $\tilde{B}_1, \tilde{B}_2, \tilde{B}_3, \cdots$. In fact, our 2-hop blockchain is bootstrapped from an "already mature" blockchain denote as $B_{-N} \ldots, B_{-1}, B_0$ for an integer N; see the dark blocks in the figure.

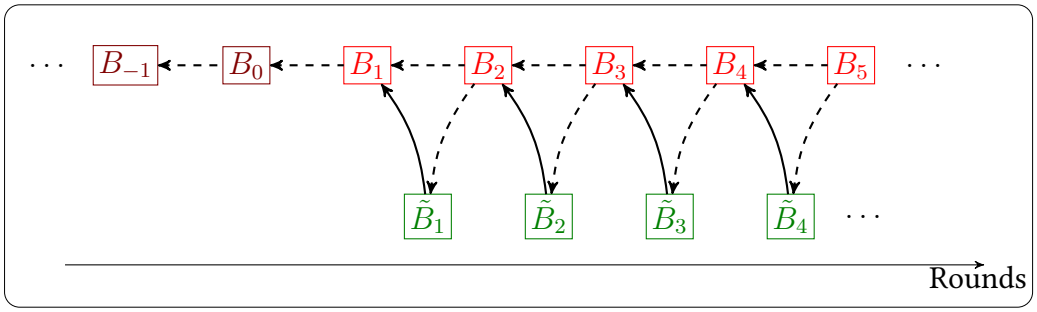

Fig. 10.: 2-hop blockchain structure

Here, dot arrows denote the first hops, and solid arrows denote the second hops. Red blocks B_i's denote the proof-of-work blocks, and green blocks $\tilde{B}_i$'s denote the corresponding proof-of-stake blocks. Note that the dark-red blocks are from the "mature blockchain".

In our scheme, there are 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 $\tilde{\mathcal{C}}$, 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 con-

sidered as a biased random beacon for choosing stakeholder (to extend the proof-of-stake chain).

- *Extending the PoW-chain:* To generate a new PoW-block, each miner first computes the hash $h_{\mathsf{w}} \in \{0,1\}^{\kappa}$ of the previous PoW-block (the head of the PoW-chain), the hash $h_{\mathsf{s}} \in \{0,1\}^{\kappa}$ of the head of the PoS-chain. To be noted that, the PoW chain and PoS chain must have same length in a valid chain pair. The miner then attempts to solve the following hash inequality

  $$\mathsf{H}(h_{\mathsf{w}}, h_{\mathsf{s}}, \omega) < \mathtt{T}$$

  by finding a suitable solution ω where $\mathtt{T}$ denotes the current proof-of-work target. PoW-block B is in the form $B = \langle h_{\mathsf{w}}, h_{\mathsf{s}}, \omega \rangle$.

  If the miner succeeds in finding the proper ω, he then generates a new PoW-block which includes the hash values h_{w}, h_{s} and the solution ω, and shares this PoW-block to other players in the network.

- *Extending the PoS-chain:* In our design, each PoS player holds two pairs of keys $(\mathsf{vk}, \mathsf{sk})$ and $(\mathsf{vk}', \mathsf{sk}')$ for digital signature schemes $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ and $(\mathsf{Gen}', \mathsf{Sign}', \mathsf{Verify}')$, respectively. Here $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ is a *unique* digital signature scheme, while $(\mathsf{Gen}', \mathsf{Sign}', \mathsf{Verify}')$ can be an ordinary digital signature scheme.

  We note that, in our 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

  $$\tilde{\mathsf{H}}(h_{\mathsf{w}}, \tilde{\omega}, \mathsf{vk}) < \tilde{\mathtt{T}}$$

  is allowed to generate a new PoS block, where $\tilde{\mathtt{T}}$ is the current proof-of-stake target,

h_{w} is the hash value of B and $\tilde{\omega} \leftarrow \mathsf{Sign}_{\mathsf{sk}}(B)$.

PoS-block $\tilde{B}$ is in the form $\tilde{B} = \langle (B, \tilde{\omega}, \mathsf{vk}), X, \sigma, \mathsf{vk}' \rangle$, where$X \in \{0,1\}^*$ is the payload of the proof-of-stake block $\tilde{B}$ (also denoted as $\mathsf{payload}(\tilde{B})$); and σ is produced by the PoS player but by using a different signing key sk', i.e., $\sigma \leftarrow \mathsf{Sign}'_{\mathsf{sk}'}((B, \tilde{\omega}, \mathsf{vk}), X)$.

We emphasize again that $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ is a unique signature scheme, which is critical for our design.

4.1.2 The main protocol

2-hop blockchain. It is important to note that in the Nakamoto PoW-based blockchain, the assumption to secure the system is that malicious miners control less than the half of computing power since if so they can fork a valid blockchain which breaks the consensus of the blockchain protocol. In our system, in order to secure against such attack, we need to combine two different resources: physical resource (i.e., computing power) and virtual resource (i.e., stake). Sequentially, we have two types of blockchains (PoW-chain and PoS-chain) corresponding to two types of rounds — PoW-round and PoS-round executing in turn — making 2-hop blockchain. Note that, in reality, one player could play both roles, PoW-miner and PoS-holder; however, without loss of generality, we treat the two roles separately. In order to tie them hard, the scheme maps each PoW-block to no more than 1 stakeholder. Only the stakeholder who has the privilege is able to generate the corresponding PoS-block of each PoW-block. Note that PoW-chains and PoS-chains existing in our system are represented as pairs, and each player locally stores a chain-pair. Therefore, the two member chains of each valid chain-pairs should have the same structure.

We now present our main protocol that describes the behaviour of PoW-miners and PoS-holders. The PoW and PoS executions vary slightly. On one hand, in the PoW execution, PoW-miners search for proof-of-work solutions via $\mathcal{F}^*_{\mathsf{PoW}}$. On the other hand, in the

PoS execution, PoS-holders follow the growth of the PoW-chain and use that to extend the PoS-chain via $\mathcal{F}^*_{\mathsf{PoS}}$. In general, PoW-miners and PoS-holders collect blockchain information from the network functionality $\mathcal{F}_{\mathsf{NET}}$, perform some validation and generating blocks, and then share their states with the network through $\mathcal{F}_{\mathsf{NET}}$.

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 [GKL15; PSS17a]. Initialization of each party's execution sets the round clock to zero and sets the local chain-pair $\langle \mathcal{C}_i, \tilde{\mathcal{C}}_i \rangle$, for $1 \leq i \leq n + \tilde{n}$ (n is the number of PoW-miners and $\tilde{n}$ is the number of PoS-holders), such that $\mathcal{C}_i := \mathcal{C}_{\mathsf{init}}$, $\tilde{\mathcal{C}}_i := \tilde{\mathcal{C}}_{\mathsf{init}}$, where $\mathcal{C}_{\mathsf{init}}$ is our initial blockchain, i.e., $\mathcal{C}_{\mathsf{init}} = B_{-N} \| \ldots \| B_{-1} \| B_0$, $\tilde{\mathcal{C}}_{\mathsf{init}} = 0 \| \ldots \| 0 \| 0$, and $\mathsf{len}(\mathcal{C}_{\mathsf{init}}) = \mathsf{len}(\tilde{\mathcal{C}}_{\mathsf{init}})$.

We assume at the beginning of the protocol execution, a set of PoS-holders are already registered. The registration information are included in $\mathcal{C}_{\mathsf{init}}$. We emphasize that new players may be registered later during the execution. The new registration information will be included in the payload of the following PoS blocks.

Please refer to Figure 11 for more details of our main protocol.

4.1.3 The best chain-pair strategy

In this subsection, we describe the rules in which a single valid chain-pair is selected for consensus. Roughly speaking, a chain-pair is the best valid pair if it has the longest valid PoW-chain. We introduce process BestValid, which is run locally by PoW-miners or PoS-holders, to select the best chain-pair. The BestValid process is parameterized by a content validation predicate $V(\cdot)$ and an initial chain $\mathcal{C}_{\mathsf{init}}$ where $V(\cdot)$ determines the proper structure of the information that is stored into the blockchain as in [GKL15], and takes as input 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.

We emphasize that since each valid PoS-block is tied to a PoW-block, and each

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

The protocol is parameterized by a content validation predicate $V(\cdot)$. Initially, set $\mathcal{C}_i := \mathcal{C}_{\mathsf{init}}$, $\tilde{\mathcal{C}}_i := \tilde{\mathcal{C}}_{\mathsf{init}}$, where $\mathcal{C}_{\mathsf{init}} = B_{-N}\| \ldots \|B_{-1}\|B_0$ and $\tilde{\mathcal{C}}_{\mathsf{init}} = 0\| \ldots \|0\|0$, $\mathsf{len}(\mathcal{C}_{\mathsf{init}}) = \mathsf{len}(\tilde{\mathcal{C}}_{\mathsf{init}})$, and then set $state_i := \{\langle \mathcal{C}_i, \tilde{\mathcal{C}}_i \rangle\}$ for $1 \leq i \leq n + \tilde{n}$.

1. **PoW-Miner** Π^{w}. Each PoW-miner $\mathcal{W}_i$, for $1 \leq i \leq n$, with local state $state_i$ proceeds as follows. Without lost of generality, we assume that the PoW-miners already registered to the functionality $\mathcal{F}^*_{\mathsf{PoW}}$. For each odd round, upon receiving message (INPUT-WORK, $\mathcal{W}_i$) from the environment $\mathcal{Z}$, proceed as follows.

   (a) *Select the best local chain-pair:* Upon receiving many messages of the form (MESSAGE, P, $\langle \mathcal{C}, \tilde{\mathcal{C}} \rangle$) from $\mathcal{F}_{\mathsf{NET}}$ for any party $P \in \{\mathcal{W}_1, \ldots, \mathcal{W}_n, \mathcal{S}_{n+1}, \ldots, \mathcal{S}_{n+\tilde{n}}\}$, let $\mathbb{C}$ be the set of all chain-pair collected from $\mathcal{F}_{\mathsf{NET}}$, then compute $\langle \mathcal{C}_{\mathrm{best}}, \tilde{\mathcal{C}}_{\mathrm{best}} \rangle := \mathsf{BestValid}(\mathbb{C} \cup \langle \mathcal{C}_i, \tilde{\mathcal{C}}_i \rangle, \textit{PoW-miner})$, and then set $\mathcal{C}_i := \mathcal{C}_{\mathrm{best}}$ and $\tilde{\mathcal{C}}_i := \tilde{\mathcal{C}}_{\mathrm{best}}$.

   (b) *Attempt to extend PoW-chain:*

   - *Create the pointer to the previous block:* Send $(\mathsf{head}(\mathcal{C}_i), \mathsf{head}(\tilde{\mathcal{C}}_i))$ to the ideal functionality $\mathcal{F}_{\mathsf{RO}}$ and receive h from $\mathcal{F}_{\mathsf{RO}}$.
   - *Search for a puzzle solution:* If $h \neq \perp$, then send (WORK-REGISTER, $\mathcal{W}_i$) to the ideal functionality $\mathcal{F}^*_{\mathsf{PoW}}$ and receive ω. Then send (SEARCH, ω, h) to the ideal functionality $\mathcal{F}^*_{\mathsf{PoW}}$, and then receive (SEARCHED, $\mathcal{W}_i, b_i^{\mathsf{w}}$) from $\mathcal{F}^*_{\mathsf{PoW}}$.
   - *Generate a new PoW-block:* If $b_i^{\mathsf{w}} = 1$, set $B := \langle h, \omega \rangle$ (which means $\mathcal{W}_i$ is the player who found the puzzle solution in this round), set $\mathcal{C}_i := \mathcal{C}_i \| B$, and $state_i := state_i \cup \{\langle \mathcal{C}_i, \tilde{\mathcal{C}}_i \rangle\}$. Then send (BROADCAST, $\langle \mathcal{C}_i, \tilde{\mathcal{C}}_i \rangle$) to $\mathcal{F}_{\mathsf{NET}}$.

   Return (RETURN-WORK, $\mathcal{W}_i$) to the environment $\mathcal{Z}$.

2. **PoS-Holder** Π^{s}. Each PoS-holder $\mathcal{S}_j$, for $n + 1 \leq j \leq n + \tilde{n}$, with local state $state_j$, proceeds as follows. Note that, initially, each PoS-holder sends (STAKE-REGISTER, $\mathcal{S}_j$) to the ideal functionality $\mathcal{F}^*_{\mathsf{PoS}}$ and receive vk. Also, each PoS-holder registers to the ideal functionality $\mathcal{F}_{\mathsf{SIG}}$ and receive vk' Upon receiving message (INPUT-STAKE, $\mathcal{S}_j, X$) from the environment $\mathcal{Z}$ where X denotes the block-payload, proceed as follows.

   (a) *Select the best local chain-pair:* Upon receiving many messages of the form (MESSAGE, P, $\langle \mathcal{C}, \tilde{\mathcal{C}} \rangle$) from $\mathcal{F}_{\mathsf{NET}}$ for any party $P \in \{\mathcal{W}_1, \ldots, \mathcal{W}_n, \mathcal{S}_{n+1}, \ldots, \mathcal{S}_{n+\tilde{n}}\}$, let $\mathbb{C}$ be the set of all chain-pair collected from $\mathcal{F}_{\mathsf{NET}}$, then compute $\langle \mathcal{C}_{\mathrm{best}}, \tilde{\mathcal{C}}_{\mathrm{best}} \rangle := \mathsf{BestValid}(\mathbb{C} \cup \langle \mathcal{C}_j, \tilde{\mathcal{C}}_j \rangle, \textit{PoS-holder})$, and then set $\mathcal{C}_j := \mathcal{C}_{\mathrm{best}}$ and $\tilde{\mathcal{C}}_j := \tilde{\mathcal{C}}_{\mathrm{best}}$.

   (b) *Attempt to extend PoS-chain:* Consider there is a new PoW-block B in $\mathcal{C}_{\mathrm{best}}$.

   - *Stake election:* Send (ELECT, vk, B) to the ideal functionality $\mathcal{F}^*_{\mathsf{PoS}}$, and receive (ELECTED, $\mathcal{S}_j, \tilde{\omega}, b_j^{\mathsf{s}}$) from $\mathcal{F}^*_{\mathsf{PoS}}$.
   - *Generate a signature:* If $b_j^{\mathsf{s}} = 1$, send (SIGN, $\langle B, X, \mathcal{S}_j, \tilde{\omega} \rangle$) to the ideal functionality $\mathcal{F}_{\mathsf{SIG}}$, and receive (SIGNED, $\mathcal{S}_j, \langle B, X, \mathcal{S}_j, \tilde{\omega} \rangle, \sigma$) from $\mathcal{F}_{\mathsf{SIG}}$.
   - *Generate a new PoS-block:* Set $\tilde{B} := \langle B, \tilde{\omega}, \mathsf{vk}, X, \sigma, \mathsf{vk}' \rangle$. Next, set $\tilde{\mathcal{C}}_j = \tilde{\mathcal{C}}_j \| \tilde{B}$, and $state_j := state_j \cup \{\langle \mathcal{C}_j, \tilde{\mathcal{C}}_j \rangle\}$. Then send (BROADCAST, $\langle \mathcal{C}_j, \tilde{\mathcal{C}}_j \rangle$) to $\mathcal{F}_{\mathsf{NET}}$.

   Return (RETURN-Stake, $\mathcal{S}_j$) to the environment $\mathcal{Z}$.

Fig. 11.: Our main protocol $\Pi = (\Pi^{\mathsf{w}}, \Pi^{\mathsf{s}})$ in the $\{\mathcal{F}^*_{\mathsf{PoW}}, \mathcal{F}_{\mathsf{RO}}, \mathcal{F}^*_{\mathsf{PoS}}, \mathcal{F}_{\mathsf{SIG}}, \mathcal{F}_{\mathsf{NET}}\}$-hybrid model with respect to the local process BestValid (See Figure 12).

PoW-block or PoS-block is valid if their peers are valid. Thus, a chain-pair is valid if all block-pairs in this chain-pair are valid. A valid chain-pair with respect to PoW-miners should have two member chains (PoW-chain and PoS-chain) of the same length. On the other hand, a valid chain-pair with respect to PoS-holders may have the PoW-chain longer than the PoS-chain by one block since the PoW-chain might be extended by one new block in the previous PoW-round. That said, if the player who executes this process is a PoS-holder and if there exists a new PoW-blocks, this block would be validated separately since its corresponding PoS-block has not been generated yet. Thus, for every chain-pair, the process first checks if the length of the PoW-chain in the pair is longer than the PoS-chain by one block and validates this new PoW-block first, and then evaluates every block-pair of this chain-pair. As said, PoS-blocks are generated from PoW-blocks; thus, PoS-blocks without corresponding PoW-blocks are not valid. After the validation, the best valid chain-pair is the one with the longest PoW-chain. Please refer to Figure 12 for more details.

Remark 2 (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. [ES14]), we follow the simple strategy considered in [GKL15]; 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.*

4.2 Security analysis

Our 2-hop blockchain can be viewed as a natural generalization of Nakamoto's famous 1-hop blockchain [Nak08]. The security analysis of the 2-hop blockchain here is inspired and influenced by previous analysis of Nakamoto's 1-hop protocol [GKL15;

PROCESS BestValid

The process BestValid is parameterized by a content validation predicate $V(\cdot)$ and an initial chain $\mathcal{C}_{\mathsf{init}}$. The input is $(\mathbb{C}', \mathit{Type})$.
For every chain-pair $\langle \mathcal{C}, \tilde{\mathcal{C}} \rangle \in \mathbb{C}'$, and proceed as follows.

1. Check if $\mathsf{len}(\mathcal{C}) - 1 = \mathsf{len}(\tilde{\mathcal{C}})$ and $\mathit{Type} = \mathit{PoS\text{-}holder}$, then set $\ell := \mathsf{len}(\mathcal{C})$. Then verify the new PoW-block $\mathcal{C}[\ell]$ as follows.

