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

Tuyet Duong† Lei Fan‡ Hong-Sheng Zhou§

Cryptography Lab
Virginia Commonwealth University

November 11, 2016

Abstract

Cryptocurrencies like Bitcoin have proven to be a phenomenal success. Bitcoin-like systems use proof-of-work mechanism which is therefore considered as 1-hop blockchain, and 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.

We propose the first provably secure 2-hop blockchain by combining proof-of-work (first hop) and proof-of-stake (second hop) mechanisms. On top of Bitcoin's brilliant ideas of utilizing the power of the honest miners, via their computing resources, to secure the blockchain, we further leverage the power of the honest users/stakeholders, via their coins/stake, to achieve this goal. The security of our blockchain holds if the honest players control majority of the *collective* resources (which consists of both computing power and stake). 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.

*An early version with title "Securing Bitcoin-like Blockchains against a Malicious Majority of Computing Power" appeared in ePrint Archive in July 2016. The current version shares the same motivation. But the construction idea and modeling approach have been completely revised.

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

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

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

Contents

1 Introduction

Cryptocurrencies like Bitcoin [29] 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 puzzle [14, 1]) 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 come easy go" 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 resources — computing power.

This intuition has recently been investigated in academia. For example, Garay et al [17] and then Pass et al [33] 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 comprehensive 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 [18]. 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. Efforts have been made to address this crisis. In [27], 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?*

1.1 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 fragile 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 Bitcoin 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 say 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.

[1] You may also say the system *cost* a huge amount of computing resources in the past years.

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 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, no practical solution to an open blockchain is known 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.

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 1.5 (Related Work) for more discussion about open blockchain via alternative physical resources.

One may wonder if it is possible to emulate physical resources via virtual resources first, and then construct a fast protocol based on the emulated physical resources. This can be an interesting approach. However, solutions via high communication between players (e.g., using secure multi-party computation or similar techniques to generate a random beacon, then constructing a blockchain via the beacon) won't be practical because our goal is to enable an open blockchain. Communication overhead which depends on the number of players will limit the scope of the system immediately. A closed system may help manage the virtual resources; but then "converting" a closed system into an open blockchain could be very difficult which may introduce lots of communication between the protocol players; again, if the introduced communication overhead depends on the number of players, the designed protocol may not be scalable to a huge network like Bitcoin. In one word, it seems it is extremely difficult to mimic Nakamoto's footprint via virtual resources *only*.[2]

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 constructed by 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 [10]. However, random oracles do enable an elegant solution to open blockchains in practice.[3]

Additional considerations. There are many reasons that Bitcoin has become a successful system. 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. Secondly, 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.

Next, we provide our solution which meets all above considerations.

1.2 Our Scheme

From 1-hop to 2-hop blockchain. Nakamoto's system is powered by physical computing resources, and 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. That means, in addition to PoW-miners, a new type of players — PoS-holder (stakeholder) — is introduced

[2] Claims of building secure pure proof-of-stake blockchain have been made in the cryptocurrency community [6, 5, 12]. But, at the moment of writing the paper, no practical open blockchains are available.

[3] We note that Nakamoto's design is consistent with the folklore wisdom of a "nothing up my sleeve number" [38] which has been widely used in practical cryptographic designs.

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 1: green blocks are generated by PoW-miners in the first hops, while red blocks are produced by PoS-holders in the second hops; now naturally a PoW-chain consists the sequence of green blocks $B_1, B_2, B_3, \cdots$, and a PoS-chain consists the sequence of red 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 blue blocks in the figure.

In our protocol, PoW-chains and PoS-chains are plaited together in every time step, and these PoW/PoS-chains are extended alternately. In order to plait them tightly, we expect that in our scheme, each proof-of-work block (PoW-block) can be mapped to no more than one proof-of-stake block (PoS-block) and each PoW-block is linked to *both* previous PoW-block and PoS-block (See Figure 1). In this way, all valid PoS-chains will have nearly the same length as their corresponding PoW-chains. Naturally, a *chain-pair* consists of a valid PoS-chain and its corresponding PoW-chain.

Next, we provide more formal details. As mentioned above, the PoW/PoS-chains in the 2-hop protocol are extended alternately. Thus, the protocol consists of *PoW-rounds* and *PoS-rounds*, which execute alternately. In each round, each player (PoW-miner or PoS-holder) first determines a valid chain-pair with the longest PoW-chain; then the player attempts to extend the chain-pair. More concretely, in each PoW-round, PoW-miners extend the best valid chain-pair via proof-of-work (i.e., solving a hash inequality) where each new PoW-block is pointed to the previous PoW-block and PoS-block (Note that, this is the difference of our PoW-block from ordinary PoW-block); on the other hand, in the next PoS-round, a PoS-holder is chosen (i.e., based on the proof-of-work chain), and this PoS-holder then has the privilege to extend the best valid chain-pair on its view. We point out that, very intuitively here we treat the proof-of-work blockchain as a *biased random beacon* for electing a stakeholder in the corresponding PoS-round. We may view our scheme as a proof-of-stake scheme which uses a proof-of-work chain as a biased random beacon. Our scheme enjoys almost the same efficiency and scalability as the original Nakamoto scheme.