   - *Verify the pointer in $\mathcal{C}[\ell]$:* Parse $\mathcal{C}[\ell]$ to obtain $\langle h_\ell, \omega_\ell \rangle$, send $(\mathcal{C}[\ell-1], \tilde{\mathcal{C}}[\ell-1])$ to $\mathcal{F}_{\mathsf{RO}}$ and then receive (h). If $h \neq h_\ell$, remove this chain-pair from $\mathbb{C}'$.
   - *Verify the PoW solution in $\mathcal{C}[\ell]$:* Parse $\mathcal{C}[\ell]$ to obtain $\mathcal{C}[\ell] := \langle h, \omega \rangle$. Send message $(\text{RO-VERIFY}, h, \omega)$ to the functionality $\mathcal{F}^*_{\mathsf{PoW}}$, and then receive $(\text{RO-VERIFIED}, f)$. If $f = 0$, remove this chain-pair from $\mathbb{C}'$.

2. Check if $\mathsf{len}(\mathcal{C}) = \mathsf{len}(\tilde{\mathcal{C}})$ and $V(\mathsf{payload}(\tilde{\mathcal{C}})) = 1$, or $\mathsf{len}(\mathcal{C}) - 1 = \mathsf{len}(\tilde{\mathcal{C}})$ and $\mathit{Type} = \mathit{PoS\text{-}holder}$. If yes, for i from $\mathsf{len}(\tilde{\mathcal{C}})$ to $\mathsf{len}(\mathcal{C}_{\mathsf{init}})$, proceed as follows.

   - *Verify PoW-block $\mathcal{C}[i]$:*
     - *Verify the pointer in $\mathcal{C}[i]$:* Parse $\mathcal{C}[i]$ to obtain $\langle h_i, \omega_i \rangle$, send $(\mathcal{C}[i-1], \tilde{\mathcal{C}}[i-1])$ to $\mathcal{F}_{\mathsf{RO}}$ and then receive (h).
     - *Verify the PoW solution in $\mathcal{C}[i]$:* Send message $(\text{RO-VERIFY}, \mathcal{C}[i])$ to the functionality $\mathcal{F}^*_{\mathsf{PoW}}$, and then receive $(\text{RO-VERIFIED}, h')$.

     If $h_i \neq h$ or $f = 0$, set $\mathbb{b}_1 := 0$. Else, set $\mathbb{b}_1 := 1$.
   - *Verify PoS-block $\tilde{\mathcal{C}}[i]$:* Parse $\tilde{\mathcal{C}}[i]$ to obtain $\langle B, \tilde{\omega}, \mathbb{S}, \mathsf{vk}, X, \sigma, \mathsf{vk}' \rangle$.
     - send $(\text{VERIFY}, ((B, \tilde{\omega}, \mathbb{S}), X), \sigma, \mathsf{vk}'_i)$ to $\mathcal{F}_{\mathsf{CERT}}$ ans receive $(\text{VERIFIED}, ((B, \tilde{\omega}, \mathbb{S}), X,)f)$. if $f = 1$set $\mathbb{b}_2 := 1$; $\mathbb{b}_2 := 0$ otherwise.
     - Then send message $(\text{STAKE-VERIFY}, (B, X, \mathbb{S}_j), \tilde{\omega}, \mathsf{vk})$ to the functionality $\mathcal{F}^*_{\mathsf{PoS}}$. Upon receiving message $(\text{STAKE-VERIFIED}, (B, X, \mathbb{S}_j), f)$ from $\mathcal{F}^*_{\mathsf{PoS}}$, if $f = 0$ or $B \neq \mathcal{C}[i]$, set $\mathbb{b}_3 := 0$. Else, set $\mathbb{b}_3 := 1$.
   - If $\mathbb{b}_1 = 0$ or $\mathbb{b}_2 = 0$, or $\mathbb{b}_3 = 0$, remove this chain-pair from $\mathbb{C}'$.

3. Otherwise, remove this chain-pair from $\mathbb{C}'$.

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

Fig. 12.: The chain set validation process BestValid.

Intuitively, BestValid ensures that, if $\mathit{Type} = \mathit{PoS\text{-}miner}$, 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 longer than the PoS-chain by one block since there may a new PoW-block produced in the previous rounds.

PSS17a]. In order to improve the readability, in general, we will keep consistency of notations. We use the original letters to denote the PoW parameters for the first hop, e.g., α; we use letters with tilde to denote the PoS parameters for the second hop, e.g., $\tilde{\alpha}$; finally, we use letters with hat to denote the collective parameters for both hops, e.g., $\hat{\alpha}$.

We use the flat mode to simplify of the security analysis. Consider the total number of PoW-miners is n, the portion of malicious computing power is ρ, and a PoW parameter p.

- Let $\alpha = 1-(1-p)^{(1-\rho)n}$ be the probability that at least one honest PoW-miner mines a block successfully in a round .
- Let $\beta = \rho np$ be the expected number of PoW-blocks that malicious PoW-miners can find in a round.

Here, when $pn \ll 1$, we have $\alpha \approx (1-\rho)np$, and thus $\frac{\alpha}{\beta} \approx \frac{1-\rho}{\rho}$. We assume $0 < \alpha \ll 1$, $0 < \beta \ll 1$ and $\alpha = \lambda\beta$ where $\lambda \in (0, \infty)$. We note that, in Nakamoto protocol, λ must be greater than 1; but in our setting λ could be less than 1, i.e., the malicious parties could control more computing resource.

We then describe the important parameters in the second hop (i.e., proof-of-stake blockchain). Similar to that in the first hop, we consider the total number of PoS-holders is $\tilde{n}$, the portion of malicious stakes is $\tilde{\rho}$. Let $\tilde{p}$ be the probability that a PoW-block is mapped to a PoS-holder. We assume $\tilde{p}\tilde{n} \ll 1$, so we have:

- 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.

It is obvious that 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 $\hat{\beta} = \beta\tilde{\beta}$ it 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.

We consider the network delay model as in [PSS17a]. We will show in section 4.2.4, $\hat{\gamma} = \frac{\hat{\alpha}}{1+2\Delta\hat{\alpha}}$ 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. We also assume $(\alpha + \beta)\Delta \ll 1$.

We are now ready to state our main theorems.

Theorem 3 (Chain Growth for PoS-chain). *Consider protocol $\Pi = (\Pi^{\mathsf{w}}, \Pi^{\mathsf{s}})$ in Section 4.1.2. For any honest PoS-holder $\mathcal{S} \in \{\mathcal{S}_{n+1}, \ldots, \mathcal{S}_{n+\tilde{n}}\}$ with the local PoS-chain $\tilde{\mathcal{C}}$ in round r and $\tilde{\mathcal{C}}'$ in round r' where $t = r' - r > 0$, in $\mathsf{EXEC}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}$, the probability that $\mathsf{len}(\tilde{\mathcal{C}}') - \mathsf{len}(\tilde{\mathcal{C}}) \geq g \cdot t$ is at least $1 - e^{-\Omega(t)}$ where $g = (1 - \delta)\hat{\gamma}$.*

Theorem 4 (Chain Quality for PoS-chain). *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 4.1.2. For any honest PoS-holder $\mathcal{S} \in \{\mathcal{S}_{n+1}, \ldots, \mathcal{S}_{n+\tilde{n}}\}$ with PoS-chain $\tilde{\mathcal{C}}$ in $\mathsf{EXEC}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}$, the probability that, for large enough ℓ consecutive PoS-blocks of $\tilde{\mathcal{C}}$ which are generated in s rounds, the ratio of honest blocks is no less than $\mu = 1 - (1 + \delta)\frac{(\alpha+\beta)\tilde{\beta}}{\hat{\gamma}}$ is at least $1 - e^{-\Omega(\ell)}$.*

Theorem 5 (Common Prefix for PoS-chain). *We assume $\hat{\alpha} = \hat{\lambda}(\alpha+\beta)\tilde{\beta}$ and $\hat{\lambda} > 1$. For any $\delta > 0$, consider protocol $\Pi = (\Pi^{\mathsf{w}}, \Pi^{\mathsf{s}})$ in Section 4.1.2. Let κ be the security parameter. For any two honest PoS-holders $\mathcal{S}_i$ in round r and $\mathcal{S}_j$ in round r', with the local best PoS-chains $\tilde{\mathcal{C}}_i$, $\tilde{\mathcal{C}}_j$, respectively, in $\mathsf{EXEC}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}$ where $r \leq r'$ and $i, j \in \{n + 1, \ldots, n + \tilde{n}\}$, the probability that $\tilde{\mathcal{C}}_i[1, \ell_i] \preceq \tilde{\mathcal{C}}_j$ where $\ell_i = \mathsf{len}(\tilde{\mathcal{C}}_i) - \Theta(\kappa)$ is at least $1 - e^{-\Omega(\kappa)}$.*

Theorem 6 (Chain Soundness for PoS-chain). *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 4.1.2. Let κ be the security parameter. For any new spawned honest player PoS-holder $\mathcal{S}_i$ and any existing honest player $\mathcal{S}_j$ in round r,*

with the local best PoS-chains $\tilde{\mathcal{C}}_i$, $\tilde{\mathcal{C}}_j$, respectively, in $\mathsf{EXEC}_{(\Pi^{\mathrm{w}},\Pi^{\mathrm{s}}),\mathcal{A},\mathcal{Z}}$*, the probability that* $\tilde{\mathcal{C}}_i[1,\ell] \preceq \tilde{\mathcal{C}}_j$ *and* $\tilde{\mathcal{C}}_j[1,\ell] \preceq \tilde{\mathcal{C}}_i$ *where* $\ell = \mathsf{len}(\tilde{\mathcal{C}}_i) - \Theta(\kappa)$ *is at least* $1 - e^{-\Omega(\kappa)}$.

We notice that the assumptions of $\hat{\gamma}$ are different in chain quality and common prefix properties. This is because if the malicious players want to destroy chain quality property, he could recycle the computing power from honest miners.

4.2.1 Analysis ideas

In this section, we informally introduce the analysis idea. In our scheme, there are two types of players, PoW-miners and PoS-holders (stakeholders). 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 (via $\mathcal{F}^*_{\mathsf{PoS}}$) . 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 4 types of compositions.

- **Case 1:** An honest PoW-miner finds a new PoW-block which is mapped to an honest PoS-holder. The honest PoS-holder will sign the corresponding PoS-block.
- **Case 2:** A malicious PoW-miner finds a new PoW-block which is mapped to a malicious PoS-holder. The malicious PoS-holder will sign the corresponding PoS-block.
- **Case 3:** An honest PoW-miner finds a new PoW-block which is mapped to a malicious PoS-holder. The malicious PoS-holders may sign it or just discard it.
- **Case 4:** When a malicious PoW-miner finds a new PoW-block which is mapped to an honest PoS-holder. He may ask the honest PoS-holder to sign or just discard it.

We first explain the proof intuition. The malicious players cannot prevent **Case 1**; therefore, they cannot prevent the growth of blockchain and **Case 1** will guarantee chain growth property. Furthermore, the malicious parties can generate PoS-blocks from **Case**

2 and **Case 3**. In some consecutive rounds, the total number of PoS-blocks from malicious players are bounded by these two cases. If the probability of **Case 2** and **Case 3** is smaller than **Case 1**, the malicious players cannot generate more PoS-blocks than the honest players. Since the malicious players cannot prevent **Case 1**, what they can do is to compete the blocks from the honest players with their own. Even if they win all of the competition there are still some blocks from honest players remaining. This will guarantee the chain quality property. Finally, we assume the probability that all of the PoW-miners find a new PoW-block in a round is very small. This means there is no new block being broadcast in most of the rounds. If all of the honest players do not receive the new block for some rounds, they would take the same best chain-pair, because they have the same view of the main chain-pair. If the malicious players want to diverge the view honest players, they must send new blocks regularly. From our assumption, the malicious players do not have enough resources to do that. This will bring us common prefix property.

We emphasize here, in **Case 2** or **Case 3** the malicious stakeholder may sign multiple PoS-blocks based on one PoW-block, and he may send different PoS-blocks to different honest miners. However, our scheme will not be effectively attacked by this malicious strategy. Here, all the PoS blocks (generated by the malicious stakeholder) are valid candidate bases for generating the next PoW-block. Once a PoW-block is generated based on one candidate PoS-block, all of other PoS-blocks (generated by the malicious stakeholder based on the previous PoW-block) will not be extended by any honest 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.

We also assume the malicious players may delay the messages from honest players for a while. We assume that the malicious players can delay any messages in at most Δ rounds. The intuition is that if the malicious players delay the messages, then the

honest players might not get the real best chain-pair on time. This will cause the honest miners working on a wrong chain-pair during the delayed rounds. As a result, the honest computing power is wasted during these delayed rounds. We use discounted resource to measure the actual probability that the honest players can generate a block in a round. The worst case for the chain growth property is that the malicious players will delay all honest messages to exact Δ rounds in order to decrease the probability that honest players succeed.

For simplicity, we only discuss the average case for some rounds in this section. The worst case is guaranteed by Chernoff bound if it is in a long time.

Chain growth. In order to calculate the chain growth rate, we consider the worst case for the honest players. As we discussed, the malicious players cannot prevent **Case 1**. The best strategy for the malicious players is to delay all of the messages from the honest players to discount the honest resources (computing power and stake) of honest players at most. In our scheme, there are two hops to extend a pair of blocks, so the malicious players have two chances to delay the messages. They can delay at most 2Δ rounds for a PoS-block generation. We use $\hat{\gamma}$ to denote the discounted collective honest resources where $\hat{\gamma} = \frac{\hat{\alpha}}{1+2\Delta\hat{\alpha}}$.

We use a hybrid execution to formalize the worst delay setting in the formal proof. In the hybrid execution, the malicious players contribute nothing to the chain growth and delay all honest messages to decrease the chain growth rate. We use best public chain-pair to imply the chain-pair with the longest PoW-chain and known to all honest players. It is easy to see that when **Case 1** happens, the best public chain-pair will increase by 1 block-pair (one PoW-block and one PoS-block). Probability $\hat{\gamma}$ is the probability a round is **Case 1**. The worst chain growth rate is guaranteed by $\hat{\gamma}$.

In the real execution, the probability of **Case 1** will not be smaller than that in the

hybrid execution. The message from malicious players will not decrease the chain growth that is contributed by honest players. Therefore, the real chain growth rate is not worse than that in the hybrid execution.

Chain quality. In order to decrease the chain quality, the best strategy for malicious parties is to generate as many PoS-blocks as they can. The malicious players can generate PoS-blocks in **Case 2** and **Case 3**. The total probability of these two cases for a round is $(\alpha + \beta)\tilde{\alpha}$.

During any t consecutive rounds, the chain growth rate is at least $\hat{\gamma}t$ on average. The malicious players will contribute at most $(\alpha+\beta)\tilde{\alpha}t$. The chain quality will remain at least $1 - \frac{(\alpha+\beta)\tilde{\alpha}}{\hat{\gamma}}$.

Common prefix. The adversary can delay any honest messages in at most Δ rounds, if an honest PoW-miner generates a new block, the corresponding PoS-block will be received in 2Δ later rounds by all honest players. This implies that if no honest player broadcasts new block in the past 2Δ rounds, all honest players would have same view on the blockchains, unless malicious players send new block. We say it is a silent round. A new block will effect the system in at most 2Δ rounds. We assume $2(\alpha + \beta)\Delta \ll 1$. The probability that a round is a silent round is $1 - 2(\alpha + \beta)\Delta$.

We say a round is an honest successful round if there is at least one honest player generating a new block. An honest successful round is guaranteed by α. We consider a special case of an honest successful round where only one honest miner generates a new block. We say it is a pure successful round. In the flat model, we can view the hash queries are independent. Thus, the success of one query will not effect other queries. In a successful round, there is at least one successful query. We get rid of this successful query and consider the others. The probability of a successful query of the others is still no more

than α. Thus, the probability of pure successful round is $\alpha(1-\alpha)$. It approximates to α if we assume $\alpha \ll 1$.

We assume the malicious players don't send any new block in the following case first. We consider the case that a round is a silent round and also a pure successful round. This means the successful PoW-miner will generate a unique public chain-pair with the longest PoW-chain for all honest players. If in the following 2Δ rounds, no honest PoW-miner sends a new block, then all honest players would take the same best chain-pair. This means all honest players will be convergent. The only exception is that the malicious players generate blocks to prevent this convergence. However, by our assumption, the malicious players cannot generate enough blocks to do this.

Chain soundness. In our protocol, the players select the best chain doesn't depend on any existing status they collected. This means that the new spawn player can take the same strategy as existing players.

More specifically, the new spawned player $\mathcal{S}_i$ can receive the local best chain of any existing honest player $\mathcal{S}_j$. Let $\tilde{\mathcal{C}}_i$ and $\mathcal{S}_j$ be the best local chain $\mathcal{S}_i$ and $\mathcal{S}_j$.

- If $\tilde{\mathcal{C}}_i = \tilde{\mathcal{C}}_j$, the chain soundness is guaranteed.
- If $\tilde{\mathcal{C}}_i \neq \tilde{\mathcal{C}}_j$, we have $\tilde{\mathcal{C}}_i$ is better than $\tilde{\mathcal{C}}_j$. The adversary can let another player $\mathcal{S}_m$ take $\tilde{\mathcal{C}}_i$. If the soundness property is broken then the common prefix property is broken for $\mathcal{S}_m$ and $\mathcal{S}_j$.

We emphasize that the soundness property doesn't hold directly in some pure PoS schemes such as [CM17; Kia+17; DPS17]. In these schemes, the existing players take best chain depending on the state of previous rounds. New spawned players must get help from extra assumption to get the correct state.