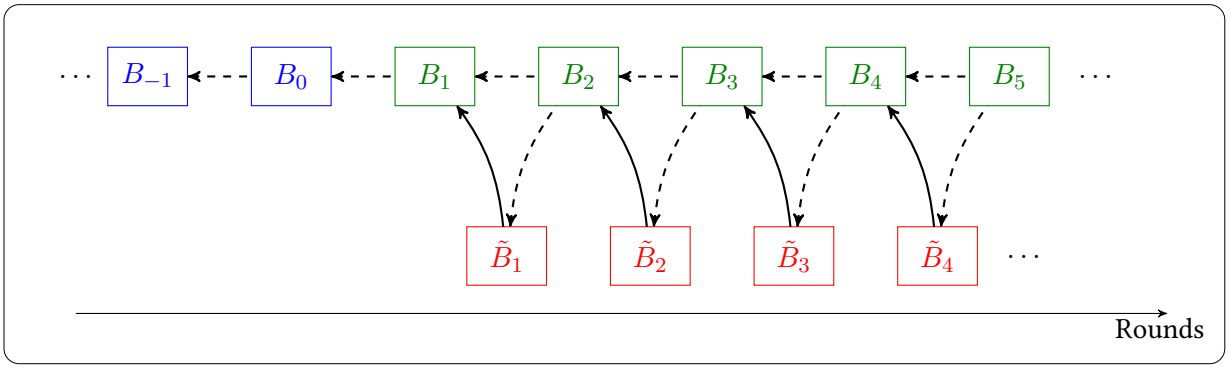


Figure 1: 2-hop blockchain structure

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

Why the scheme works? We here present the basic intuition for arguing the security of our scheme. Based on the protocol description above, although we do not link the PoS-chain explicitly, the adversary cannot manipulate an existing PoS-block because it is locked by the next PoW-block in the chain (i.e., each PoW-block is linked to its previous PoW-block and PoS-block.) In addition, in order to extend a PoW/PoS chain-pair, the adversary needs to control both hops: the adversary needs to first find a valid PoW solution (which defines a valid PoW block); this PoW block specifies a valid stakeholder; and now the adversary also needs to control

such stakeholder to complete the chain-pair extension. Intuitively, it is difficult to predict the identity of the specified stakeholder. Even in the setting that the adversary can find many PoW solutions, if he controls a very small portion of stakeholders, then the adversary may still not be able to produce more PoS-blocks than the honest players do. Based on this intuition, we can essentially prove the security of our blockchain if the honest players control majority of the *collective* resources (which consists of both computing power and stake). 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.

1.3 Our Modeling

We take the provable security approach in our design. Inspired by Garay et al [17] and Pass et al [33], we introduce a new analysis framework for (more involved) blockchain protocols. Under this framework, we prove the security for our proof-of-work/proof-of-stake 2-hop blockchain.

Along the way, we identify and formulate a set of resource setup functionalities, including resource random oracle functionality $\mathcal{F}^*_{\mathsf{rRO}}$ and resource certification functionality $\mathcal{F}^*_{\mathsf{rCERT}}$. Those resource setup functionalities precisely describe real world physical resources and virtual resources which are suitable for blockchain protocols. We note that these resource functionalities can be instantiated in the real world without introducing any *trapdoor*. More precisely, the resource random oracle functionality $\mathcal{F}^*_{\mathsf{rRO}}$ can be implemented via a random oracle and physical computing resource. On the other hand, the resource certification functionality $\mathcal{F}^*_{\mathsf{rCERT}}$ can be implemented via a "mature blockchain" (which can be further implemented via a random oracle and physical computing resource) and digital signature scheme.

We remark that identifying these "blockchain friendly" resource setup functionalities will significantly simplify the design and analysis of blockchain protocols since the details of managing physical/virtual resources are encapsulated inside these setup functionalities. We believe our way of formulating resource setup functionalities will help us identify and then formulate more useful resource setup functionalities (for, e.g., physical memory resources). Our way of formulating resource setup functionalities may eventually help us unify many different assumptions such as "honest majority of players", and "honest majority of computing power".

1.4 Summary of Our Contributions

Let's summarize our contributions here. In this work, we mimic Nakamoto's footprint and for the first time securely extend Nakamoto's idea beyond the proof-of-work mechanism, and design a simple, provably secure 2-hop consensus protocol by utilizing both physical computing resources and virtual resources (i.e., coins). We note that, this is the first effort to leverage the power of virtual resources for building provably secure open blockchains. Although the ideas of leveraging the power of virtual resources have been discussed in Bitcoin community, most of them cannot lead to scalable open blockchains; and before our work, none of them has been carefully analyzed. We also note that, ours is the first effort to combine two type of different resources for building practical open blockchains with provable security.

In addition, in this work, we put forth a rigorous framework which is suitable for analyzing more blockchain protocols. Previous analysis frameworks [17, 33] can be extended for achieving this goal but here we make it explicit. We want to emphasize that, we identify and formulate several resource setup functionalities which capture the exact intuition of real world physical resources and virtual resources. Put it differently, we are making effort, thru these resource setup functionalities, to mathematically describe real world assumptions such as the assumption of "honest majority of computing power in Bitcoin system". It is important to note that, our resource setup functionalities will much simplify our design and analysis of blockchain protocols.

Finally, we remark that leveraging virtual resources for building provably secure, open blockchain is a subtle and difficult task. Not very careful treatments may lead to non-scalable[4] open blockchains, or make it

[4]Blockchain protocols such as PoW-based blockchains [29] that utilize physical assumptions (e.g., computational power) have a

difficult/infeasible to achieve provable security. Our 2-hop design benefits from our careful understanding of the power of physical resources and virtual resources.[5] Note that, our 2-hop design can be viewed as a natural extension of Nakamoto's 1-hop design via proof-of-work mechanism (i.e., the second hop is deterministic and always true.) Our 2-hop design can also be viewed as a proof-of-stake scheme (which uses a proof-of-work chain as a *biased* random beacon.) However, we explicitly remark here that our design will **not** lead to any *pure* proof-of-stake blockchain.

1.5 Related Work

We list closely related work below. A more comprehensive related work will be included in a future version.

Digital currency and Bitcoin. Anonymous digital currency was introduced by Chaum [11] in the early 1980s. The first decentralized currency system, Bitcoin [29], was launched about 30 years later, by incentivizing a set of players to solve moderately-hard cryptographic puzzles (also called proofs-of-work puzzles [14, 1]). Recently, the security of Bitcoin system has been analyzed in the rational setting, e.g., [16, 15, 30, 21, 34, 35], and also in cryptographic setting [17, 33, 36, 22, 23]. Three 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 [17]. The chain growth property was first formally defined by Kiayias et al [22]. The common prefix property was later strengthened by Pass et al [33]. In our study, we adopt the stronger variant of the common prefix property by Pass et al [33] together with the chain quality and chain growth from [22, 17].

Cryptocurrency via alternative physical resources. Similar to PoW, alternative consensus techniques via different physical resources have been considered to replace computing power. For example, the physical storage resource (as opposed to PoW's computational time,) is used in [32, 26]. Between the use of space/memory and the use of time, are *proofs of space time* introduced in [28]. This is a hybrid proof system utilizing both computational and space resources. Intel proposes the use of trusted hardware for blockchain protocols in [20].

Cryptocurrency via virtual resources. The final proof system we consider, proof-of-stake (PoS), is a relatively inexpensive technique in computational overhead when compared to PoW as it relies on internal resources to the system. 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 [4], several variants of PoS that have been proposed and implemented in real cryptocurrencies including [12, 25, 37, 2]. 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. Interesting ideas of combining virtual resource with physical resource are also proposed. For example, in [24, 13, 3], the proof of work and proof of stake can be combined together. We remark that, there is no known practical PoS related consensus with provable security.

Organization. In Section 2, we present our analysis framework. In Section 3, we present the details of our construction. Then, in Section 4 we analyze the security of our construction.

very efficient communication complexity in terms of the number of players. Those feature more than thousands of network nodes, demonstrating node *scalability* in practice. A not careful design may not be able to give us a communication efficient, thus scalable, protocol as (or close to) Nakamoto's.

[5] Our protocol achieves the node scalability. We utilize physical computing resources to maintain a proof-of-work blockchain which is informally treated as a *biased random beacon* to elect stakeholders for maintaining a proof-of-stake blockchain.

2 Model

2.1 2-hop Blockchain Protocol Executions

In order to study the security of Bitcoin-like protocols, Garay et al [17] and then Pass et al [33] set up the first cryptographic models by following Canetti's formulation of the "real world" executions [7, 8]. 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 this asynchronous communication model. 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 specify the delivery time to each player in the system. Note that, if the delivery time specified by the adversary 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' messages before deciding his strategy; the adversary may "spoof" the source of a message they transmit and impersonate the (honest) sender of the message. In more detail, each player may request $\mathcal{F}_{\mathsf{NET}}$ to broadcast a message m by command $(\textsc{Broadcast}, m)$; on the other hand, the functionality may deliver a message m to a player P by command $(\textsc{Message}, P, m)$.

PoW-miners and PoS-holders. We specify two types of players PoW-miner and 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-miners/PoS-holders may have a different amount of computing power/stake; nevertheless, this idealized model does not sacrifice generality since one can imagine that real honest 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 are chain-pairs from the network, each PoW-miner would choose a best valid chain-pair, and then exploit its 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 its 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 $\{\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_{\mathsf{NET}}\}$-hybrid execution of 2-hop blockchain protocol. Existing rigorous formulations (e.g., [17, 33]) apply for 1-hop protocols (e.g., Nakamoto's protocol) where the system is maintained by a single type of players, i.e., PoW-miners. Here, we move from 1-hop protocols to 2-hop protocols (i.e., hybrid proof-of-work/proof-of-stake protocols) and present a formal treatment for it.

Following the framework for Universal Composability [8], we present an abstract model for hybrid proof-

of-work/proof-of-stake blockchain protocol $\Pi = (\Pi^{\mathsf{w}}, \Pi^{\mathsf{s}})$ in the $\{\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_{\mathsf{NET}}\}$-hybrid model where Π^{w} and Π^{s} denote the code run by PoW-miners and by 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 a n number of PoW-miners and $\tilde{n}$ number of PoS-holders. The execution proceeds in *rounds*. Without lost of generality, we assume that even rounds correspond to PoW-miners, and odd rounds correspond to PoS-holders. The environment $\mathcal{Z}$ can "manage" protocol players thru a *mobile* adversary [31]: the adversary $\mathcal{A}$ can corrupt an honest party and uncorrupt a corrupted party. Note that, here, the blockchain protocol Π is parameterized by a predicate $V(\cdot)$ which determines the proper structure of the information that is stored into the blockchain.

More concretely, the $\{\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_{\mathsf{NET}}\}$-hybrid execution proceeds as follows. The environment $\mathcal{Z}$ first actives the adversary $\mathcal{A}$ and provides instructions for the adversary. The execution proceeds in rounds, and in each round, a protocol party could be activated by the environment or the functionalities.