Addressing adaptive corruption. In our protocol, the adversary can corrupt any player adaptively at any time. The only limitation is the collective resource is honest majority. We give the analysis idea in two cases:

- In the PoW phase, adversary cannot predict which player will succeed to find a solution of PoW hash puzzle before the solution is broadcast. The adaptive corruption of PoW miner will not bring extra advantage comparing with static corruption.
- In the PoS phase, the input of hash inequality includes the signature of stake holder which is unforgeable for adversary. This will guarantee that adversary cannot predict which stake holder will be elected before the PoS block is broadcast. The adaptive corruption of stake holder will not bring extra advantage comparing with static corruption.

More details can be found in 4.2.3.

On adaptive key generation. In PoS shceme, the malicious players may generate and register their verification key vk adaptively to increase the probability that they are elected. We will prove that our scheme is secure for adaptive key generation. The intuition is we combine the PoW and PoS blocks as a pair, and the PoW block will contribute randomness for the following PoS block election. The adversary cannot generate a verification key vk adaptively before the PoW block is generated. However, after a PoW block is generated, it is late to register an adaptive key in the transaction. More details can be found in 4.2.5.

4.2.2 Important terms

We now provide some important terms which are useful for our analysis.

Definition 14 (Honest Successful Round)**.** *We say a* PoW-round r *is an honest successful round, if in this round, at least one honest PoW-miner finds a new solution.*

Definition 15 (Pure Successful Round)**.** *We say a* PoW-round r *is a pure successful round, if in this round, exact one honest PoW-miner finds a new solution.*

Definition 16 (Honest Stake Successful Round)**.** *We say a* PoS-round r *is an* honest stake successful round*, if a new PoW-block B, which is generated from an honest successful round, is mapped to an honest PoS-holder, and the PoS-holder broadcasts a new PoS-block after this round.*

Remark 3. *As discussed in previous section, a PoS-block may be invalid because the delay of the messages on the network. Here, the honest stake successful round also only consider that the real valid PoS-block is generated.*

Definition 17 (Valid Malicious PoW-block)**.** *We say a PoW-block B is a valid malicious PoW-block if (1) the PoW-block B is generated by a corrupted (malicious) PoW-miner, and (2) it is mapped to a corrupted PoS-holder.*

Definition 18 (Hidden chain-pair)**.** *We say a chain-pair $\langle \mathcal{C}, \tilde{\mathcal{C}} \rangle$ is a hidden chain-pair, if it is not known to any honest players.*

Definition 19 (Hidden PoS-chain)**.** *We say $\tilde{\mathcal{C}}$ is a hidden PoS-chain, if $\langle \mathcal{C}, \tilde{\mathcal{C}} \rangle$ is a hidden chain-pair.*

Definition 20 (Public chain-pair)**.** *We say a chain-pair $\langle \mathcal{C}, \tilde{\mathcal{C}} \rangle$ is public, if is known to all honest players.*

Definition 21 (Best Public PoS-chain)**.** *We say a PoS-chain $\tilde{\mathcal{C}}$ is the best public PoS-chain, if a) $\tilde{\mathcal{C}}$ has been received by all of the honest players, b) $\tilde{\mathcal{C}}$ is the PoS-chain of the best chain-pair among all of the public chain-pairs.*

Now we briefly analyze the probability that a PoS-round is an *honest stake successful round*. We have the probability that a round is an *honest successful round* is α. Since the new block generated in the honest successful round will be mapped to a PoS-holder uniformly by $\mathcal{F}^*_{\mathsf{PoS}}$, the probability that an honest stakeholder is selected is $\tilde{\alpha}$. Therefore, the probability that a PoS-round is an honest stake successful round is $\alpha\tilde{\alpha}$ on average.

We consider the probability that a round is a pure successful round. It is easy to see that this probability is less than the probability that a round is honest successful because in some honest successful rounds there may be more than one honest PoW-miner find a new solution. If $\alpha \ll 1$, we will see the two probabilities are close.

Lemma 3. *Let $\alpha \ll 1$. For any r, the probability that round r is a pure successful round is at least $\alpha - \alpha^2$.*

Proof. If round r is a pure successful round, round r must be an honest successful round. Suppose an honest miner finds a new solution in round r. The computing power is flat among all of the miners, get rid of one successful honest miner will not effect the total computing power very much. If the other miners find a new solution in round r, then the probability that the other miners don't find any new solution is at least $(1 - \alpha)$ on average. Putting them together, the probability that round r is a pure successful round is at least

$$\alpha(1 - \alpha) = \alpha - \alpha^2 \approx \alpha$$

□

Following a similar argument, the number of valid malicious PoW-blocks that the malicious miners will generate in a PoS-round is $\beta\tilde{\beta}$ on average. We remark that, in an honest stake successful round, the honest PoS-holder can generate a valid PoS-block for the best local chain chain-pair and send to all of the parties. The malicious stakeholders

can append the valid malicious PoS-blocks to their (hidden) chain-pair and send the new solution to some (not necessary all) honest parties.

Furthermore, the honest miners may generate a PoW-block which is mapped to a malicious stakeholder and vice versa. We describe the following cases:

- **Case 1:** When an honest PoW-miner finds a new PoW-block which is mapped to an honest PoS-holder he would broadcast the PoW-block. The chosen honest PoS-holder will query the corresponding signature to $\mathcal{F}^*_{\mathsf{PoS}}$ for that PoW-block and then produce the corresponding PoS-block. In general, this case will help the honest public best chain-pair to grow.

- **Case 2:** When a malicious PoW-miner finds a new PoW-block which is mapped to a malicious PoS-holder, he would send the block to the corresponding PoS-holder and get the PoS-block. The malicious parties may keep this pair of blocks hidden to grow a hidden chain-pair.

- **Case 3:** When an honest PoW-miner finds a new PoW-block which is mapped to a malicious PoS-holder he would still broadcast it. The malicious PoS-holders may take it to increase the length of chain-pair or just discard this block in order to prevent the growth of the best public chain-pair.

- **Case 4:** When a malicious PoW-miner finds a new PoW-block which is mapped to an honest PoS-holder. If he sends the PoW-block to the corresponding PoS-holder, then all honest parties would receive it eventually. If it is accepted, it may help to grow the best public chain-pair, or the malicious PoW-miner may just discard the block to prevent the growth of chain-pair.

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

respectively. We remark that, the malicious players may choose different strategy on the 4 cases for different purposes. For example, if they want to prevent the chain growth, they may discard as more PoW-blocks as possible. If they want to decrease the chain quality, they may recycle all of the PoW-blocks that they can "sign" (via $\mathcal{F}_{\mathsf{PoS}}^{*}$) to generate more PoS-blocks.

4.2.3 Analysis with adaptive corruption

In our model, we consider the adversary who can adaptive corrupt honest parties at any time. The only limitation is the total number of corrupted players. The collective resources are honest majority at any time.

In this section, we will prove that such adaptive corruption will not help the adversary gain any advantages other than static corruption. Static corruption means that the corruption happens at the very beginning of the protocol execution. Let's consider two different executions $\mathsf{EXEC}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$ with static corruption and $\mathsf{EXEC}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),\mathcal{A}',\mathcal{Z}}(\sigma)$ with adaptive corruption above, where $\sigma > 0$ is the randomness in the execution. Note that, the adaptive adversary $\mathcal{A}'$ has the same amount of resources as a static adversary $\mathcal{A}$, we have the following lemma:

Lemma 4. *Consider protocol* $\Pi = (\Pi^{\mathsf{w}}, \Pi^{\mathsf{s}})$ *in Figure 11. Let* $\mathsf{EXEC}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$ *denotes the execution with static corruption and* $\mathsf{EXEC}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),\mathcal{A}',\mathcal{Z}}(\sigma)$ *denotes the execution with adaptive corruption where* $\sigma > 0$ *is the randomness in the execution. Consider the two executions at round* r *are identical. It holds that the two distributions* $\mathsf{EXEC}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$ *and* $\mathsf{EXEC}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),\mathcal{A}',\mathcal{Z}}(\sigma)$ *are identical at round* $r + 1$.

Proof. If there is no new corruption instruction at round $r + 1$. The two executions are identical.

Consider that, in $\mathsf{EXEC}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),\mathcal{A}',\mathcal{Z}}(\sigma)$, the adversary send $(\textsc{Corrupt}, P)$ at round

$r+1$ and in $\mathsf{EXEC}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$, P is corrupted at the beginning.

With the functionality $\mathcal{F}^{*}_{\mathsf{PoW}}$, we have:

If P is a PoW miner, the adversary can not predict if P will generate a PoW block in round $r+1$ even adversary collect the context at round r. The probability that P generates a PoW block is identical in the two execution.

With the functionality $\mathcal{F}^{*}_{\mathsf{PoS}}$, we have:

If P is a stake holder, the adversary can not predict if P is elected to sign a PoS block in round $r+1$ even adversary collect the context at round r. The probability that P generates a PoS block is identical in the two execution. □

By Lemma 4, we only consider static corruption in the analysis.

4.2.4 Analysis with bounded delay

We assume that the malicious parties can delay messages in some bounded time through the functionality $\mathcal{F}_{\mathsf{NET}}$ which captures the asynchronous network scenario. The malicious parties can delay any messages on the network in at most Δ rounds (this is guaranteed by $\mathcal{F}_{\mathsf{NET}}$) which we say it is a Δ-bounded channel. When an honest PoW-miner finds a new PoW-block, he will broadcast it to the system and hope all parties will receive it. If some parties don't receive the message, the view of the honest stakeholders will keep remaining divergent for this block. Following a similar argument, if the adversary delays a message of PoS-block from an honest stakeholder, the view of the honest parties will diverge for that PoS-block. It is easy to see that for a pair of PoW-block and PoS-block, the malicious parties have two chances to delay the message to prevent the honest parties achieving the same view.

As discussed, if any message can be delayed for Δ rounds, a new pair of blocks can be delayed in at most 2Δ rounds. The intuition is that the delay of messages will de-

crease the efficiency of block mining. This is because the honest players may not get the real best chain-pair including the real best PoW-chain and PoS-chain and will therefore work on some inferior chain-pair. If honest players produce a new block-pair (including a PoW-block and its corresponding PoS-block) during the delay time and later receive a better chain-pair, the new block-pair will be useless and the work for mining the PoW-block in this block-pair is wasted. As mentioned, we will introduce the "effective" honest collective resources $\hat{\gamma}$ to capture this intuition later.

Before we do the formal analysis we will give the definition of a silent round.

Definition 22 (Silent Round)**.** *We say round r is a silent round, if no PoW-miners (both honest and malicious) publish any new PoW-chain in the 2Δ previous rounds of round r to any honest party.*

It is easy to see that all honest players will take the same best chain-pair at silent rounds, unless the malicious players send a better chain-pair. This is because the malicious players can delay a new solution from honest player in at most 2Δ rounds. We will show that in our parameter setting, most of the rounds are silent rounds.

Lemma 5. *Let $2(\alpha+\beta)\Delta \ll 1$. For $r > 0$ and any $\delta > 0$, the probability that round r is a silent round is $1 - 2(1+\delta)(\alpha+\beta)\Delta$ with probability at least $1 - e^{-\Omega(t)}$, where $t > 0$.*

Proof. During any t consecutive rounds, the PoW-miners will generate $X = (\alpha+\beta)t$ blocks on average. By Chernoff bound, we have $\Pr[X > (1+\delta)(\alpha+\beta)t] < 1 - e^{-\Omega(t)}$.

Any new blocks may effect at most 2Δ rounds. This means, there are at most $2X\Delta$ rounds that are not silent in t rounds. The total number of silent rounds in t rounds is $t - 2X\Delta$. If t is large, we ignore the bad event that $X > (1+\delta)(\alpha+\beta)t$. The probability that a round is silent round in t rounds is:

$$\frac{t - 2X\Delta}{t} > 1 - 2(1+\delta)(\alpha+\beta)\Delta$$

. □

4.2.4.1 Hybrid expriment

To analyze the best strategy of the adversary and the worst scenario that may happen to the honest players, we here consider the following executions. Let $\mathsf{REAL}(\sigma) = \mathsf{EXEC}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$ denote the typical execution of $(\Pi^{\mathsf{w}}, \Pi^{\mathsf{s}})$ where

1. $\sigma > 0$ is the randomness in the execution,
2. Messages of honest players may be delayed by $\mathcal{F}_{\mathsf{NET}}$ in at most Δ rounds.

Without loss of generality, we assume that the messages produced in PoW-rounds may be delayed to PoS-rounds, and messages produced in PoS-rounds may be delayed to PoW-rounds.

Let $\mathsf{HYB}_r(\sigma) = \mathsf{EXEC}^r_{(\Pi^{\mathsf{w}}_\Delta,\Pi^{\mathsf{s}}_\Delta),\mathcal{A},\mathcal{Z}}(\sigma)$ denote the hybrid execution as in real execution except that after round r, $\mathsf{HYB}_r(\sigma)$ has the following modifications from $\mathsf{REAL}(\sigma)$:

1. The randomness is fixed to σ as in $\mathsf{HYB}_r(\sigma)$,
2. $\mathcal{F}_{\mathsf{NET}}$ delays messages generated by *honest* PoW-miners to exact Δ rounds if the new PoW-block is mapped to an honest stakeholder,
3. $\mathcal{F}_{\mathsf{NET}}$ delays all messages generated by *honest* PoS-holders to exact Δ rounds,
4. Remove all new messages sent by the adversary to honest players, and delay currently undelivered messages from corrupted parties to the maximum of Δ rounds,
5. Whenever some message is being delayed, no *honest* PoW-miners query the functionality $\mathcal{F}^*_{\mathsf{PoW}}$ until the message is delivered.

In the $\mathsf{REAL}(\sigma)$ executions, the number of honest stake successful rounds is not less than in the $\mathsf{HYB}_r(\sigma)$.

The following lemma shows that the PoS-chain of every honest player cannot decrease in length if we maximally delay messages from honest parties, freeze all honest players during this delay, and drop all adversarial messages for every fixed randomness σ. This means the hybrid execution is not worse than real execution.

Lemma 6. *For all $\sigma, r, t > 0$, given two executions $\mathsf{REAL}(\sigma)$ and $\mathsf{HYB}_r(\sigma)$. Let $s = r + t$. For any honest PoS-holder $\mathcal{S}_j$ at round s, let $\tilde{\mathcal{C}}_{s,j}$ denote the PoS-chain of $\mathcal{S}_j$ at round s in the execution $\mathsf{REAL}(\sigma)$ and $\tilde{\mathcal{C}}'_{s,j}$ denote the PoS-chain of $\mathcal{S}_j$ at round s in the $\mathsf{HYB}_r(\sigma)$. We then have $\mathsf{len}(\tilde{\mathcal{C}}_{s,j}) \geq \mathsf{len}(\tilde{\mathcal{C}}'_{s,j})$.*

Proof. We prove this lemma by induction. We consider the initial state before round r. From the definition of hybrid experiment, all players have same view at round r. We have $\mathsf{len}(\tilde{\mathcal{C}}_{\cdot,\cdot}) \geq \mathsf{len}(\tilde{\mathcal{C}}'_{\cdot,\cdot})$. We use $\cdot$ to denote all players and all rounds before round r.

We suppose it holds for all players before round $s-1$. The only possibility that $\mathsf{len}(\tilde{\mathcal{C}}_{s,j}) < \mathsf{len}(\tilde{\mathcal{C}}'_{s,j})$ is the player $\mathcal{S}_j$ received a new chain to extend $\tilde{\mathcal{C}}'_{s,j}$ at round s in $\mathsf{HYB}_r(\sigma)$. According to the definition of hybrid experiment, this extended PoW-block must be generated at round $s - 2\Delta$ by an honest player $\mathcal{W}_{j'}$, that makes $\mathsf{len}(\tilde{\mathcal{C}}'_{s,j}) = \mathsf{len}(\tilde{\mathcal{C}}'_{s-2\Delta,j'}) + 1$. By the induction hypothesis, $\mathsf{len}(\tilde{\mathcal{C}}_{s-2\Delta,j'}) \geq \mathsf{len}(\tilde{\mathcal{C}}'_{s-2\Delta,j'})$. At the same time, the player $\mathcal{W}_{j'}$ must succeed to extend PoW-block at round $s - 2\Delta$ in $\mathsf{REAL}(\sigma)$. This extension will make $\tilde{\mathcal{C}}_{s-2\Delta,j'}$ increase by one block. For player $\mathcal{W}_{j'}$ is honest, $\mathcal{S}_j$ must have received the extension at (or before) round s.

Putting them together, $\mathsf{len}(\tilde{\mathcal{C}}_{s,j}) \geq \mathsf{len}(\tilde{\mathcal{C}}'_{s,j})$. □

4.2.4.2 Analysis in the worst delay setting

As mentioned earlier, the malicious players can delay the messages on the network in at most Δ rounds. This will make some efforts of honest players be wasted. This means the network delay will discount the collective resources of honest players, the following

lemma will measure the discount in the execution of $\mathsf{HYB}_r(\sigma)$.

Lemma 7. *Consider* $\mathsf{HYB}_r(\sigma)$. *Let* $\alpha, \tilde{\alpha} > 0$. *We assume that the malicious players will delay any message for at most* Δ *rounds. Let* $\hat{\gamma}$ *be the actual probability that a round* $s > r$ *is an honest successful stake round, we have* $\hat{\gamma} = \frac{\hat{\alpha}}{1+2\Delta\hat{\alpha}}$.

Proof. Consider the $\mathsf{HYB}_r(\sigma)$ execution, if round $r' > r$ is an *honest stake successful round*, then all PoW-miners will not query $\mathcal{F}^*_{\mathsf{PoW}}$ during 2Δ rounds.

Now, assume there are actual c honest stake successful rounds from round r to $r + t$,$t > 0$ in $\mathsf{HYB}_r(\sigma)$. We then have the number of actual working rounds for honest miners will remain $t - 2\Delta c$. For each round, the probability that it is an honest stake successful round is $\hat{\alpha}$. We have $\hat{\alpha}(t - 2\Delta c) = c$. This implies that $c = \frac{\hat{\alpha}t}{1+2\Delta\hat{\alpha}} = \hat{\gamma}t$. We get $\hat{\gamma} = \frac{\hat{\alpha}}{1+2\Delta\hat{\alpha}}$. □

Let view_r denote the view at round r for any execution of $\mathsf{REAL}(\sigma)$ and $r > 0$. Let $\mathsf{len}(\mathsf{view}_r)$ denote the length of the best public PoS-chain chain at round r in view_r. The following lemma demonstrates that each stake successful round would contribute one PoS-block to the best chain-pair after 2Δ rounds in an execution of $\mathsf{HYB}_r(\sigma)$.