1. In each odd round, each PoW-miner $\mathcal{W}_i$, for $1 \leq i \leq n$, proceeds as follows.
   - When PoW-miner $\mathcal{W}_i$ is activated by the environment $\mathcal{Z}$ with message $(\textsc{Input-Work}, \mathcal{W}_i, x)$ where x is the input of the execution, and potentially receive incoming message $(\textsc{Message}, \mathcal{W}_i, m)$ from $\mathcal{F}_{\mathsf{NET}}$ where m is the message received from the network. It then interacts with the functionality $\mathcal{F}_1$ and receives some output y.
   - Then execute the protocol Π^{w} on input its local state $state_i$, the value y received from the functionality $\mathcal{F}_1$, the input from the environment $\mathcal{Z}$, and the message m received from the functionality $\mathcal{F}_{\mathsf{NET}}$; and then output an update local state $state_i$ and an outgoing message m', i.e.,$\{state_i, m'\} \leftarrow \Pi^{\mathsf{w}}(state_i, x, y, m)$. After that, send $(\textsc{Broadcast}, m')$ to $\mathcal{F}_{\mathsf{NET}}$ and then send $(\textsc{Return-Work}, \mathcal{W}_i)$ to the environment $\mathcal{Z}$.
2. In each even round, each PoS-holder $\mathcal{S}_j$, for $n+1 \leq j \leq n+\tilde{n}$, proceeds as follows.
   - When PoS-holder $\mathcal{S}_j$ is activated by the environment $\mathcal{Z}$ with message $(\textsc{Input-Stake}, \mathcal{S}_j, \tilde{x})$ where $\tilde{x}$ is the input from the environment, and potentially receive subroutine output message $(\textsc{Message}, \mathcal{S}_j, m)$ from $\mathcal{F}_{\mathsf{NET}}$. It then interacts with the functionality $\mathcal{F}_2$ and receives some output $\tilde{y}$.
   - Next, execute the protocol Π^{s} on input its local state $state_j$, the value $\tilde{y}$ received from the functionality $\mathcal{F}_2$, an input from the environment $\tilde{x}$, and the message m received from the functionality $\mathcal{F}_{\mathsf{NET}}$; and then obtain an update local state $state_j$ and an outgoing message m', i.e.,$\{state_j, m'\} \leftarrow \Pi^{\mathsf{s}}(state_j, \tilde{x}, \tilde{y}, m)$. After that, send $(\textsc{Broadcast}, m')$ to $\mathcal{F}_{\mathsf{NET}}$ and then return $(\textsc{Return-Stake}, \mathcal{S}_j)$ to the environment $\mathcal{Z}$.
3. At any point of the execution, $\mathcal{Z}$ can send message $(\textsc{Corrupt}, \mathcal{W}_i)$ or $(\textsc{Corrupt}, \mathcal{S}_j)$ to adversary $\mathcal{A}$ instruct $\mathcal{A}$ to corrupt a PoW-miner $\mathcal{W}_i$ or PoS-holder $\mathcal{S}_j$. From that point, $\mathcal{A}$ has access to the party's local state and controls $\mathcal{W}_i$.
4. At any point of the execution, $\mathcal{Z}$ can send message $(\textsc{Uncorrupt}, \mathcal{W}_i)$ or $(\textsc{Uncorrupt}, \mathcal{S}_j)$ to adversary $\mathcal{A}$ instruct $\mathcal{A}$ to uncorrupt a PoW-miner $\mathcal{W}_i$ or PoS-holder $\mathcal{S}_j$. This means that $\mathcal{A}$ no longer controls $\mathcal{W}_i/\mathcal{S}_j$ and instead player $\mathcal{W}_i$ starts executing $\Pi^{\mathsf{w}}/\Pi^{\mathsf{s}}$ with a fresh state.

Let $\mathsf{EXEC}^{\mathcal{F}_1,\mathcal{F}_2,\mathcal{F}_{\mathsf{NET}}}_{\Pi,\mathcal{A},\mathcal{Z}}$ 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 $\{\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_{\mathsf{NET}}\}$-hybrid execution; note that this joint view fully determines the execution. Whenever $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_{\mathsf{NET}}$ are clear from context we often write $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$ or $\mathsf{EXEC}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}$.

2.2 Resource Random Oracle Functionality $\mathcal{F}^*_{\mathsf{rRO}}$

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 $\mathcal{F}^*_{\mathsf{rRO}}$ which manages a huge "farm of computing devices", and these devices are provided by the environment $\mathcal{Z}$ through the adversary. In order to exploit the computing power of the functionality, each player needs to register the computing services of $\mathcal{F}^*_{\mathsf{rRO}}$ (by paying money in reality). In addition, he can disconnect the services (by stopping the payment). Indeed, this captures the dynamic computing power setting where different players could consume the computing resource for different windows of time. Functionality $\mathcal{F}^*_{\mathsf{rRO}}$ abstracts out of 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 query $\mathcal{F}^*_{\mathsf{rRO}}$ one *search* which consumes one unit of computing power granted in each round. Besides the computing services, the setup also provides the verification services which allow any player to verify solutions in many times, or computing services which allow any player to perform regular random oracle queries in many times.

More concretely, at any time, a PoW-miner $\mathcal{W}_i$ can send a register command $(\textsc{Work-Register}, \mathcal{W}_i)$ to ask for registration. The functionality then asks the adversary to specify whether the player can register or not. If the adversary allows $\mathcal{W}_i$ to use the computing resource ($\mathcal{W}_i$ is granted the computing resource), the functionality would record $(\mathcal{W}_i, \mathsf{b}_i)$ where $\mathsf{b}_i = 1$. If the player unregisters the services, this bit would be set to 0 indicating that the resource will not be granted to this player any longer. Note that, here, we follow Universal Composability [8] letting the environment $\mathcal{Z}$ (though the adversary) specify who will receive the computing resource and who will not.