Lemma 8. *Consider execution* $\mathsf{HYB}_r(\sigma)$. *For any honest stake successful round* $s > r$, $\mathsf{len}(\mathsf{view}_s) - \mathsf{len}(\mathsf{view}_{s-2\Delta}) \geq 1$.

Proof. By Definition 16, there is at least one honest PoS-holder producing a PoS-block at round s. Let $\tilde{\mathcal{C}}_{s-2\Delta}$ be the PoS-chain that is extended by the PoS-holder at round $s - 2\Delta$. We have $\mathsf{len}(\tilde{\mathcal{C}}_{s-2\Delta}) \geq \mathsf{len}(\mathsf{view}_{s-2\Delta})$. Let $\tilde{\mathcal{C}}_s$ be the PoS-chain that is extended by the PoS-holder at round s. At the end of round s, all honest players will receive the extended chain, we have $\mathsf{len}(\tilde{\mathcal{C}}_s) = \mathsf{len}(\tilde{\mathcal{C}}_{s-2\Delta}) + 1$. Thus, we have $\mathsf{len}(\mathsf{view}_s) \geq \mathsf{len}(\tilde{\mathcal{C}}_s)$. Putting them together, we have $\mathsf{len}(\mathsf{view}_s) - \mathsf{len}(\mathsf{view}_{s-2\Delta}) \geq 1$. □

Corollary 7. *Consider execution* $\mathsf{HYB}_r(\sigma)$. *Suppose there are* h *honest stake successful rounds from round* r *to round* $r+t$*, it holds that* $\mathsf{len}(\mathsf{view}_{r+s}) - \mathsf{len}(\mathsf{view}_r) \geq h$.

Proof. Let r_k be the kth honest stake successful round where $r < r_k < r+t$ and $1 \leq k \leq h$. From Lemma 8, we have $\mathsf{len}(\mathsf{view}_{r_k}) - \mathsf{len}(\mathsf{view}_{r_k-\Delta}) \geq 1$.

$\mathsf{len}(\mathsf{view}_{r+t}) - \mathsf{len}(\mathsf{view}_r) \geq \sum_{i=1}^{h}\{\mathsf{len}(\mathsf{view}_{r_k}) - \mathsf{len}(\mathsf{view}_{r_k-\Delta})\} \geq h$ □

4.2.5 Analysis with adaptive key generation

In the protocol Π^{s}, PoS-holder is elected by the ideal functionality $\mathcal{F}^*_{\mathsf{PoS}}$ with some probability. In the implementation, if a Π^{s} is elected in a round depends on his account (public key) and some auxiliary inputs. If a malicious players can choose and register a biased key after he collects all of the information of the environment, he may get some advantage to be elected. We call this type of malicious players as adaptive malicious players and this attack strategy as adaptive key generation attack.

First we will show that adaptive key generation attack will take effect on pure proof of stake blockchain scheme under some assumption.

Lemma 9. *Let* Π *be any pure proof stake blockchain scheme. The player election function is* $F(\cdot, pk)$*. Let* p *be the probability that a malicious player is elected in a round if there is no adaptive key generation attack. That is* $\Pr[F(\cdot, pk) = 1] = p$ *for any* pk*. Let* p_a *be the probability that a malicious player is elected in a round with adaptive key generation attack. If* $F(\cdot, pk)$ *can be predicted by players and the key registration procedure doesn't affect the input of* $F(\cdot, pk)$ *, we have* $p_a = 1 - (1-p)^{ploy(\kappa)}$.

Proof. For adaptive malicious players, he can generate $ploy(\kappa)$ different public keys in a round. He will predict $F(\cdot, \mathrm{PK})$ for all of the $ploy(\kappa)$ public keys. The probability that all of the $ploy(\kappa)$ keys are not be elected is $(1-p)^{ploy(\kappa)}$. We have $p_a = 1 - (1-p)^{ploy(\kappa)}$. He then register the elected key if there is one.

□

Our 2hop blockchain is immune to adaptive key generation attack. The intuition is in our stake election function is $F(\cdot, \text{PK})$, key registration procedure will affect the input $\cdot$. In our scheme, the concrete input $\cdot$ will contain the output of PoW block. The key registration transaction will be included in the previous PoW blocks. If the malicious players want to register a key adaptively, he must try PoW which will still consume the computing power.

Lemma 10. *Consider the execution* $\mathsf{EXEC}_{(\Pi^{\text{w}},\Pi^{\text{s}}),\mathcal{A},\mathcal{Z}}$. *Let* $\beta, \tilde{\beta} > 0$. *We assume that the malicious players will take adaptive key generation strategy. Let* $\tilde{\gamma}$ *be the expected number of PoW blocks which are mapped to malicious stakes that malicious players can find in round. We have* $\tilde{\gamma} = \beta\tilde{\beta}$.

Proof. In our scheme, when a valid PoW block is generated, all of the public key of stake holders are fixed in the previous blocks. Any change of public key (registration of new public key) will make the PoW block be invalid. If the malicious players take adaptive strategy, he must redo the PoW hash puzzle. That is adaptive strategy will not bring advantage to the malicious players comparing with the original strategy. We have $\tilde{\gamma} = \beta\tilde{\beta}$. □

That is in our protocol the key registration will affect the input of PoW block, the malicious players can not take adaptive strategy. How ever in pure proof of stake blockchain scheme, the key registration is almost free, the malicious players may can register key adaptively. In order to avoid this attack, there must be some extra assumption in PoS blockchain, such as a key must have been registered several rounds before it can be elected.

4.2.6 Achieving the chain growth property

We here demonstrate that our protocol satisfies the chain growth property for PoS-chain (Definition 10). The concrete statement to be proved can be found in Theorem 3. We first prove that our blockchain protocol achieves the chain growth property in the execution $\mathsf{HYB}_r(\sigma)$ before moving to the main theorem.

Lemma 11. *Consider the execution* $\mathsf{HYB}_r(\sigma)$*. For any* $\delta > 0$*, during any* t *consecutive rounds after round* r*, the number of honest successful rounds is* $(1-\delta)\hat{\gamma}t$ *with probability at least* $1 - e^{-\Omega(t)}$*.*

Proof. We will prove that honest stake successful rounds will happen following the honest collective resources. From Lemma 7, in any t consecutive rounds, the number of honest stake successful rounds is $\hat{\gamma}t$ on average. Let X be the number of honest stake successful rounds in the t consecutive rounds. By Chernoff bound, we have $\Pr[X \leq (1-\delta)\hat{\gamma}t] \leq e^{-\delta^2\hat{\gamma}t/2}$.

Thus, $\Pr[X > (1-\delta)\hat{\gamma}t] > 1 - e^{-\delta^2\hat{\gamma}t/2} = 1 - e^{-\Omega(t)}$. □

The honest stake successful rounds will increase the length of the PoS-chain of best public chain-pair.

Lemma 12. *Consider the execution* $\mathsf{HYB}_r(\sigma)$*. For any* $\delta > 0$*, and for any honest PoS-holder* $\mathcal{S}$ *with the best PoS-chain* $\tilde{\mathcal{C}}_r$ *and* $\tilde{\mathcal{C}}_{r'}$ *in round* r *and* r'*, respectively, where* $t = r' - r \gg \Delta$*, the probability that* $\mathsf{len}(\tilde{\mathcal{C}}_{r'}) - \mathsf{len}(\tilde{\mathcal{C}}_r) \geq g \cdot t$ *where* $g = (1-\delta)\hat{\gamma}$ *is at least* $1 - e^{-\Omega(t)}$*.*

Proof. For $\mathcal{S}$ is honest, $\tilde{\mathcal{C}}_r$ will be received by all honest players no later than round $r + \Delta$. We have $\mathsf{len}(\tilde{\mathcal{C}}_r) \leq \mathsf{len}(\mathsf{view}_{r+\Delta})$. For $t \gg \Delta$, we consider $t \approx t - \Delta$ for simplicity. From Lemma 11, in any t consecutive rounds the number of honest successful round is more than $(1-\delta)\gamma t$ with the probability at least $1 - e^{-\Omega(t)}$. Together with Lemma 8 and Corollary 7, we have $\mathsf{len}(\mathsf{view}_{r'}) - \mathsf{len}(\mathsf{view}_{r+\Delta}) \geq (1-\delta)\hat{\gamma}t$.

Chain $\tilde{\mathcal{C}}_{r'}$ is the best valid PoS-chain accepted by the honest PoS-holder $\mathcal{S}_j$ at round r'. We have $\mathsf{len}(\tilde{\mathcal{C}}_{r'}) \geq \mathsf{len}(\mathsf{view}_{r'})$. Put them together, $\mathsf{len}(\tilde{\mathcal{C}}_{r'}) - \mathsf{len}(\tilde{\mathcal{C}}_r) \geq \mathsf{len}(\mathsf{view}_{r'}) - \mathsf{len}(\mathsf{view}_{r+\Delta}) \geq (1-\delta)\hat{\gamma}t$ with probability at least $1 - e^{-\Omega(t)}$. The corresponding growth rate is $g = (1-\delta)\hat{\gamma}$. □

Reminder of Theorem 3. *Consider protocol $\Pi = (\Pi^{\mathsf{w}}, \Pi^{\mathsf{s}})$ in Section 4.1.2. For any honest PoS-holder $\mathcal{S} \in \{\mathcal{S}_{n+1}, \ldots, \mathcal{S}_{n+\tilde{n}}\}$ with the local PoS-chain $\tilde{\mathcal{C}}$ in round r and $\tilde{\mathcal{C}}'$ in round r' where $t = r' - r > 0$, in $\mathsf{EXEC}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}$, the probability that $\mathsf{len}(\tilde{\mathcal{C}}') - \mathsf{len}(\tilde{\mathcal{C}}) \geq g \cdot t$ is at least $1 - e^{-\Omega(t)}$ where $g = (1-\delta)\hat{\gamma}$.*

Proof. In order to distinguish the notation clearly, we use $\tilde{\mathcal{C}}_\Delta$ and $\tilde{\mathcal{C}}'_\Delta$ to denote the PoS-chains of the best chain-pairs of $\mathcal{S}$ at round r ans r' in the execution of $\mathsf{HYB}_r(\sigma)$.

From Lemma 12, we have $\Pr[\mathsf{len}(\tilde{\mathcal{C}}'_\Delta) \geq \mathsf{len}(\tilde{\mathcal{C}}_\Delta) + g \cdot t] \geq 1 - e^{-\Omega(t)}$ where $t = r' - r$, in $\mathsf{HYB}_r(\sigma)$.

We now turn to the chain growth property in $\mathsf{EXEC}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}$. From the definition of hybrid execution, we know that all honest players have same initial status at round r. We have $\mathsf{len}(\tilde{\mathcal{C}}) = \mathsf{len}(\tilde{\mathcal{C}}_\Delta)$.

By Lemma 6, we have $\mathsf{len}(\tilde{\mathcal{C}}') \geq \mathsf{len}(\tilde{\mathcal{C}}'_\Delta)$.

It follows that,

$$\begin{aligned} &\Pr[\mathsf{len}(\tilde{\mathcal{C}}') \geq \mathsf{len}(\tilde{\mathcal{C}}) + g \cdot t] \\ &\geq \Pr[\mathsf{len}(\tilde{\mathcal{C}}'_\Delta) \geq \mathsf{len}(\tilde{\mathcal{C}}_\Delta) + g \cdot t] \\ &\geq 1 - e^{-\Omega(t)} \end{aligned} \tag{4.1}$$

where $g = (1-\delta)\hat{\gamma}$. This completes the proof. □

4.2.7 Achieving the chain quality property

The chain-quality property (Definition 12) ensures that the rate of honest input contributions in a continuous part of an honest party's chain has a lower bound. We then find the lower bound of the number of PoS-blocks produced by the honest players. We further show that the number of blocks produced by the adversarial miners is bounded by their collective resources. Finally, we demonstrate that the ratio of honest PoS-blocks in an honest player's PoS-chain is under a suitable lower bound in a sufficient number of rounds with an overwhelming probability.

First, we will build the relationship between length of a chain and the number of rounds.

Lemma 13. *Consider in* $\mathsf{REAL}(\sigma)$. *For any* $\delta > 0$, *let* Z *be the number of rounds which generate* ℓ *blocks, we then have* $\Pr[Z > c\ell] > 1 - e^{-\Omega(\ell)}$, *where* $c \gg 1$.

Proof. Putting all of the resources together, all players can generate $\hat{\alpha} + \hat{\beta}$ PoS-blocks in a round on average. In order to generate ℓ blocks, it will consume $\frac{\ell}{\hat{\alpha}+\hat{\beta}}$ rounds on average.

Let $c = \frac{1}{\hat{\alpha}+\hat{\beta}}$, and Z be the number of rounds which generate ℓ PoS-blocks. For any $\delta > 0$, by using Chernoff bounds, we have $\Pr[Z \leq (1-\delta)c\ell] \leq e^{-\delta^2 c\ell/3}$. That is, $\Pr[Z > (1-\delta)c\ell] > 1 - e^{-\delta^2 c\ell/3} = 1 - e^{-\Omega(\ell)}$. This completes the proof. □

Now we consider the contribution of PoS-blocks from honest players in some consecutive rounds. If the adversarial players want to contribute more PoS-blocks on the chain, they will try to generate more PoS-blocks and beat the PoS-blocks from honest players in the competition. Thus, the worst case is the adversarial players make use of all the computing power from both honest and malicious miners to generate PoS-blocks and win all of the competition. We recall the 4 cases in Section 4.2.2. The adversarial players can make use of **Case 2** and **Case 3** to generate PoS-blocks. The honest players will use **Case**

1 to generate PoS-blocks. First, we will prove the chain quality property in t consecutive rounds.

Lemma 14. *Consider in* $\mathsf{REAL}(\sigma)$ *and consider* ℓ *PoS-blocks of* $\tilde{\mathcal{C}}$ *that are generated in consecutive rounds from* r *to* $r+t$*. We assume* $\hat{\gamma} = \hat{\lambda}(\alpha+\beta)\tilde{\beta}$ *where* $\hat{\lambda} > 1$ *and any* $\delta > 0$*. Then for any honest PoS-holder* $\mathcal{S}$ *with PoS-chain* $\tilde{\mathcal{C}}$*, we have* $\Pr[\mu \geq 1-(1+\delta)\frac{(\alpha+\beta)\tilde{\beta}}{\hat{\gamma}}] > 1-e^{-\Omega(t)}$*, for any* $\delta > 0$*, where* μ *is the ratio of honest blocks the PoS-chain* $\tilde{\mathcal{C}}$*.*

Proof. Consider ℓ consecutive PoS-blocks of $\tilde{\mathcal{C}}$ that are generated from round r to round $r+t$. From Theorem 3, we have $\Pr[\ell \geq (1-\delta^*)\hat{\gamma}\cdot t] \geq 1-e^{-\Omega(t)}$ for some $\delta^* > 0$.

Let Y be the number of valid malicious PoW-blocks which are actually generated in t rounds in the **Case 2**. By Chernoff bound, we have

$$\Pr[Y < (1+\delta')\beta\tilde{\beta}\cdot t] > 1-e^{-\Omega(t)}$$

Furthermore, let Z be the number of blocks of generated in **Case 3** in t rounds. By Chernoff bound, we have

$$\Pr[Z < (1+\delta'')\alpha\tilde{\beta}\cdot t] > 1-e^{-\Omega(t)}$$

Putting them together, the malicious parties will contribute at most $(Y+Z)$ PoS-blocks in the t rounds. We then have

$$\Pr\left[\mu \geq \frac{\ell - Y - Z}{\ell}\right] > 1-e^{-\Omega(t)}$$

That is, By picking δ^*, δ', and δ'' sufficiently small, we have

$$\Pr\left[\mu \geq 1-(1+\delta)\frac{(\alpha+\beta)\tilde{\beta}}{\hat{\gamma}}\right] > 1-e^{-\Omega(t)}$$

for any $\delta > 0$. This completes the proof. □

Now we are ready to prove the chain quality property for consecutive blocks on a

chain.

Reminder of Theorem 4. *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 4.1.2. For any honest PoS-holder $\mathcal{S} \in \{\mathcal{S}_{n+1}, \ldots, \mathcal{S}_{n+\tilde{n}}\}$ with PoS-chain $\tilde{\mathcal{C}}$ in $\mathsf{EXEC}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}$, the probability that, for large enough ℓ consecutive PoS-blocks of $\tilde{\mathcal{C}}$ which are generated in s rounds, the ratio of honest blocks is no less than $\mu = 1-(1+\delta)\frac{(\alpha+\beta)\tilde{\beta}}{\hat{\gamma}}$ is at least $1-e^{-\Omega(\ell)}$.*

Proof. Let t be the rounds that the ℓ blocks are generated. From Lemma 13, we have $\Pr[t > c\ell] > 1-e^{-\Omega(\ell)}$.

From Lemma 14, the ratio of honest PoS-blocks in t consecutive rounds with ℓ PoS-blocks is $\mu \geq 1-(1+\delta)\frac{(\alpha+\beta)\tilde{\beta}}{\hat{\gamma}}$ with probability at least $1-e^{-\Omega(t)}$.

Putting them together, the probability is at least $1-e^{-\Omega(\ell)}$. This completes the proof. □

4.2.8 Achieving the common prefix property

We now turn our attention to proving the common prefix property for PoS-chain (Definition 11) for the proposed protocol. The concrete statement can be found in Theorem 5.