The protocol executes in rounds, and for each round, the functionality sets a bit $b^{\mathsf{w}}_i = 0$ for every registered player $\mathcal{W}_i$ meaning that the player $\mathcal{W}_i$ is granted *one unit of computational resource*. Note that if the computational unit has been used by $\mathcal{W}_i$ by issuing a search query, the bit $b^{\mathsf{w}}_i = 0$ is set to 1, and then this player will not be able to request any other search query. A registered PoW-miner $\mathcal{W}_i$ may request the query to search for a PoW solution, and he can only find it with a certain probability p. More precisely, once he queried the computing services from the functionality by command $(\textsc{Search}, h, \mathcal{W}_i)$, the functionality then checks if there exists a record $(\mathcal{W}_i, \mathsf{b}_i)$; this means the functionality checks if this player is already granted the required computing resource. If the resource is granted, and $b^{\mathsf{w}}_i = 0$ meaning that this player has not used the resource allocated in the current round, the functionality would then with probability p, choose a random pair (w, h) and record an entry $(\mathcal{W}_i, \langle h, w\rangle, h)$. There, if $\mathcal{W}_i$ queries more than one time in this round, the functionality would not perform any computation since the resource is exploited.

In contrast to the computing services, every player has access to the verification or regular random oracle services many times by sending command $(\text{RO-}\textsc{Verify}, B)$ or $(\textsc{Compute}, B, X)$ to the functionality where B is a PoW-block and X is a generic payload. In regular random oracle services, the functionality on the command $(\textsc{Compute}, B, X)$, where (B, X) is specified by the environment, executes an ordinary random oracle query (i.e., check if (B, X) is queried and returns a random string corresponding to (B, X)). In verification services, the functionality $\mathcal{F}^*_{\mathsf{rRO}}$ then returns the random string mapped to $B = \langle h, w\rangle$ if stored. We emphasize that the regular random oracle queries are independent from the search queries. This implies that the random oracle used for the search query is different from that for the regular query. We then denote h as the random string returned by the random oracle for the regular query, and denote h' as that for the search query. Please refer to Figure 2 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.

Roughly, our $\mathcal{F}^*_{\mathsf{rRO}}$ is closely related to $\mathcal{F}_{\mathsf{Tree}}$ in [33]. In [33], a "per protocol" approach is taken. That is, for different blockchain protocol, say GHOST protocol [36], 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 resource random oracle functionality $\mathcal{F}^*_{\mathsf{rRO}}$ can be used for different PoW based blockchain protocols, and we don't need to revise the

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

The functionality is parameterized by a PoW hardness parameter p, a PoW security parameter κ, and interacts with PoW-miners, PoS-holders, as well as an adversary $\mathcal{A}$.

Computing Resource Registration.

1. Upon receiving a message $(\text{WORK-REGISTER}, \mathcal{W}_i)$ from party $\mathcal{W}_i$, it then passes the message to the adversary. Upon receiving a message $(\text{WORK-REGISTERED}, \mathcal{W}_i)$ from the adversary, set $\mathsf{b}_i := 1$, record $(\mathcal{W}_i, \mathsf{b}_i)$, and pass the message to the party $\mathcal{W}_i$ (the party $\mathcal{W}_i$ registered.)
2. Upon receiving a message $(\text{WORK-UNREGISTER}, \mathcal{W}_i)$ from party $\mathcal{W}_i$, it then sets $\mathsf{b}_i := 0$, and updates $(\mathcal{W}_i, \mathsf{b}_i)$, and then sends $(\text{WORK-UNREGISTERED}, \mathcal{W}_i)$ to the party $\mathcal{W}_i$(the party $\mathcal{W}_i$ unregistered.)

For each round, set $b^{\mathsf{w}}_i := 0$ for every registered party $\mathcal{W}_i$, then proceed as follows.

Regular Query. Upon receiving $(\text{COMPUTE}, B, X)$ from a party P, if there is record of the form (B, X, h), send $(\text{COMPUTED}, h)$ to the player P. Otherwise, choose random $h \in \{0,1\}^\kappa$, send $(\text{COMPUTED}, h)$ to the player P and record (B, X, h).

Work Query. Upon receiving $(\text{SEARCH}, \mathcal{W}_i, h)$ from a PoW-miner $\mathcal{W}_i$ where $h \in \{0,1\}^\kappa$, proceed as follows.

1. If $(\mathcal{W}_i, \mathsf{b}_i)$ is recorded where $\mathsf{b}_i = 1$ and $b^{\mathsf{w}}_i = 0$ (the party $\mathcal{W}_i$ registered and granted one unit of computational resource), then
   - with probability p, choose uniformly a pair (w, h') where $w, h' \in \{0,1\}^\kappa$. Then set $b^{\mathsf{w}}_i := 1$, and record $(\mathcal{W}_i, \langle h, w\rangle, h')$. Then send $(\text{SEARCHED}, \mathcal{W}_i, w)$ to the player $\mathcal{W}_i$ (the party $\mathcal{W}_i$ discovers the solution,)
   - with probability $1-p$, set $b^{\mathsf{w}}_i := 1$, and send $(\text{SEARCHED}, \mathcal{W}_i, \perp)$ to the player $\mathcal{W}_i$ (the party $\mathcal{W}_i$ does not discover the solution.)
2. Otherwise, if any of the following cases occur:
   - if $(\mathcal{W}_i, \mathsf{b}_i)$ is not recorded (the party $\mathcal{W}_i$ is not registered yet),
   - or if $(\mathcal{W}_i, \mathsf{b}_i)$ is recorded and $\mathsf{b}_i = 0$ *(the party $\mathcal{W}_i$ registered and then unregistered),*
   - or if $b^{\mathsf{w}}_i = 1$ (the party $\mathcal{W}_i$ already used the granted computational resource in this round),

   Then send $(\text{SEARCHED}, \mathcal{W}_i, \perp)$ to the player $\mathcal{W}_i$ (the party $\mathcal{W}_i$ does not discover the solution.)

Work Verification Query. Upon receiving (RO-VERIFY, B) from a party P, parse B as $\langle h, w\rangle$ check if there exists a recorded entry $(\cdot, \langle h, w\rangle, \cdot)$ (the party $\mathcal{W}_i$ found the solution.) If yes and the entry is $(\mathcal{W}_i, \langle h, w\rangle, h')$ send (RO-VERIFIED, h') to the party P. Otherwise, send (RO-VERIFIED, $\perp$) to P.

Figure 2: Resource Random Oracle Functionality.

setup *per protocol*. Furthermore, we note that, we push the functionality even more formal by modeling the distribution of computing resource from the environment to parties. In our model, each party can register and receive the computing resource under the control of the environment. That said, this model naturally captures the joining of new players or the rejoining of old players.

We remark that our $\mathcal{F}^*_{\mathsf{rRO}}$ is a very costly setup in the sense that, lots of computing resources can be provided in each time window. In our paper, we will use this physical resource (i.e., computing power) setup together with another virtual resource (i.e., stake) setup $\mathcal{F}^*_{\mathsf{rCERT}}$ to design Bitcoin like blockchain. Note that virtual resource setup $\mathcal{F}^*_{\mathsf{rCERT}}$ is much less costly.

2.2.1 How to Implement $\mathcal{F}^*_{\mathsf{rRO}}$

As discussed in section 2.2, the functionality $\mathcal{F}^*_{\mathsf{rRO}}$ is implemented by a random oracle functionality $\mathcal{F}_{\mathsf{RO}}$ [19]. We denote ϕ^*_{rRO} as the ideal protocol for an ideal functionality $\mathcal{F}^*_{\mathsf{rRO}}$ and π_{RO} as the protocol in the $\mathcal{F}_{\mathsf{RO}}$-hybrid

model. In the ideal protocol ϕ^*_{rRO}, players are dummy since they just forward the messages received from the environment $\mathcal{Z}$ to the functionality $\mathcal{F}^*_{\mathsf{rRO}}$, 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_{RO} 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. The protocol π_{RO} is described in Figure 3.

PROTOCOL π_{RO}

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

Each PoW-miner $\mathcal{W}_i$ proceeds as follows.

1. Upon receiving (WORK-REGISTER, $\mathcal{W}_i$) from the environment $\mathcal{Z}$, pass the message to the adversary. Upon receiving (WORK-REGISTERED, $\mathcal{W}_i$) from the adversary, set $\mathsf{b}_i := 1$, record b_i, and pass the message (WORK-REGISTERED, $\mathcal{W}_i$) to the environment.

   Upon receiving (WORK-UNREGISTER, $\mathcal{W}_i$) from the environment $\mathcal{Z}$, pass the message to the adversary. Upon receiving (WORK-UNREGISTERED, $\mathcal{W}_i$) from the adversary set $\mathsf{b}_i := 0$, record b_i, and pass the message (WORK-UNREGISTERED, $\mathcal{W}_i$) to the environment.

2. For each round, each registered party $\mathcal{W}_i$ sets $b_i^{\mathsf{w}} := 0$, then proceeds as follows. Upon receiving (SEARCH, $\mathcal{W}_i, h$) from the environment $\mathcal{Z}$,

   - If $\mathsf{b}_i = 1$ and $b_i^{\mathsf{w}} := 0$, choose random $w \in \{0,1\}^\kappa$, and then query the functionality $\mathcal{F}_{\mathsf{RO}}$ on input (h, w) and then obtain output h'. If $h > \mathrm{D}$ where $\mathrm{D} = p \cdot 2^\kappa$, set $w :=\perp$. Then send (SEARCHED, $\mathcal{W}_i, w$) to the environment.
   - Otherwise, if b_i is not recorded, or if b_i is recorded and $\mathsf{b}_i = 0$, or if $b_i^{\mathsf{w}} = 1$, send (SEARCHED, $\mathcal{W}_i, \perp$) to the environment.

Each player P proceeds as follows.

1. Upon receiving (COMPUTE, B, X) from the environment $\mathcal{Z}$, send (B, X) to the functionality $\mathcal{F}_{\mathsf{RO}}$ and receive h. Then record h and send (COMPUTED, h) to the environment.
2. Upon receiving (RO-VERIFY, B) from the environment $\mathcal{Z}$, send B to the functionality $\mathcal{F}_{\mathsf{RO}}$ and receive h'. If $h' \leq \mathrm{D}$, send (RO-VERIFIED, h') to the environment. Otherwise, if $h' > \mathrm{D}$, send (RO-VERIFIED, $\perp$) to the environment.

Figure 3: Resource Random Oracle Protocol.

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

Lemma 2.1. *Consider ϕ^*_{rRO} described above and π_{RO} in Figure 3. It holds that the two ensembles $\mathsf{EXEC}^{\mathcal{F}_{\mathsf{RO}}}_{\pi_{\mathsf{RO}},\mathcal{A},\mathcal{Z}}$ and $\mathsf{EXEC}^{\mathcal{F}^*_{\mathsf{rRO}}}_{\phi^*_{\mathsf{rRO}},\mathcal{S},\mathcal{Z}}$ are perfectly indistinguishable.*

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

1. Upon receiving (WORK-REGISTER, $\mathcal{W}_i$) from $\mathcal{A}$, pass the message to the functionality $\mathcal{F}^*_{\mathsf{rRO}}$ and receive (WORK-REGISTERED, $\mathcal{W}_i$) from the functionality. Then output (WORK-REGISTERED, $\mathcal{W}_i$) to $\mathcal{A}$.
2. Upon receiving (WORK-UNREGISTER, $\mathcal{W}_i$) from $\mathcal{A}$, pass the message to the functionality $\mathcal{F}^*_{\mathsf{rRO}}$ and receive (WORK-UNREGISTERED, $\mathcal{W}_i$) from the functionality. Then output (WORK-UNREGISTERED, $\mathcal{W}_i$) to

$\mathcal{A}$.