Now we will give some informal proof ideas before the formal proof.

- First, from the assumption, we know that if the malicious parties do not get any help from the honest parties, then they cannot produce PoS-blocks faster than the honest parties. That means if the malicious parties keep a forked chain-pair hidden and try to extend it by themselves, then the growth rate of the hidden chain-pair is smaller than the growth rate of the public longest chain-pair on average. When considering an extended period of time, the hidden chain-pair will be shorter than the public chain-pair with an overwhelming probability.

- Second, we assume there is no new block being generated in most rounds. This implies no new chain will lead the honest players take divergent view in most rounds. The honest players will try to be convergent after some silent rounds. If the adversary players want to keep them be divergent, they must send new blocks in silent rounds. The adversary players don't have enough resources to do that.

Recall the definition of best public PoS-chain. Best public chain $\tilde{\mathcal{C}}$ is: a) $\tilde{\mathcal{C}}$ has been received by all of the honest players which means public. b) $\tilde{\mathcal{C}}$ is the best one among all of the public chains. This implies each honest player will not take any chain worse than best public chain in any round . Next we will prove, it the adversary players hide some blocks for a chain, the hidden chain will be worse than the best public chain with high probability if the hidden length is long. This lemma imply that is the adversary players keep some blocks privacy, they will be invalid soon. So the adversarial players cannot store a lot of hidden blocks to destroy the best PoS-chain later.

Lemma 15. *Let $\hat{\gamma} = \hat{\lambda}\hat{\beta}$ and $\hat{\lambda} > 1$. For any $\delta > 0$, consider the execution $\mathsf{REAL}(\sigma)$. Let $\tilde{\mathcal{C}}$ be the PoS-chain of the best public chain-pair in round r. Let $\tilde{\mathcal{C}}'$ be the PoS-chain of a hidden valid chain-pair in round r. Let ℓ be the length of the hidden part of $\tilde{\mathcal{C}}'$. We have* $\Pr[\mathsf{len}(\tilde{\mathcal{C}}) > \mathsf{len}(\tilde{\mathcal{C}}')] > 1 - e^{\Omega(\ell)}$.

Proof. Let round $s = r - t$ be the round that last public block in $\tilde{\mathcal{C}}'$ is generated . From Lemma 13, ℓ hidden blocks need t rounds to generate with probability at least $1 - e^{\Omega(\ell)}$. That is, $\Pr[t > c\ell] > 1 - e^{\Omega(\ell)}$.

The hidden blocks are contributed by adversarial players only, otherwise they are not hidden. Thus, the growth of hidden blocks is from the **Case 2**. In t rounds, the adversarial players can generate $\hat{\beta}t$ PoS-blocks on average. Let X be the number of adversarial blocks generated in t rounds, by Chernoff bound, we have

$$\Pr[X > (1+\delta)\hat{\beta}t] < e^{-\Omega(t)}$$

From Theorem 3, during t rounds the best public PoS-chain will increase $Y > (1-\delta)\hat{\gamma}t$ blocks with probability at least $1 - e^{-\Omega(t)}$. For $\hat{\gamma} = \hat{\lambda}\hat{\beta}$, we have

$$\Pr[X < Y] > 1 - e^{-\Omega(t)} = 1 - e^{\Omega(\ell)}$$

We denote chain $\tilde{\mathcal{C}}'$ at round s as $\tilde{\mathcal{C}}'^s$. We have

$$\mathsf{len}(\tilde{\mathcal{C}}) > \mathsf{len}(\tilde{\mathcal{C}}'^s) + Y > \mathsf{len}(\tilde{\mathcal{C}}'^s) + X > \mathsf{len}(\tilde{\mathcal{C}}')$$

with probability at least $1 - e^{-\Omega(\ell)}$.

□

We first prove the common prefix property for any t consecutive rounds. The proof intuition is as follows:

- We assume $\alpha + \beta \ll 1$, that means in most rounds, players will not generate block.
- If only honest players broadcast a new PoW-block, in 2Δ silent rounds, all honest players will take the same best chain-pair (or PoS-chain), unless adversarial players send new valid blocks.
- If $(\alpha + \beta)\Delta \ll 1$, there are no new messages being broadcast in most rounds.
- From a round, the honest players will often have opportunities to take unique best chain-pair (or PoS-chain). If the adversarial players want to keep them divergent, they must broadcast new valid blocks for every opportunity.
- We will prove the adversarial players don't have enough resource to do that under our reasonable assumption.

Lemma 16. *Let* $\hat{\alpha} = \hat{\lambda}(\alpha + \beta)\tilde{\beta}$, $\hat{\lambda} > 1$, $(\alpha + \beta)\Delta \ll 1$. *Assume* $0 < \delta, \delta', \delta'', \delta''' < 1$, *consider the execution* $\mathsf{REAL}(\sigma)$. *Except with probability* $e^{-\Omega(t)}$, *there does not exist round*

$r \leq r'$ and PoS-holders $\mathcal{S}_i$, $\mathcal{S}_j$ such that $\mathcal{S}_i$ is honest at r, $\mathcal{S}_j$ is honest at r' and $\tilde{\mathcal{C}}_i^r$ and $\tilde{\mathcal{C}}_j^{r'}$ diverge at round $s = r - t$.

Proof. For each round k, we define a random variable $\mathbf{X}_k$. If round k is both pure successful round and silent round, and round $k + 2\Delta$ is also a silent round $\mathbf{X}_k = 1$, otherwise $\mathbf{X}_k = 0$. Let $X = \sum \mathbf{X}_k$.

Let X' be the number of pure successful rounds in t round. By Lemma 3, X' is $(\alpha - \alpha^2)t$ on average. By Chernoff bound, we have

$$\Pr[X' < (1 - \delta)(\alpha - \alpha^2)t] < e^{-\Omega(t)}$$

Let X'' be the number rounds which are both pure successful round and silent round. A round is pure successful round is independent with the event it is silent round. By Lemma 5, we have

$$\Pr[X'' < (1 - \delta)(\alpha - \alpha^2)t(1 - 2(1 + \delta')(\alpha + \beta)\Delta)] < e^{-\Omega(t)}$$

For $(\alpha + \beta)\Delta \ll 1$ and $\alpha \ll 1$, $\exists \delta''$ $s.t.$

$$\Pr[X'' < (1 - \delta'')\alpha t] < e^{-\Omega(t)}$$

A round k is also independent with the event round $k + 2\Delta$ is a silent round. Also by Lemma 5, we have

$$\Pr[X < (1 - \delta''')\alpha t] < e^{-\Omega(t)}$$

If $\mathbf{X}_k = 1$, the new generated block in round k will be mapped to an honest stakeholder with the probability $\tilde{\alpha}$. We use a random variable $\mathbf{Y}_k$ for round k. If $\mathbf{X}_k = 1$ and the new generated block is mapped to an honest stakeholder $\mathbf{Y}_k = 1$, otherwise $\mathbf{Y}_k = 0$. Let $Y = \sum \mathbf{Y}_k$. We have

$$\Pr[Y < (1 - \delta''')\alpha\tilde{\alpha}t] < e^{-\Omega(t)}$$

If $\mathbf{Y}_k = 1$ all honest players will be convergent to a unique PoS-chain unless adversarial players send a new block in the next 2Δ rounds. This is because, a) at round r all honest players have the best PoS-chain with same length. b) at round r the new extended PoS-block will increase the best public PoS-chain by 1 block. If there is no adversarial players send a new chain $\tilde{\mathcal{C}}'$ with at least the same length, all honest parties will receive the new PoS-chain in 2Δ rounds and take it as the best chain. The last block of $\tilde{\mathcal{C}}'$ must be generated by adversarial players because if is longer than the best public PoS-chain of honest players.

Let Z' be the number of blocks generated by adversarial players in t consecutive rounds. We have $Z' = (\alpha + \beta)\tilde{\beta}t$ on average. By Chernoff bound, we have $\Pr[Z' > (1+\delta)(\alpha+\beta)\tilde{\beta}t] < e^{-\Omega(t)}$. From Lemma 15, before round s, the adversarial players can hide at most κ blocks with an overwhelming probability in κ. Let $Z = Z' + \kappa$, the adversarial players can use Z new blocks to prevent the Y convergence opportunities. If t is large enough, we can have δ' that $\Pr[Z > (1+\delta')(\alpha+\beta)\tilde{\beta}t] < e^{-\Omega(t)}$.

Putting them together, $Y > (1-\delta''')\alpha\tilde{\alpha}t$ and $Z < (1+\delta')(\alpha+\beta)\tilde{\beta}t$ with probability at least $1 - e^{-\Omega(t)}$. By the assumption $\alpha\tilde{\alpha} = \hat{\lambda}(\alpha+\beta)\tilde{\beta}$ and $\hat{\lambda} > 1$, we have

$$\begin{aligned} Y - Z &> (1-\delta''')\alpha\tilde{\alpha}t - (1+\delta')(\alpha+\beta)\tilde{\beta}t \\ &= ((1-\delta''')\hat{\lambda} - (1+\delta'))(\alpha+\beta)\tilde{\beta}t \\ &> 0 \end{aligned} \tag{4.2}$$

This implies that the adversarial players cannot prevent the best chain-pair (or PoS-chain) being convergent during the t consecutive rounds with probability at least $1 - e^{-\Omega(t)}$. If at round r, $\tilde{\mathcal{C}}_i^r$ and $\tilde{\mathcal{C}}_j^r$ are not divergent, then $\tilde{\mathcal{C}}_i^r$ and $\tilde{\mathcal{C}}_j^{r'}$ are not divergent. We have $\tilde{\mathcal{C}}_i^r$ and $\tilde{\mathcal{C}}_j^{r'}$ diverge with the probability at most $e^{-\Omega(t)}$. □

We are now ready to prove the main theorem which asserts that our protocol achieves

the common-prefix property with an overwhelming probability in the security parameter κ. The theorem is formally given as follows.

Reminder of Theorem 5. *We assume $\hat{\alpha} = \hat{\lambda}(\alpha+\beta)\tilde{\beta}$ and $\hat{\lambda} > 1$. For any $\delta > 0$, consider protocol $\Pi = (\Pi^{\mathsf{w}}, \Pi^{\mathsf{s}})$ in Section 4.1.2. Let κ be the security parameter. For any two honest PoS-holders $\mathcal{S}_i$ in round r and $\mathcal{S}_j$ in round r', with the local best PoS-chains $\tilde{\mathcal{C}}_i$, $\tilde{\mathcal{C}}_j$, respectively, in $\mathsf{EXEC}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}$ where $r \leq r'$ and $i, j \in \{n+1, \ldots, n+\tilde{n}\}$, the probability that $\tilde{\mathcal{C}}_i[1, \ell_i] \preceq \tilde{\mathcal{C}}_j$ where $\ell_i = \mathsf{len}(\tilde{\mathcal{C}}_i) - \Theta(\kappa)$ is at least $1 - e^{-\Omega(\kappa)}$.*

Proof. From Lemma 16, the probability that $\tilde{\mathcal{C}}_i$ and $\tilde{\mathcal{C}}_j$ diverge at round $s = r - t$ is at most $e^{-\Omega(t)}$.

In t consecutive rounds, the total number of PoS-blocks are produced is bounded by $(1+\delta)(\alpha+\beta)(\tilde{\alpha}+\tilde{\beta})t$ with probability at least $1 - e^{-\Omega(t)}$. Let $t = \frac{\kappa}{(1+\delta)(\alpha+\beta)(\tilde{\alpha}+\tilde{\beta})}$. We have $\tilde{\mathcal{C}}_i$ is prefix of $\tilde{\mathcal{C}}_j$ except the last κ blocks with the probability at least $1 - e^{-\Omega(\kappa)}$. □

4.2.9 Achieving the chain soundness property

The chain soundness property will guarantee that a new spawned player can take the best chain properly when he join the system. It is critical to achieve a open blockchain system. The concrete statement can be found in Theorem 6.

Reminder of Theorem 6. *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 4.1.2. Let κ be the security parameter. For any new spawned honest player PoS-holder $\mathcal{S}_i$ and any existing honest player $\mathcal{S}_j$ in round r, with the local best PoS-chains $\tilde{\mathcal{C}}_i$, $\tilde{\mathcal{C}}_j$, respectively, in $\mathsf{EXEC}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}$, the probability that $\tilde{\mathcal{C}}_i[1, \ell] \preceq \tilde{\mathcal{C}}_j$ and $\tilde{\mathcal{C}}_j[1, \ell] \preceq \tilde{\mathcal{C}}_i$ where $\ell = \mathsf{len}(\tilde{\mathcal{C}}_i) - \Theta(\kappa)$ is at least $1 - e^{-\Omega(\kappa)}$.*

Proof. We note that $\mathcal{S}_j$ is an honest existing player, $\mathcal{S}_i$ must receive $\tilde{\mathcal{C}}_j$ in round r. Because $\mathcal{S}_i$ take $\tilde{\mathcal{C}}_i$ as best chain we have $\mathsf{len}(\tilde{\mathcal{C}}_i) \geq \mathsf{len}(\tilde{\mathcal{C}}_j)$.

Consider the scenario that the adversary let another existing player $\mathcal{S}_m$ also take $\tilde{\mathcal{C}}_j$ as the best chain and then send $\mathcal{C}_i$ to $\mathcal{S}_m$. Now, $\mathcal{S}_m$ will take $\tilde{\mathcal{C}}_j$ as the best chain.

If $\tilde{\mathcal{C}}_i[1,\ell] \not\preceq \tilde{\mathcal{C}}_j$ or $\tilde{\mathcal{C}}_j[1,\ell] \not\preceq \tilde{\mathcal{C}}_i$ where $\ell = \mathsf{len}(\tilde{\mathcal{C}}_i) - \Theta(\kappa)$ then the common prefix property of $\mathcal{S}_i$ and $\mathcal{S}_m$ will be broken. From the Theorem 6 we have that the probability $\tilde{\mathcal{C}}_i[1,\ell] \preceq \tilde{\mathcal{C}}_j$ and $\tilde{\mathcal{C}}_j[1,\ell] \preceq \tilde{\mathcal{C}}_i$ where $\ell = \mathsf{len}(\tilde{\mathcal{C}}_i) - \Theta(\kappa)$ is at least $1 - e^{-\Omega(\kappa)}$.

□

CHAPTER 5

PRACTICAL POW/POS SYSTEM: TWINSCOIN

In this chapter, we design TwinsCoin—a practical improvement of the 2-hop blockchain. This is the first cryptocurrency based on a *provably secure* and *scalable* public blockchain design using both proof-of-work and proof-of-stake mechanisms from 2-hop blockchain. 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.

5.1 From 2-hop blockchain to TwinsCoin

Our 2-hop blockchain in Chapter 4 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 PoW-miners, a new type of players, PoS-holders (stakeholders) are introduced. A winning PoW-miner cannot extend the blockchain immediately as in Bitcoin. Instead, the winning PoW-miner provides a base which enables a PoS-holder to be "selected" to extend the blockchain. That is, a PoW-miner and then a PoS-holder jointly extend the blockchain with a new block. Note that, in Bitcoin, winning PoW-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, 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.

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 gen-

erated 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 [DFZ16], 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*.

Light client design. 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.

5.2 Twinscoin design

Starting with a provably secure "core", the 2-hop blockchain [DFZ16], we develop our TwinsCoin system by following the steps below.

– First, we redesign the structure of the 2-hop blockchain in [DFZ16] with the goal of enabling a new mechanism for adjusting difficulties. We call this a modified 2-hop blockchain. Please see Section 5.2.1.

– 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 5.2.2.
– 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 5.2.3 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 5.2.4.

5.2.1 Our modified 2-hop blockchain

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.
$\tilde{B}, \tilde{\mathcal{C}}$	a PoS-block, a PoS-chain
$\mathcal{B}$	a set of proof-of-work blocks
$\langle \mathcal{C}, \tilde{\mathcal{C}} \rangle$	a chain-pair
$\mathbb{C}$	a set of chain-pairs.
X	information stored in blockchain.
Σ	unique digital signature scheme $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$.
Σ'	digital signature scheme $\Sigma' = (\mathsf{Gen}', \mathsf{Sign}', \mathsf{Verify}')$.
$(\mathsf{sk}, \mathsf{vk})$	a unique signing and verification key-pair
$(\mathsf{sk}', \mathsf{vk}')$	an ordinary signing and verification key-pair
$\textsc{Broadcast}(\cdot)$	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. Now we present the behaviors of PoW-miners and PoS-holders in our system.

Miners. Initially, each PoW-miner sets the set of chain-pairs $\mathbb{C}$ to empty. This procedure is formally presented by Algorithm 1.

Algorithm 1: Main Protocol: PoW-miner.

```
1 ℂ ← ϵ;
2 while True do
3     ℂ ← all chain-pairs received from the network.
4     ⟨𝒞, 𝒞̃⟩ ← BestValid(ℂ); 𝒞_new, ← PoW(⟨𝒞, 𝒞̃⟩)
5     if 𝒞 ≠ 𝒞_new then
6         𝒞 ← 𝒞_new; ℂ ← ℂ ∪ ⟨𝒞, 𝒞̃⟩; Broadcast(⟨𝒞, 𝒞̃⟩)
```

Stakeholders. The initialization of every PoS-holder is the same as PoW-miners except that each valid PoS-holder is parameterized by a unique pair of signing and verification keys $(\mathsf{sk}, \mathsf{vk})$, a ordinary key-pair $(\mathsf{sk}', \mathsf{vk}')$, and amount of coins v. (In this appendix, the blockchain protocol is described in the flat model, and we assume $v = 1$.) This procedure is formally presented by Algorithm 2.

5.2.1.1 Proof-of-Work Blockchain

Attempting and Successful Blocks 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 13.

Algorithm 2: Main Protocol: PoS-holder. The PoS-holder algorithm is parameterized by two key-pairs $(\mathsf{sk}, \mathsf{vk})$ for the unique signature scheme $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ and $(\mathsf{sk}', \mathsf{vk}')$ for the ordinary signature scheme $\Sigma' = (\mathsf{Gen}', \mathsf{Sign}', \mathsf{Verify}')$.