3. Upon receiving (B, X) from $\mathcal{A}$, if there is a record $((B, X), h)$ in T, then send h to $\mathcal{A}$. Otherwise, if there is no record of the form $((B, X), \cdot)$ in T, send $(\textsc{Compute}, B, X)$ to $\mathcal{F}^*_{\mathsf{rRO}}$ and receive $(\textsc{Computed}, h)$. Then record $((B, X), h)$ and send h to $\mathcal{A}$.

4. Upon receiving (h, w) from $\mathcal{A}$ in the name of $\mathcal{W}_i$, if there is a record $((h, w), h')$ in T, then send h' to $\mathcal{A}$. Otherwise, if there is no record $((h, w), h')$, send $(\textsc{Search}, \mathcal{W}_i, h)$ to $\mathcal{F}^*_{\mathsf{rRO}}$ and receive $(\textsc{Searched}, \mathcal{W}_i, w')$. If $w' = \perp$, choose random $h' \in \{0,1\}^\kappa$ such that $h' > \mathrm{D}$ where $\mathrm{D} = p \cdot 2^\kappa$. Otherwise, if $w' \neq \perp$, choose random $h' \in \{0,1\}^\kappa$ such that $h' \leq \mathrm{D}$. Then record $((h, w), h')$, and send h' to $\mathcal{A}$.

5. Upon receiving B from $\mathcal{A}$ where $B = \langle h, w \rangle$, if there is a record of the form (B, h'') in T, send h'' to $\mathcal{A}$. Otherwise, if there is no record of the form $(B, \cdot)$ in T, send $(\text{RO-}\textsc{Verify}, B)$ to $\mathcal{F}^*_{\mathsf{rRO}}$ and receive $(\text{RO-}\textsc{Verified}, h')$; if $h' = \perp$, choose random $h'' \in \{0,1\}^\kappa$ such that $h'' > \mathrm{D}$, otherwise, if $h' \neq \perp$, choose random $h'' \in \{0,1\}^\kappa$ such that $h'' \leq \mathrm{D}$; record (B, h'') and send h'' to $\mathcal{A}$.

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

□

2.3 Resource Certification Functionality $\mathcal{F}^*_{\mathsf{rCERT}}$

In this subsection, we introduce our resource certification functionality $\mathcal{F}^*_{\mathsf{rCERT}}$ describing the usage of virtual resource in our system. Essentially, at any time step, a PoS-holder $\mathcal{S}_j$ can send a register command $(\textsc{Stake-Register}, \mathcal{S}_j)$ to $\mathcal{F}^*_{\mathsf{rCERT}}$ for registration. Similarly to $\mathcal{F}^*_{\mathsf{rRO}}$, the functionality then records $(\mathcal{S}_j, \mathsf{b}_j)$ where $\mathsf{b}_j = 1$, if permitted by the adversary. If the player discontinues the services, this bit is set to 0 indicating that the stake would not be granted to this player any longer. Then, for each execution round, a registered PoS-holder $\mathcal{S}_j$ is granted *one unit of the virtual resource*, and he can then request the functionality for leader election in this round. Specifically, he can send message $(\textsc{Elect}, \mathcal{S}_j, B)$ to the functionality; the functionality then with probability $\tilde{p}$ selects this party as the leader and notifies the player whether he is selected or not. Next, if the party $\mathcal{S}_j$ is elected by the functionality as the leader, the elected party then asks the functionality $\mathcal{F}^*_{\mathsf{rCERT}}$ to provide the signature of $(B, X, \mathcal{S}_j)$. The functionality therefore requests the adversary to produce the signature by command $(\textsc{Sign}, (\mathcal{S}_j, B, X))$, and then waits until the adversary responds by a signature σ. The functionality after that checks if no entry $((\mathcal{S}_j, B, X), \sigma, 0)$ has been recorded. Note that, the indicator 0 implies that this is not a valid signature (the verification fails). If this entry is not recorded, then the functionality simply passes the signature σ to $\mathcal{S}_j$ and stores $((\mathcal{S}_j, B, X), \sigma, 1)$. If entry $((\mathcal{S}_j, B, X), \sigma, 0)$ is already recorded, this implies no signature has been generated for $(\mathcal{S}_j, B, X)$ yet. Then, the functionality outputs an error and halts.

Next, the verification process of $\mathcal{F}^*_{\mathsf{rCERT}}$ proceeds as follows. Upon receiving a verification request, the functionality then asks the adversary verify the signature. The functionality, upon receiving the verification decision from the adversary, would ensure the completeness, unforgeability, and guarantees consistency properties of the signature scheme. Please refer to Figure 4 and [9] for more details.

2.3.1 How to Implement $\mathcal{F}^*_{\mathsf{rCERT}}$

Our functionality $\mathcal{F}^*_{\mathsf{rCERT}}$ can be a "resource" analog of the certificate functionality $\mathcal{F}_{\mathsf{CERT}}$ in [9]. Note that $\mathcal{F}_{\mathsf{CERT}}$ can be implemented in the $\{\mathcal{F}_{\mathsf{CA}}, \mathcal{F}_{\mathsf{SIG}}\}$-hybrid model [9]. We can follow the approach to implement

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

The functionality is parameterized by an election parameter $\tilde{p}$, a security parameter κ, and interacts with PoS-holders, PoW-miners, as well as an adversary $\mathcal{A}$.