$\mathbb{C} \leftarrow \epsilon$;

while True **do**

  $\mathbb{C} \leftarrow$ all chain-pairs received from the network.

  $X \leftarrow$ all payloads received from the network.

  $\langle \mathcal{C}, \tilde{\mathcal{C}} \rangle \leftarrow \mathsf{BestValid}(\mathbb{C}); \tilde{\mathcal{C}}_{\text{new}} \leftarrow \mathsf{PoS}(\mathsf{sk}, \mathsf{vk}, \mathsf{sk}', \mathsf{vk}', X, \langle \mathcal{C}, \tilde{\mathcal{C}} \rangle)$

  if $\tilde{\mathcal{C}} \neq \tilde{\mathcal{C}}_{\text{new}}$ **then**

    $\tilde{\mathcal{C}} \leftarrow \tilde{\mathcal{C}}_{\text{new}}; \mathbb{C} \leftarrow \mathbb{C} \cup \langle \mathcal{C}, \tilde{\mathcal{C}} \rangle;$ $\textsc{Broadcast}(\langle \mathcal{C}, \tilde{\mathcal{C}} \rangle)$

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 $\langle \mathit{ctr}, h_{\mathsf{w}}, h_{\mathsf{s}}, h_{\mathsf{a}}, \omega \rangle$, where

- Notation ctr denotes a positive integer, $1 \leq \mathit{ctr} \leq \mathsf{q}$, 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 $h_{\mathsf{w}} \in \{0,1\}^{\kappa}$,
- Notation h_{s} denotes the digest of the previous proof-of-stake block, where $h_{\mathsf{s}} \in \{0,1\}^{\kappa}$,
- Notation h_{a} denotes the digest of the latest seen attempting proof-of-work block,

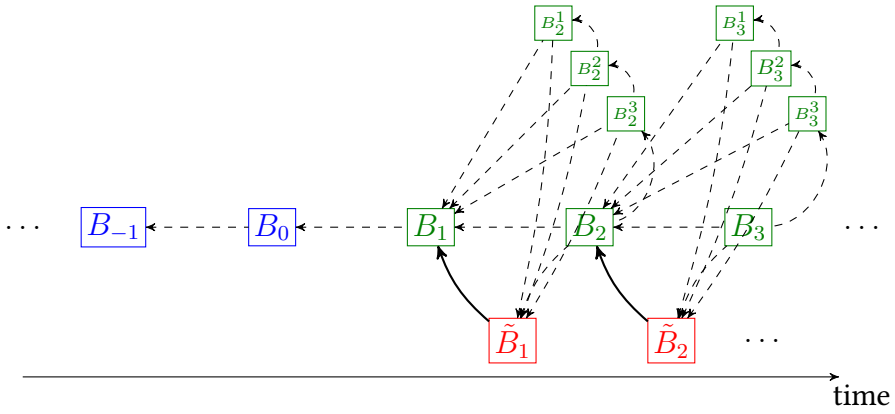

Fig. 13.: A modified 2-hop blockchain structure

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".

where $h_{\mathsf{a}} \in \{0,1\}^{\kappa}$

– Notation ω is a random nonce, where $\omega \in \{0,1\}^{\kappa}$, and block B satisfies the inequality $\mathsf{H}(ctr, \mathsf{G}(h_{\mathsf{w}}, h_{\mathsf{s}}, h_{\mathsf{a}}, \omega)) < \mathtt{T}$ where the parameters $\mathtt{T} \in \mathbb{N}$ 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 be multiple attempting blocks attaching to a successful block in our proof-of-work chain. If a chain $\mathcal{C}$ is a prefix of another chain $\mathcal{C}'$, we write $\mathcal{C} \preceq \mathcal{C}'$.

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 $\mathsf{q}, \mathtt{T}$ where q is the maximum number of random oracle queries of each PoW-miner per unit of time, and $\mathtt{T}$ determines the "current block target" of the PoW. The algorithm works as follows: given a chain-pair $\langle \mathcal{C}, \tilde{\mathcal{C}} \rangle$, 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 ω of length κ, then it increments *ctr* and checks if $\mathsf{H}(\mathit{ctr}, \mathsf{G}(h_{\mathsf{w}}, h_{\mathsf{s}}, h_{\mathsf{a}}, \omega)) < \mathtt{T}$; if a suitable $(\mathit{ctr}, h_{\mathsf{w}}, h_{\mathsf{s}}, h_{\mathsf{a}}, \omega)$ is found then the algorithm succeeds in solving the PoW and extends blockchain $\mathcal{C}$ by one block.

We illustrate the high level idea as in Figure 14. In this figure, we consider a PoW-miner attempting to mine from a valid block-pair consisting of a PoW-block B and a PoS-block $\tilde{B}$. 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 $\tilde{B}$ (by the hash h_{s} of $\tilde{B}$); however, the corresponding PoS-block of B_2^1 is not seen by the PoW-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. In the third attempt, a new successful PoW-block with the corresponding PoS-block is attached to the chain.

5.2.1.2 Proof-of-Stake Blockchain

In our system, a digital signature scheme is used by stakeholders to create new valid PoS-blocks. Let $\mathsf{H}(\cdot)$ 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. Consider each PoS player holds two pairs of keys $(\mathsf{vk}, \mathsf{sk})$ and $(\mathsf{vk}', \mathsf{sk}')$ for digital signature schemes

Algorithm 3: The *proof-of-work* function, parameterized by positive integers $\mathsf{q}, \mathtt{T}$ and hash functions $\mathsf{H}(\cdot)$, $\mathsf{G}(\cdot)$.)

function $\mathsf{PoW}(\langle \mathcal{C}, \tilde{\mathcal{C}} \rangle)$

  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 $h_{\mathsf{a}} := \perp$ if $\mathcal{B}$ is empty;$h_{\mathsf{w}} := \mathsf{H}(\mathsf{head}(\mathcal{C}))$ $h_{\mathsf{s}} := \mathsf{H}(\mathsf{head}(\tilde{\mathcal{C}}))$; $ctr \leftarrow 1$; $\omega \leftarrow \{0,1\}^{\kappa}$; $B \leftarrow \epsilon$

  if $l \neq 0$ **then**

    Let B^l be the latest PoW-block found $\mathcal{B}$ that attaches to $\mathsf{head}(\mathcal{C})$; $h_{\mathsf{a}} \leftarrow \mathsf{H}(B^l)$

  while $ctr \leq q$ **do**

    if $(\mathsf{H}(ctr, \mathsf{G}(h_{\mathsf{w}}, h_{\mathsf{s}}, h_{\mathsf{a}}, \omega)) < \mathtt{T}) \wedge (ctr \leq q)$ **then**

      $B \leftarrow \langle ctr, h_{\mathsf{w}}, h_{\mathsf{s}}, h_{\mathsf{a}}, \omega \rangle$; $\mathcal{C} \leftarrow \mathcal{C}B$

    $ctr \leftarrow ctr + 1$

  return $\mathcal{C}$; /* Return the updated chain */

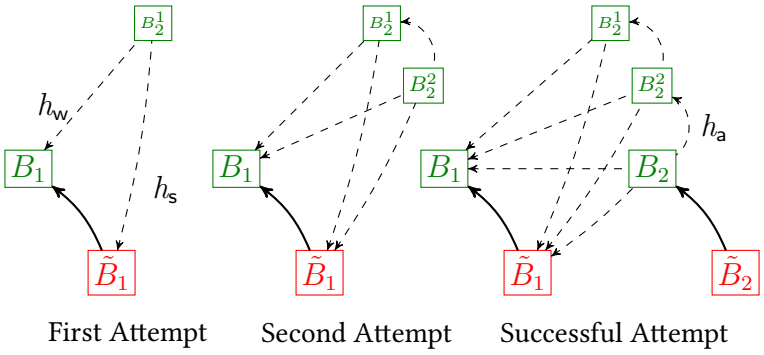

Fig. 14.: Generating new PoW-blocks

$(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ and $(\mathsf{Gen}', \mathsf{Sign}', \mathsf{Verify}')$, respectively. Note that $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ is a *unique* digital signature scheme [Lys02], while $(\mathsf{Gen}', \mathsf{Sign}', \mathsf{Verify}')$ can be an ordinary digital signature scheme.

In our system, each valid PoS-block is coupled with a valid PoW-block. Based on a given PoW-block B, any stakeholder whose verification key vk satisfies the following hash inequality $\tilde{\mathsf{H}}(h_{\mathsf{w}}, \tilde{\omega}, \mathsf{vk}) < \tilde{\mathsf{T}}$ is allowed to generate a new PoS block, where $\tilde{\mathsf{T}}$ is the current proof-of-stake target, h_{w} is the hash value of B and $\tilde{\omega} \leftarrow \mathsf{Sign}_{\mathsf{sk}}(B)$.

Then a new PoS-block $\tilde{B}$ is in the form $\tilde{B} = \langle (B, \tilde{\omega}, \mathsf{vk}), X, \sigma, \mathsf{vk}' \rangle$, where $X \in \{0,1\}^*$ is the payload of the proof-of-stake block $\tilde{B}$ (also denoted as $\mathsf{payload}(\tilde{B})$); and σ is produced by the PoS player but by using a different signing key sk', i.e., $\sigma \leftarrow \mathsf{Sign}'_{\mathsf{sk}'}((B, \tilde{\omega}, \mathsf{vk}), X)$.

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

Extending a Proof-of-Stake Blockchain. In Algorithm 4, we describe how the stakeholders extend PoS-chains. In more details, Algorithm 4 processes as follows. Upon re-

ceiving message $(\mathsf{sk}, \mathsf{vk}, \mathsf{sk}', \mathsf{vk}', X, \langle \mathcal{C}, \tilde{\mathcal{C}} \rangle)$, the algorithm attempts to extend the specified PoS-chain $\tilde{\mathcal{C}}$ in the chain-pair $\langle \mathcal{C}, \tilde{\mathcal{C}} \rangle$; The algorithm then executes the following 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 $\mathsf{H}(h_{\mathsf{w}}, \tilde{\omega}, \mathsf{vk}) < \tilde{\mathsf{T}}$ holds, where $\tilde{\omega} \leftarrow \mathsf{Sign}_{\mathsf{sk}}(B)$ and $h_{\mathsf{w}} := \mathsf{H}(B)$. If yes, the PoS-holder with the key-pair $(\mathsf{sk}, \mathsf{vk})$ is the winning PoS-holder.

Step 2—Signature generation. After Step 1 the PoS-holder with signing and verification keys $(\mathsf{sk}, \mathsf{vk})$ is the winning PoS-holder. The algorithm then generates a signature $\sigma \leftarrow \mathsf{Sign}'_{\mathsf{sk}'}((B, \tilde{\omega}, \mathsf{vk}), X)$ using the ordinary signature scheme $\Sigma'$with key-pair $(\mathsf{sk}', \mathsf{vk}')$, and forms a new PoS-block $\tilde{B} = \langle (B, \tilde{\omega}, \mathsf{vk}), X, \sigma, \mathsf{vk}' \rangle$. We say the PoS-holder with the key pair $(\mathsf{sk}, \mathsf{vk})$ extends the specified PoS-chain $\tilde{\mathcal{C}}$ with $\tilde{B}$.

The Figure 15 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.

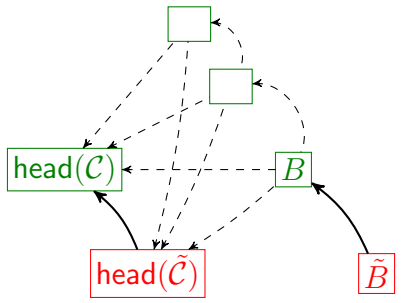

Fig. 15.: Generating new PoS-blocks

Algorithm 4: The *proof-of-stake* function, parameterized by a unique signature scheme $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$, an ordinary signature scheme $\Sigma' = (\mathsf{Gen}', \mathsf{Sign}', \mathsf{Verify}')$, a parameter $\tilde{\mathsf{T}}$, a function $\mathsf{H}(\cdot)$.

function $\mathsf{PoS}(\mathsf{sk}, \mathsf{vk}, \mathsf{sk}', \mathsf{vk}', X, \langle \mathcal{C}, \tilde{\mathcal{C}} \rangle)$

  Let $\mathcal{B}$ is the set of all attempting blocks that attach to $\mathsf{head}(\mathcal{C})$

  Let l be the number of attempting blocks in $\mathcal{B}$

  Let B be the latest PoW-block in $\mathcal{B}$ that attaches to $\mathsf{head}(\mathcal{C})$

  /* If there is one new PoW-block that attaches to $\mathsf{head}(\mathcal{C})$. */

  $\tilde{\omega} := \mathsf{Sign}_{\mathsf{sk}}(B)$

  $h_{\mathsf{w}} := \mathsf{H}(B)$

  if $l > 0$ **then**

    if $(\mathsf{H}(h_{\mathsf{w}}, \tilde{\omega}, \mathsf{vk}) < \tilde{\mathsf{T}}))$ **then**

      $\sigma \leftarrow \mathsf{Sign}'_{\mathsf{sk}'}((B, \tilde{\omega}, \mathsf{vk}), X)$

      $\tilde{B} = \langle (B, \tilde{\omega}, \mathsf{vk}), X, \sigma, \mathsf{vk}' \rangle; \tilde{\mathcal{C}} \leftarrow \tilde{\mathcal{C}}\tilde{B}$

  return $\tilde{\mathcal{C}}$; /* Return the updated chain */

5.2.1.3 Validating a Chain-pair

Chain-Pair validation is the most important process in our design. We first formally describe the following predicates used in the ValidateChain algorithm.

Predicate $\mathsf{ValidPoW}^{\mathsf{q},\mathsf{T}}_{\mathsf{H},\mathsf{G}}(B, B', \tilde{B}, \mathcal{B}, \hat{\mathcal{B}})$**.** 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 $\tilde{B}$, two sets of attempting PoW-blocks $\mathcal{B}$ and $\hat{\mathcal{B}}$. Here, the block B consists of $\langle ctr, h_{\mathsf{w}}, h_{\mathsf{s}}, h_{\mathsf{a}}, \omega \rangle$, block B' consists of $\langle ctr', h'_{\mathsf{w}}, h'_{\mathsf{s}}, h'_{\mathsf{a}}, \omega' \rangle$, the set $\mathcal{B}$ consists of l attempting blocks $\{B^1, \ldots, B^l\}$ where each block $B^i = \langle ctr^i, h^i_{\mathsf{w}}, h^i_{\mathsf{s}}, h^i_{\mathsf{a}}, \omega^i \rangle$

for $1 \leq i \leq l$, and the set $\hat{\mathcal{B}}$ consists of l attempting blocks $\{\hat{B}^1, \ldots, \hat{B}^l\}$ where each block $\hat{B}^j = \langle \hat{ctr}^j, \hat{h}_{\mathsf{w}}^j, \hat{h}_{\mathsf{s}}^j, \hat{h}_{\mathsf{b}}^j, \hat{\omega}^j \rangle$ for $1 \leq j \leq l$. 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 $\mathsf{H}(ctr, \mathsf{G}(h_{\mathsf{w}}, h_{\mathsf{s}}, h_{\mathsf{a}}, \omega)) < \mathsf{T}$; B links to the previous PoW-block B' if $h_{\mathsf{w}} = \mathsf{H}(B')$;B links to the previous PoS-block $\tilde{B}$ if $h_{\mathsf{s}} = \mathsf{H}(\tilde{B})$.
- If $l > 0$, check whether B links to the latest attempting block in the second set $\hat{\mathcal{B}}$ if $h_{\mathsf{a}} = \mathsf{H}(\hat{B}^l)$.
- If $l > 0$, check whether all attempting blocks in $\mathcal{B}$ are properly solved if for $1 \leq i \leq l$, $\mathsf{H}(ctr^i, \mathsf{G}(h_{\mathsf{w}}^i, h_{\mathsf{s}}^i, h_{\mathsf{a}}^i, \omega^i)) < \mathsf{T}$
- If $l > 0$, check whether all attempting blocks $B^i = \langle ctr^i, h_{\mathsf{w}}^i, h_{\mathsf{s}}^i, h_{\mathsf{a}}^i, \omega^i \rangle$ in $\hat{\mathcal{B}}$ are properly linked if
  - B^i links to the previous PoW-block B' if $h_{\mathsf{w}}^i = \mathsf{H}(B')$.
  - B^i links to the previous PoS-block $\tilde{B}$ if $h_{\mathsf{s}}^i = \mathsf{H}(\tilde{B})$.
  - B^i links to the previous attempting block in the set $\mathcal{B}$ if $h_{\mathsf{a}}^i = \mathsf{H}(B^{i-1})$ (Note that, we consider $B^0 = B'$.)

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

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

The predicate checks the following

- $\tilde{B}$ is generated by an elected PoS-holder if $\mathsf{H}(h_{\mathsf{w}}, \tilde{\omega}, \mathsf{vk}) < \tilde{\mathsf{T}}$, where $h_{\mathsf{w}} = \mathsf{H}(B)$
- $\tilde{B}$ links to the corresponding PoW-block B if $B = B'$
- The signature σ is properly generated if $\mathsf{Verify}'_{\mathsf{vk}'}((B, \tilde{\omega}, \mathsf{vk}), X, \sigma) = 1$

Algorithm 5: The *chain-pair validation algorithm*, parameterized by a unique signature scheme $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$, an ordinary signature scheme $\Sigma' = (\mathsf{Gen}', \mathsf{Sign}', \mathsf{Verify};)$, the stake-identity set S, parameters q, $\mathtt{T}$, $\tilde{\mathtt{T}}$, an initial chain $\mathcal{C}_{\mathsf{init}}$, the hash functions $\mathsf{H}(\cdot)$, $\mathsf{H}(\cdot)$,$\mathsf{G}(\cdot)$ and the content validation predicate $V(\cdot)$.

function ValidateChain($\langle \mathcal{C}, \tilde{\mathcal{C}} \rangle$)
  $b \leftarrow V(\mathsf{payload}(\tilde{\mathcal{C}}))$
  if b = *True* **then**
    repeat
      $B \leftarrow \mathsf{H}(\mathsf{head}(\mathcal{C}))$; $\tilde{B} \leftarrow \mathsf{H}(\mathsf{head}(\tilde{\mathcal{C}}))$
      Let $\mathcal{B}$ is the set of all attempting PoW-blocks attaching to $\mathsf{head}(\mathcal{C})$
      Truncate all PoW-blocks from the head of $\mathcal{C}$ (including the head), and truncate the head of $\tilde{\mathcal{C}}$
      /* obtain new heads */
      Let $\hat{\mathcal{B}}$ is the set of all attempting PoW-blocks attaching to the $\mathsf{newhead}(\mathcal{C})$
      $b_1 \leftarrow \mathsf{ValidPoW}^{\mathsf{q},\mathtt{T}}_{\mathsf{H},\mathsf{G}}(B, \mathsf{head}(\mathcal{C}), \mathsf{head}(\tilde{\mathcal{C}}), \mathcal{B}, \hat{\mathcal{B}})$
      $b_2 \leftarrow \mathsf{ValidPoS}^{\Sigma,\Sigma',\tilde{\mathtt{T}}}_{\mathsf{H}}(\tilde{B}, B)$; $b = b_1 \wedge b_2$
    until $(\mathcal{C} = \mathcal{C}_{\mathsf{init}}) \vee (b =$ *False*$)$;
  return b

The predicate $\mathsf{ValidPoW}^{\Sigma,\Sigma',\tilde{\mathtt{T}}}_{\mathsf{H}}$ outputs 1 if and only if the examined PoS-block $\tilde{B}$ passes all tests described above.

Our chain-pair validation algorithm, denoted ValidateChain, 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 $\tilde{\mathcal{C}}$ 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. Please refer to Algorithm 5 for more details.

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.

5.2.2 Blockchain with adjustable difficulty

Nakamoto's difficulty adjustment for PoW. In Bitcoin, in order to keep the block 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 target adjustment scheme. For some time interval, if the chain extension rate is higher than expected, the target needs to be decreased to make the successful probability lower.

The target is adjusted every m blocks, In Bitcoin, $\mathsf{m} = 2016$. 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 target in the i-th epoch. We have the target in the $(i+1)$-th

epoch as follows:

$$T_{i+1} = \frac{t_i}{\mathsf{t}} T_i \tag{5.1}$$

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 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.

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

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

$$\tilde{\mathsf{T}}_{i+1} = \frac{m_i \mathsf{V}}{\mathsf{m}} \tilde{\mathsf{T}}_i \tag{5.2}$$

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

that the PoW-block is mapped to a stakeholder is higher than the expectation V; therefore, the PoS target $\tilde{\mathsf{T}}_{i+1}$ will be decreased. It is easy to see this is a negative feedback algorithm that can ensure that the successful probability is close to V.

PoW 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 5.2.1, in order to generate a valid block a miner need to try different ω to satisfy a hash inequality, i.e., $\mathsf{H}(h_{\mathsf{w}}, h_{\mathsf{s}}, h_{\mathsf{a}}, \omega) < \mathsf{T}$. 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 $\mathsf{V} < 1$. 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 target in the i-th epoch. The target in the $(i+1)$-th epoch is defined by

$$\mathsf{T}_{i+1} = \frac{\mathsf{m}t_i}{m_i \mathsf{t} \mathsf{V}} \mathsf{T}_i \tag{5.3}$$

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 decrease T_{i+1} to increase the PoW difficulty in the $(i+1)$-th epoch. Similarly, if $t_i > \mathsf{t}$, we need to increase T_{i+1} to decrease the PoW difficulty.
- If $m_i > \mathsf{m}/\mathsf{V}$, we know that the probability that PoW-block can be mapped to a stake-

holder is lower than the expectation V. 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}). Similarly, if $m_i < \mathsf{m}/\mathsf{V}$, the PoS difficulty would be increased, then we need to decrease PoW difficulty (by increasing T_{i+1}).

Potential attack on adjustable difficulty. Garay et al. [GKL15] and then Pass et al. [PSS17b] analyze blockchain without considering difficulty adjustment. The difficulty D is fixed and reflect the probability p that a hash query is a successful query. Together with the number of players n, p will control the probability that a player finds a valid solution in a round. Furthermore, they also assume the ratio of honest players ρ which provides an upper bound on the probability that a round is a successful round. Notation α is used to denote the probability that some honest player succeeds in solving a puzzle in one round and β is used to denote the expected number of blocks that an attacker can mine in a round. they assume $\alpha \ll 1$ and $\beta \ll 1$ to analyze the protocol.

Garay et al. [GKL17] discuss backbone protocol with variable difficulty. They assume that in any consecutive s rounds the mining power will increase γ times at most. They study the security properties under this assumption. A potential attack on adjustable difficulty is that malicious parties may stop to mine new blocks in some rounds. According to the algorithm, the target D will be decreased to make the successful query easier so that the system will keep a consistent pace to generate new blocks. The malicious players then begin to mine new blocks under this difficult target. If the assumption $\alpha \ll 1$ and $\beta \ll 1$ is broken, then the system would be insecure.

Security analysis on Nakamoto's. We provide an security analysis to demonstrate that the malicious players are not able to attack the Nakamoto's system by taking advantage of the difficulty adjustment mechanism. Our analysis uses the original security proof in Garay et al. [GKL15] as a black box. They have proven the backbone protocol will achieve

chain quality and common prefix properties based on some reasonable assumptions.

We will follow the same assumption and notations as in [GKL15]. Parameter p denotes a probability of that a hash query satisfy the target. The malicious players may change p by effecting the adjustable difficulty. We will prove the attack will not break the assumption in [GKL15]. We use the number of rounds to measure the duration time t instead of the actual time. We also use t_i to denote the number of rounds for the i-th epoch.

Lemma 17. *Let n be the total number of players and ρ be the ratio of honest players. In each round, a player can make q hash queries. Let m be the number of blocks for an epoch and t be the expected rounds for an epoch. Let p be the probability that a hash query satisfies the target. For any constant $\delta > 1$, we have $\Pr[p \leq (1+\delta)\frac{\mathsf{m}}{nq\rho\mathsf{t}}] > 1 - e^{-\Omega(\delta)}$.*

Proof. First, we discuss the average case. If the malicious players stop to contribute any blocks, the epoch would become longer than the expectation. From the equation 5.1, it is easy to see that the target T would be increased, as well as the parameter p. After some time, this would reach to a stable status in the i-th epoch, that is $T_{i+1} = T_i$. We get $t_i = \mathsf{t}$. In the i-th epoch, only honest computing power tries to generate new blocks. On average, the honest miners will generate $nq\rho t_i p = \mathsf{m}$ blocks in the i-th epoch. For $t_i = \mathsf{t}$, we get $nq\rho\mathsf{t}p = \mathsf{m}$. That is $p = \frac{\mathsf{m}}{nq\rho\mathsf{t}}$. Now, we turn to the worst case. When $p \geq \frac{\mathsf{m}}{nq\rho\mathsf{t}}$, suppose that in t rounds the honest miners generate m blocks. By Chernoff bound, we have $Pr[t > (1+\delta)\mathsf{t}] < e^{-\Omega(\delta)}$. From the equation 5.1, we have $Pr[T_{i+1} > (1+\delta)T_i] < e^{-\Omega(\delta)}$. That is $\Pr[p > (1+\delta)\frac{\mathsf{m}}{nq\rho\mathsf{t}}] < e^{-\Omega(\delta)}$. We get $\Pr[p \leq (1+\delta)\frac{\mathsf{m}}{nq\rho\mathsf{t}}] > 1 - e^{-\Omega(\delta)}$. □

Lemma 18. *Let n be the total number of players and ρ be the ratio of honest players. In each round, a player can make q hash queries. Let m be the number of blocks for an epoch and t be the expected rounds for an epoch. If $\frac{\mathsf{m}}{\mathsf{t}} \ll 1$, we have $\alpha \ll 1$ and $\beta \ll 1$ with probability that is no less than $1 - e^{-\Omega(\delta)}$, for any $\delta > 1$.*

Proof. From the definition, we have $\alpha \approx n\rho qp$. From lemma 17, we have $\alpha \leq (1+\delta)\frac{\mathsf{m}}{\mathsf{t}} \ll 1$ with probability that is greater than or equal to $1 - e^{-\Omega(\delta)}$. We also have $\beta \approx n(1-\rho)qp$. From lemma 17, we have $\beta \leq (1+\delta)\frac{\mathsf{m}(1-\rho)}{\mathsf{t}\rho} \ll 1$ with probability that is no less than $1 - e^{-\Omega(\delta)}$. □

We therefore conclude that the malicious players cannot break the assumption by tuning the target.

Security analysis on TwinsCoin. We here provide 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 [DFZ16], 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. They also assume the probability that a PoW block is mapped to a valid stake is much less than 1. A potential attack is that the malicious players can increase the target to make it is easier to generate a PoW block, or increase the probability that a PoW block is mapped to a stake.. In this section we prove that this attack does not work for our modified 2-hop blockchain under a certain assumption.

The intuition is that in order to increase the target, the malicious players can stop to contribute new PoW-blocks and PoS-blocks from some moments. This will make the block extension rate lower. By our difficulty adjustment algorithm, the target will increase to speed up the block generation. At some later point, the malicious players will begin to work and sign under the increased target. We will follow the notations in [DFZ16]. 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. We also assume the total amount of coins is $\hat{n}$ and the honest ratio is $\hat{\rho}$.

Lemma 19. *Let* V *be the expectation of the probability that a PoW-block is successfully*

mapped to a stakeholder. Total amount of coins is $\hat{n}$ and the honest ratio is $\hat{\rho}$. Let $\tilde{\alpha}$ and $\tilde{\beta}$ be the probabilities that a PoW-block is mapped to an honest stake and a malicious stake, respectively. For for any $\delta \in (0,1)$, we have $\tilde{\alpha} \leq \frac{\mathsf{V}}{1-\delta}$ and $\tilde{\beta} \leq \frac{1-\tilde{\rho}}{\tilde{\rho}} \frac{\mathsf{V}}{1-\delta}$ with probability that is no less than $1 - e^{-\Omega(\delta \frac{\mathsf{m}}{\mathsf{V}})}$.

Proof. First, we discuss the average case. If the malicious players stop sign PoS blocks, the PoS target $\tilde{T}$ would be increased. When the honest players can sign PoS block with the expectation probability V, the PoS target will be stable. That is $\tilde{T}_{i+1} = \tilde{T}_i$. From the equation 5.2, we have $\frac{\mathsf{m}}{m_i} = \mathsf{V}$. That is $m_i = \frac{\mathsf{m}}{\mathsf{V}}$. Now, we turn to the worst case. When the malicious players stop sign PoS blocks, we have $\tilde{\alpha} = \frac{\mathsf{m}}{m_i} = \mathsf{V}$. with Chernoff bound, for any $\delta \in (0,1)$ we have $\Pr[m_i < (1-\delta)\frac{\mathsf{m}}{\mathsf{V}}] < e^{-\Omega(\delta \frac{\mathsf{m}}{\mathsf{V}})}$. That is $\Pr[\tilde{\alpha} = \frac{\mathsf{m}}{m_i} > \frac{\mathsf{V}}{1-\delta}] < e^{-\Omega(\delta \frac{\mathsf{m}}{\mathsf{V}})}$. The malicious will take the same PoS target, we have $\Pr[\tilde{\beta} > \frac{1-\tilde{\rho}}{\tilde{\rho}} \frac{\mathsf{V}}{1-\delta}] < e^{-\Omega(\delta \frac{\mathsf{m}}{\mathsf{V}})}$. □

From the Lemma 19, $\tilde{\alpha}$ and $\tilde{\beta}$ will not break the assumption by difficulty adjustment with high probability.

Lemma 20. *Let V be the expectation of the probability that a PoW-block is successfully mapped to a stakeholder. Total amount of coins is $\hat{n}$ and the honest ratio is $\hat{\rho}$ the malicious ratio is $1 - \hat{\rho}$. Let n be the total number of players and ρ be the ratio of honest players. In each round, a player can make q hash queries. Let m be the number of blocks for an epoch and t be the expected rounds for an epoch. If $\frac{\mathsf{m}}{\mathsf{t}} \ll 1$, we have $\alpha \ll 1$ and $\beta \ll 1$ with probability that is no less than $1 - e^{-\Omega(\delta)}$.*

Proof. From Lemma 19, we have $\tilde{\alpha} \leq \mathsf{V}$ with probability that is greater than or equal to $1 - e^{-\Omega(\frac{\mathsf{m}}{\mathsf{V}})}$. Putting it together with the equation 5.3, we have $\mathsf{T}_{i+1} \leq \frac{t_i}{\mathsf{t}} \mathsf{T}_i$ with probability that is no less than $1 - e^{-\Omega(\frac{\mathsf{m}}{\mathsf{V}})}$. This means the target T_i will be increased no faster than it is in equation 5.1. We can show the proof in the similar way as in Lemma 18. □

From the Lemma 20, α and β will not break the assumption by difficulty adjustment with a high probability.

5.2.3 PoS blockchain in the non-flat model

Moving PoS blockchain from the flat to the non-flat model. The (modified) 2-hop blockchain in section 5.2.1 is described in the flat model with a pre-fixed difficulty parameter. Intuitively, if PoS-holders 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 PoS-holder can keep different amount of stake in an account. Next, we prove that our construction is fair in the non-flat model so a PoS-holder would 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 $\langle \mathcal{C}, \tilde{\mathcal{C}} \rangle$ be the best chain-pair, and $\mathcal{C}$ is PoW-chain, $\tilde{\mathcal{C}}$ 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 would attempt to generate a new PoS-block as the following: Let $\tilde{\mathsf{T}}$ 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 $\mathsf{H}(\cdot)$ is κ. Let $p = \frac{\tilde{\mathsf{T}}}{2^\kappa}$, we assume $\hat{n}p < 1$. Let v be the number of coins in the account of a stakeholder with vk. If the inequality $\tilde{\mathsf{H}}(h_{\mathsf{w}}, \tilde{\omega}, \mathsf{vk}) < \tilde{\mathsf{T}}$ is satisfied, the stakeholder with the key-pair $(\mathsf{sk}, \mathsf{vk})$ would win 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.

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 would not change the probability for PoS-block mapping. Intuitively, from our non-flat construction, if a PoS-holder 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 21. *Let $p = \frac{\tilde{\mathsf{T}}}{2^\kappa}$ 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 $\Pr[\tilde{\mathsf{H}}(h_{\mathsf{w}}, \tilde{\omega}, \mathsf{vk}) < v\tilde{\mathsf{T}}] = vp$.*

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

From the Lemma 21, 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 PoS-holder 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 PoS-holder 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 $\tilde{\mathsf{H}}(h_{\mathsf{w}}, \tilde{\omega}, \mathsf{vk})$ are independent for different unque pair $(\mathsf{vk}, \tilde{\omega})$. The total probability that the PoS-holder 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 PoS-holder, the probability that he is selected only depends on the total amount of stake (coins) he controls.

We summarize our TwinsCoin design in Figure 16. Starting with the 2-hop blockchain 4, we first propose a slightly modified version (Section 5.2.1) 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 5.2.2), 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 with light client design, we complete the blockchain design in our TwinsCoin system.

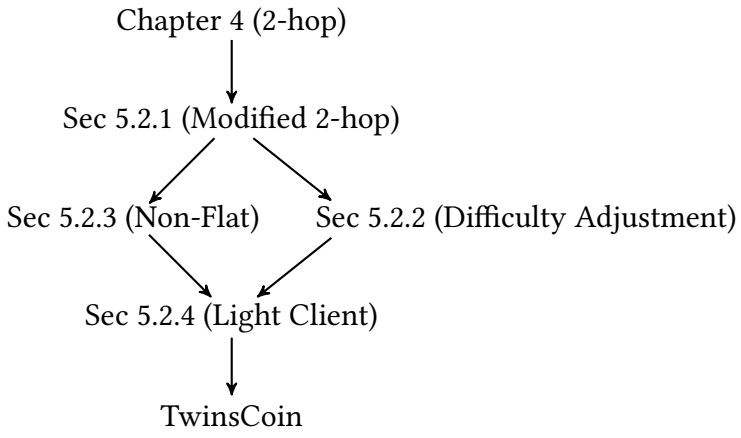

Fig. 16.: Roadmap for blockchain design in TwinsCoin.

5.2.4 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 [Blo17]), 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 [**FC:RMCI17**] 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 block-header 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 6.2.0.2. 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 6.2.0.2.

CHAPTER 6

IMPLEMENTATION AND EXPERIMENTS

We provide a full-fledged implementation of TwinsCoin. Details on the implementation are provided in Section 6.1. 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 6.2. We also run fully functional TwinsCoin nodes over a testing network which is described in Section 6.2.1.

6.1 Implementation

We implement TwinsCoin using the Scorex framework [Inp16] 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 [Inp16], Sawtooth Lake [Int16a], and Fabric [IBM16]. We choose to use Scorex 2.0 [Inp16] 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 [Goo]. 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 influ-

[1] https://bitbucket.org/TwCoin/twinscoin

enced 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[SLZ16]), 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 [Wikc]: 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 [Wikb] 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 [Gui] 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 PoW-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 [Kot]. 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 [Wika]. 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.

6.2 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 ([Nak08], Section 11). The goal of a second experiment is to measure efficiency of the light client proposal from the Section 5.2.4. Third experiment examines proof-of-stake difficulty readjustment procedure in a simulated environment.

6.2.0.1 Chain race experiment

In the original Bitcoin paper [Nak08], 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 17. 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.

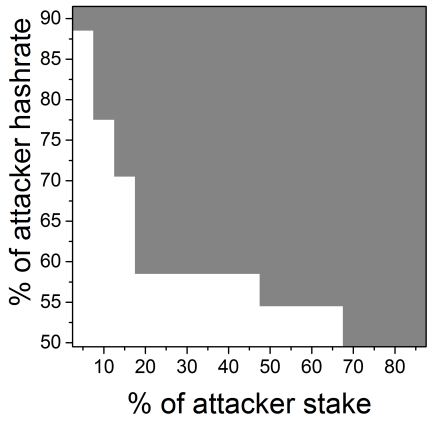

Fig. 17.: Influence of attacker's stake to his hashrate.

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 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 [Hei+15]. Nevertheless, a malicious miner needs to spend a lot of money to overtake the honest parties.

6.2.0.2 Light validation experiment

We estimate practically how efficient is the light validation procedure proposed in the Section 5.2.4. 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 [Rey+16]. 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 [Blo17] 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.

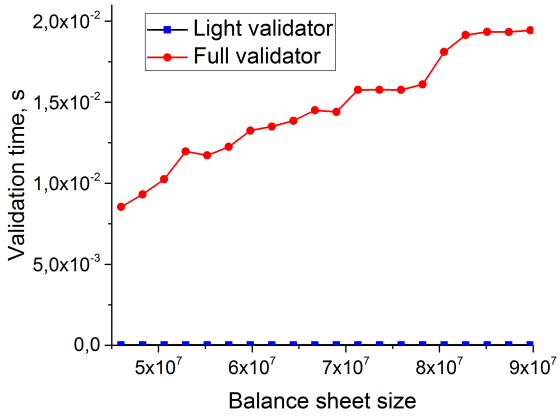

Fig. 18.: Balance lookup time.

Figure 18 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 in the Figure 19. 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.

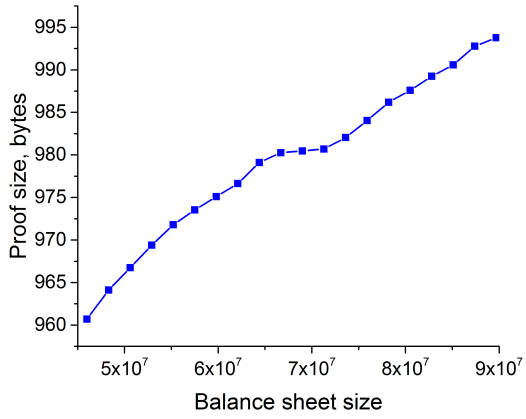

Fig. 19.: Balance lookup proof size.

6.2.0.3 Proof-of-stake difficulty experiments

In the Proof-of-Stake difficulty readjustment formula (see Formula 5.2) we use the V parameter to show the probability that a PoW-block is a successful block. In other words, V indicates the probability that a PoW-block is *successfully mapped* to a stakeholder. Also, $1 - V$ 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 20 show that in this scenario experimental numbers are close to the planned ones.

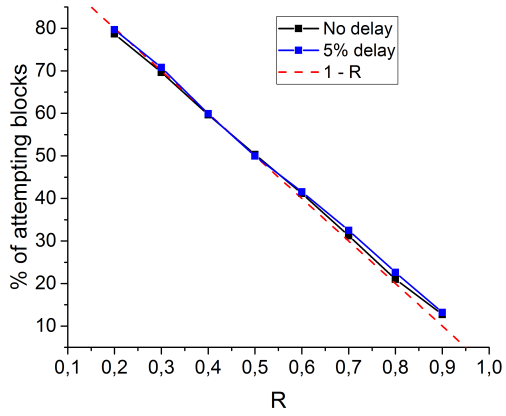

Fig. 20.: Percentage of Attempting Blocks

We found that, given a constant stake, target value is changing with number of stakeholders growing. The dependency is presented in the Figure 21.

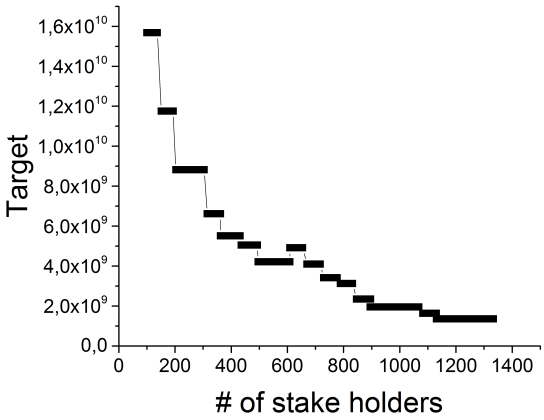

Fig. 21.: $\tilde{T}$ Value

6.2.1 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 $\mathsf{V} = 0.8$.

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).

Appendix A

ABBREVIATIONS

VCU	Virginia Commonwealth University
RVA	Richmond Virginia
PoW	Proof-of-work
PoS	Proof-of-stake

REFERENCES

[Abu+17] Hamza Abusalah et al. “Beyond Hellman’s Time-Memory Trade-Offs with Applications to Proofs of Space”. In: 2017, pp. 357–379.

[Bac02] Adam Back. “Hashcash — A Denial of Service Counter-Measure”. In: http://hashcash.org/papers/hashcash.pdf. 2002.

[Bad+17] Christian Badertscher et al. “Bitcoin as a Transaction Ledger: A Composable Treatment”. In: 2017, pp. 324–356.

[Ben+14] Iddo Bentov et al. “Proof of Activity: Extending Bitcoin’s Proof of Work via Proof of Stake [Extended Abstract]”. In: *SIGMETRICS Perform. Eval. Rev.* 42.3 (Dec. 2014), pp. 34–37. URL: http://doi.acm.org/10.1145/2695533.2695545.

[BG17] Vitalik Buterin and Virgil Griffith. “Casper the Friendly Finality Gadget”. In: https://arxiv.org/pdf/1710.09437.pdf. Nov. 2017.

[BGM16] Iddo Bentov, Ariel Gabizon, and Alex Mizrahi. “Currencies without Proof of Work”. In: *Bitcoin Workshop- Financial Cryptography and Data Security (FC).* 2016.

[Bit11] Bitcointalk. “Proof of stake instead of proof of work”. In: Online post by QuantumMechanic, available at https://bitcointalk.org/index.php?topic=27787.0. July 2011.

[Blo17] Blockchain.info. “Number of Unspent Transaction Outputs”. In: https://bitcoin.org/en/developer-guide. 2017.

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. “Short Signatures from the Weil Pairing”. In: 2001, pp. 514–532.

[BR93] Mihir Bellare and Phillip Rogaway. “Random Oracles Are Practical: A Paradigm for Designing Efficient Protocols”. In: *Proceedings of the 1st ACM Conference on Computer and Communications Security*. CCS ’93. Fairfax, Virginia, USA: ACM, 1993, pp. 62–73.

[Can00a] Ran Canetti. “Security and Composition of Multiparty Cryptographic Protocols”. In: 13.1 (2000), pp. 143–202.

[Can00b] Ran Canetti. *Universally Composable Security: A New Paradigm for Cryptographic Protocols*. Cryptology ePrint Archive, Report 2000/067. http://eprint.iacr.org/2000/067. 2000.

[Can01] Ran Canetti. “Universally Composable Security: A New Paradigm for Cryptographic Protocols”. In: 2001, pp. 136–145.

[Can03] Ran Canetti. *Universally Composable Signatures, Certification and Authentication*. Cryptology ePrint Archive, Report 2003/239. http://eprint.iacr.org/2003/239. 2003.

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. “The Random Oracle Methodology, Revisited (Preliminary Version)”. In: 1998, pp. 209–218.

[Cha82] David Chaum. “Blind Signatures for Untraceable Payments”. In: 1982, pp. 199–203.

[CM17] Jing Chen and Silvio Micali. “Algorand”. In: *arXiv:1607.01341*. http://arxiv.org/abs/1607.01341. May 2017.

[Coh+] Jonah Brown Cohen et al. “Formal Barriers to Proof-of-Stake Protocols”. In: Video link https://www.youtube.com/watch?v=PGrWGMRbdvw.

[Cry14] CryptoManiac. “Proof of Stake”. In: *NovaCoin wiki* (2014). https://github.com/novacoin-project/novacoin/wiki/Proof-of-stake.

[Dav+17] Bernardo David et al. “Ouroboros Praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain”. In: *Cryptology ePrint Archive, Report 2017/573*. http://eprint.iacr.org/2017/573. June 2017.

[DFZ16] Tuyet Duong, Lei Fan, and Hong-Sheng Zhou. “2-hop Blockchain: Combining Proof-of-Work and Proof-of-Stake Securely”. In: *Cryptology ePrint Archive, Report 2016/716*. https://eprint.iacr.org/2016/716. 2016.

[DN93] Cynthia Dwork and Moni Naor. “Pricing via Processing or Combatting Junk Mail”. In: 1993, pp. 139–147.

[DPS17] Phil Daian, Rafael Pass, and Elaine Shi. “Snow White: Robustly Reconfigurable Consensus and Applications to Provably Secure Proofs of Stake”. In: *Cryptology ePrint Archive, Report 2016/919*. http://eprint.iacr.org/2016/919. Apr. 2017.

[ES14] Ittay Eyal and Emin Gün Sirer. “Majority Is Not Enough: Bitcoin Mining Is Vulnerable”. In: 2014, pp. 436–454. DOI: 10.1007/978-3-662-45472-5_28.

[Eya15] Ittay Eyal. “The Miner’s Dilemma”. In: 2015, pp. 89–103. DOI: 10.1109/SP.2015.13.

[FZ17] Lei Fan and Hong-Sheng Zhou. “A Scalable Proof-of-Stake Blockchain in the Open Setting (or, How to Mimic Nakamoto’s Design via Proof-of-Stake)”. In: https://eprint.iacr.org/2017/656. July 2017.

[Gil+17] Yossi Gilad et al. “Algorand: Scaling Byzantine Agreements for Cryptocurrencies”. In: *Cryptology ePrint Archive, Report 2017/454*. https://eprint.iacr.org/2017/454. May 2017.

[GKL15] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. “The Bitcoin Backbone Protocol: Analysis and Applications”. In: 2015, pp. 281–310. DOI: 10.1007/978-3-662-46803-6_10.

[GKL17] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. “The Bitcoin Backbone Protocol with Chains of Variable Difficulty”. In: 2017, pp. 291–323.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. “How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority”. In: 1987, pp. 218–229.

[Goo] L. M Goodman. “Tezos: A Self-Amending Crypto-Ledger. Position Paper”. In:

[Goo14] Dan Goodin. “Bitcoin security guarantee shattered by anonymous miner with 51% network power”. In: http://arstechnica.com/. 2014.

[Gui] Bitcoin Developer Guide. “UTXO Definition”. In: https://bitcoin.org/en/developer-guide.

[Hei+15] Ethan Heilman et al. “Eclipse Attacks on Bitcoin’s Peer-to-Peer Network.” In: *USENIX Security*. 2015, pp. 129–144.

[HM04] Dennis Hofheinz and Jörn Müller-Quade. “Universally Composable Commitments Using Random Oracles”. In: 2004, pp. 58–76.

[IBM16] IBM Corp. “Hyperledger-Fabric”. In: https://github.com/hyperledger/fabric. 2016.

[Inp16] Input Output Hong Kong. “The Scorex Project”. In: https://github.com/input-output-hk/Scorex. 2016.

[Int16a] Intel. “Hyperledger-Sawtooth Lake”. In: https://github.com/hyperledger/sawtooth-core. 2016.

[Int16b] Intel. “Proof of Elapsed Time (PoET)”. In: (2016). https://intelledger.github.io/introduction.html.

[Jep15] Christina Jepson. “DTB001: Decred Technical Brief”. In: Available at https://coss.io/documents/white-papers/decred.pdf Additional information available at https://www.decred.org/. 2015.

[Kia+16] Aggelos Kiayias et al. “Blockchain Mining Games”. In: *Proceedings of the 2016 ACM Conference on Economics and Computation (EC)*. 2016, pp. 365–382.

[Kia+17] Aggelos Kiayias et al. “Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol”. In: *CRYPTO*. http://eprint.iacr.org/2016/889. 2017.

[KMS14] Jonathan Katz, Andrew Miller, and Elaine Shi. *Pseudonymous Broadcast and Secure Computation from Cryptographic Puzzles*. Cryptology ePrint Archive, Report 2014/857. http://eprint.iacr.org/2014/857. 2014.

[KN12] Sunny King and Scott Nadal. “PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake”. In: https://peercoin.net/assets/paper/peercoin-paper.pdf. 2012.

[Kot] Jan Kotek. “IODB Storage Engine”. In: https://iohk.io/blog/scorex/iodb-storage-engine/.

[KP15] Aggelos Kiayias and Giorgos Panagiotakos. *Speed-Security Tradeoffs in Blockchain Protocols*. Cryptology ePrint Archive, Report 2015/1019. http://eprint.iacr.org/2015/1019. 2015.

[KP16] Aggelos Kiayias and Giorgos Panagiotakos. *On Trees, Chains and Fast Transactions in the Blockchain*. Cryptology ePrint Archive, Report 2016/545. http://eprint.iacr.org/2016/545. 2016.

[Kwo14] Jae Kwon. “TenderMint: Consensus without Mining”. In: https://tendermint.com/static/docs/tendermint.pdf. 2014.

[LK14] Yehuda Lindell and Jonathan Katz. *Introduction to modern cryptography*. Chapman and Hall/CRC, 2014.

[Lys02] Anna Lysyanskaya. “Unique Signatures and Verifiable Random Functions from the DH-DDH Separation”. In: 2002, pp. 597–612.

[Mil+14] Andrew Miller et al. “Permacoin: Repurposing Bitcoin Work for Data Preservation”. In: 2014, pp. 475–490. DOI: 10.1109/SP.2014.37.

[Mil+15] Andrew Miller et al. “Nonoutsourceable Scratch-Off Puzzles to Discourage Bitcoin Mining Coalitions”. In: 2015, pp. 680–691.

[MO16] Tal Moran and Ilan Orlov. *Proofs of Space-Time and Rational Proofs of Storage*. Cryptology ePrint Archive, Report 2016/035. http://eprint.iacr.org/2016/035. 2016.

[Nak08] Satoshi Nakamoto. “Bitcoin: A peer-to-peer electronic cash system”. In: https://bitcoin.org/bitcoin.pdf. 2008.

[Nay+15] Kartik Nayak et al. *Stubborn Mining: Generalizing Selfish Mining and Combining with an Eclipse Attack*. Cryptology ePrint Archive, Report 2015/796. http://eprint.iacr.org/2015/796. 2015.

[NXT14] NXT Community. “Nxt Whitepaper”. In: https://www.dropbox.com/s/cbuwrorf672c0yy/NxtWhitepaper_v122_rev4.pdf. 2014.

[Oka95] Tatsuaki Okamoto. “An Efficient Divisible Electronic Cash Scheme”. In: 1995, pp. 438–451.

[OO92] Tatsuaki Okamoto and Kazuo Ohta. “Universal Electronic Cash”. In: 1992, pp. 324–337.

[Par+15] Sunoo Park et al. *Spacemint: A Cryptocurrency Based on Proofs of Space*. Cryptology ePrint Archive, Report 2015/528. http://eprint.iacr.org/2015/528. 2015.

[PS17] Rafael Pass and Elaine Shi. "The Sleepy Model of Consensus". In: *Cryptology ePrint Archive, Report 2016/918*. http://eprint.iacr.org/2016/918. May 2017.

[PSS17a] Rafael Pass, Lior Seeman, and Abhi Shelat. "Analysis of the Blockchain Protocol in Asynchronous Networks". In: *EUROCRYPT*. https://eprint.iacr.org/2016/454. 2017.

[PSS17b] Rafael Pass, Lior Seeman, and Abhi Shelat. "Analysis of the Blockchain Protocol in Asynchronous Networks". In: 2017, pp. 643–673.

[Rey+16] Leonid Reyzin et al. "Improving Authenticated Dynamic Dictionaries, with Applications to Cryptocurrencies". In: http://eprint.iacr.org/2016/994. 2016.

[Sch+16] Okke Schrijvers et al. "Incentive Compatibility of Bitcoin Mining Pool Reward Functions". In: 2016, pp. 477–498.

[SLZ16] Yonatan Sompolinsky, Yoad Lewenberg, and Aviv Zohar. *SPECTRE: A Fast and Scalable Cryptocurrency Protocol*. Cryptology ePrint Archive, Report 2016/1159. http://eprint.iacr.org/2016/1159. 2016.

[SSZ16] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. "Optimal Selfish Mining Strategies in Bitcoin". In: 2016, pp. 515–532.

[SZ15] Yonatan Sompolinsky and Aviv Zohar. "Secure High-Rate Transaction Processing in Bitcoin". In: 2015, pp. 507–527. DOI: 10.1007/978-3-662-47854-7_32.

[Tor] Kyle Torpey. "One mining pool has had 50% of the Zcash network hashrate for the past month". In: https://twitter.com/kyletorpey/status/910622595388715020.

[Vas14] Pavel Vasin. “Blackcoin’s proof-of-stake protocol v2”. In: http://blackcoin.co/blackcoin-pos-protocol-v2-whitepaper.pdf. 2014.

[Wika] Bitcoin Wiki. “Protocol Documentation”. In: https://en.bitcoin.it/wiki/Protocol_documentation.

[Wikb] Bitcoin Wiki. “Script”. In: https://en.bitcoin.it/wiki/Script.

[Wikc] Bitcoin Wiki. “Transaction”. In: https://en.bitcoin.it/wiki/Transaction.

[Wikd] Wikipedia. *Nothing up my sleeve*. https://en.wikipedia.org/wiki/Nothing_up_my_sleeve_number.

[Zha+17] Fan Zhang et al. “REM: Resource-Efficient Mining for Blockchains”. In: *USENIX Security*. https://eprint.iacr.org/2017/179. 2017.