Stake Resource Registration.

1. Upon receiving a message $(\text{STAKE-REGISTER}, \mathcal{S}_j)$ from party $\mathcal{S}_j$, it then passes the message to the adversary. Upon receiving a message $(\text{STAKE-REGISTERED}, \mathcal{S}_j)$ from the adversary, set $\mathsf{b}_j := 1$, record $(\mathcal{S}_j, \mathsf{b}_j)$, and pass the message to the party (the party $\mathcal{S}_j$ registered.)
2. Upon receiving a message $(\text{STAKE-UNREGISTER}, \mathcal{S}_j)$ from party $\mathcal{S}_j$, it then sets $\mathsf{b}_j := 0$, and updates $(\mathcal{S}_j, \mathsf{b}_j)$, and sends $(\text{STAKE-UNREGISTERED}, \mathcal{S}_j)$ to the party (the party $\mathcal{S}_j$ unregistered.)

For each round, set $b^{\mathsf{s}}_j := 0$ for every registered party $\mathcal{S}_j$, then proceed as follows.

Stake Election: Upon receiving $(\text{ELECT}, \mathcal{S}_j, B)$ from a PoS-holder $\mathcal{S}_j$, proceed as follows.

1. If $(\mathcal{S}_j, \mathsf{b}_j)$ is recorded where $\mathsf{b}_j = 1$ and $b^{\mathsf{s}}_j = 0$ (the party $\mathcal{S}_j$ registered and granted one unit of virtual resource), then
   - with probability $\tilde{p}$, choose random $h \in \{0,1\}^\kappa$. Then set $b^{\mathsf{s}}_j := 1$, and record $(B, \mathcal{S}_j, h)$. Then set $\mathsf{f} := 1$, send $(\text{ELECTED}, \mathcal{S}_j, \mathsf{f})$ to $\mathcal{S}_j$, and record the entry $(B, \mathcal{S}_j, h)$ (the party $\mathcal{S}_j$ is elected.)
   - with probability $1 - \tilde{p}$, set $b^{\mathsf{s}}_j := 1$ and $\mathsf{f} := 0$, and send $(\text{ELECTED}, \mathcal{S}_j, \mathsf{f})$ to $\mathcal{S}_j$ (the party $\mathcal{S}_j$ is not elected.)
2. Otherwise, if any of the following cases occur:
   - $(\mathcal{S}_j, \mathsf{b}_j)$ is not recorded (the party $\mathcal{S}_j$ is not registered yet),
   - or $(\mathcal{S}_j, \mathsf{b}_j)$ is recorded and $\mathsf{b}_j = 0$ (the party $\mathcal{S}_j$ registered and then unregistered),
   - or $b^{\mathsf{s}}_j = 1$ (the party $\mathcal{S}_j$ already used the granted resource unit),

   Then set $\mathsf{f} := 0$ and send $(\text{ELECTED}, \mathcal{S}_j, \mathsf{f})$ to $\mathcal{S}_j$ (the party $\mathcal{S}_j$ is not elected.)

Signature Generation: Upon receiving $(\text{SIGN}, \mathcal{S}_j, B, X)$ from a party $\mathcal{S}_j$, send $(\text{SIGN}, (\mathcal{S}_j, B, X))$ to the adversary. Upon receiving $(\text{SIGNATURE}, (\mathcal{S}_j, B, X), \sigma)$ from the adversary, verify that no entry $((\mathcal{S}_j, B, X), \sigma, 0)$ is recored. If it is, then output an error message and halt. Else, output $(\text{SIGNED}, (\mathcal{S}_j, B, X), \sigma)$ to $\mathcal{S}_j$, and record the entry $((\mathcal{S}_j, B, X), \sigma, 1)$.

Stake Verification: Upon receiving $(\text{STAKE-VERIFY}, (\mathcal{S}_j, B, X), \sigma)$ from a party P,

1. If there exists a record of the form $(B, \mathcal{S}_j, \cdot)$ (the party $\mathcal{S}_j$ is elected), hand $(\text{STAKE-VERIFY}, (\mathcal{S}_j, B, X), \sigma)$ to the adversary. Upon receiving $(\text{STAKE-VERIFIED}, (\mathcal{S}_j, B, X), \phi)$ from the adversary, do:
   - If $((\mathcal{S}_j, B, X), \sigma, 1)$ is recorded, then set $f := 1$.
   - Else, if $\mathcal{S}_j$ is not corrupted, and no entry $((\mathcal{S}_j, B, X), \sigma', 1)$ for any σ' is recorded, then set $f := 0$ and record the entry $((\mathcal{S}_j, B, X), \sigma, f)$.
   - Else, if there is an entry $((\mathcal{S}_j, B, X), \sigma, f')$, then set $f := f'$.
   - Else, set $f := \phi$, and record the entry $((\mathcal{S}_j, B, X), \sigma, f)$.

   Output $(\text{STAKE-VERIFIED}, (\mathcal{S}_j, B, X), f)$ to the party P.
2. Otherwise, if there is no record of the form $(B, \mathcal{S}_j, \cdot)$ (the party $\mathcal{S}_j$ is not elected), set $f := 0$ and output $(\text{STAKE-VERIFIED}, (\mathcal{S}_j, B, X), f)$ to the party P.

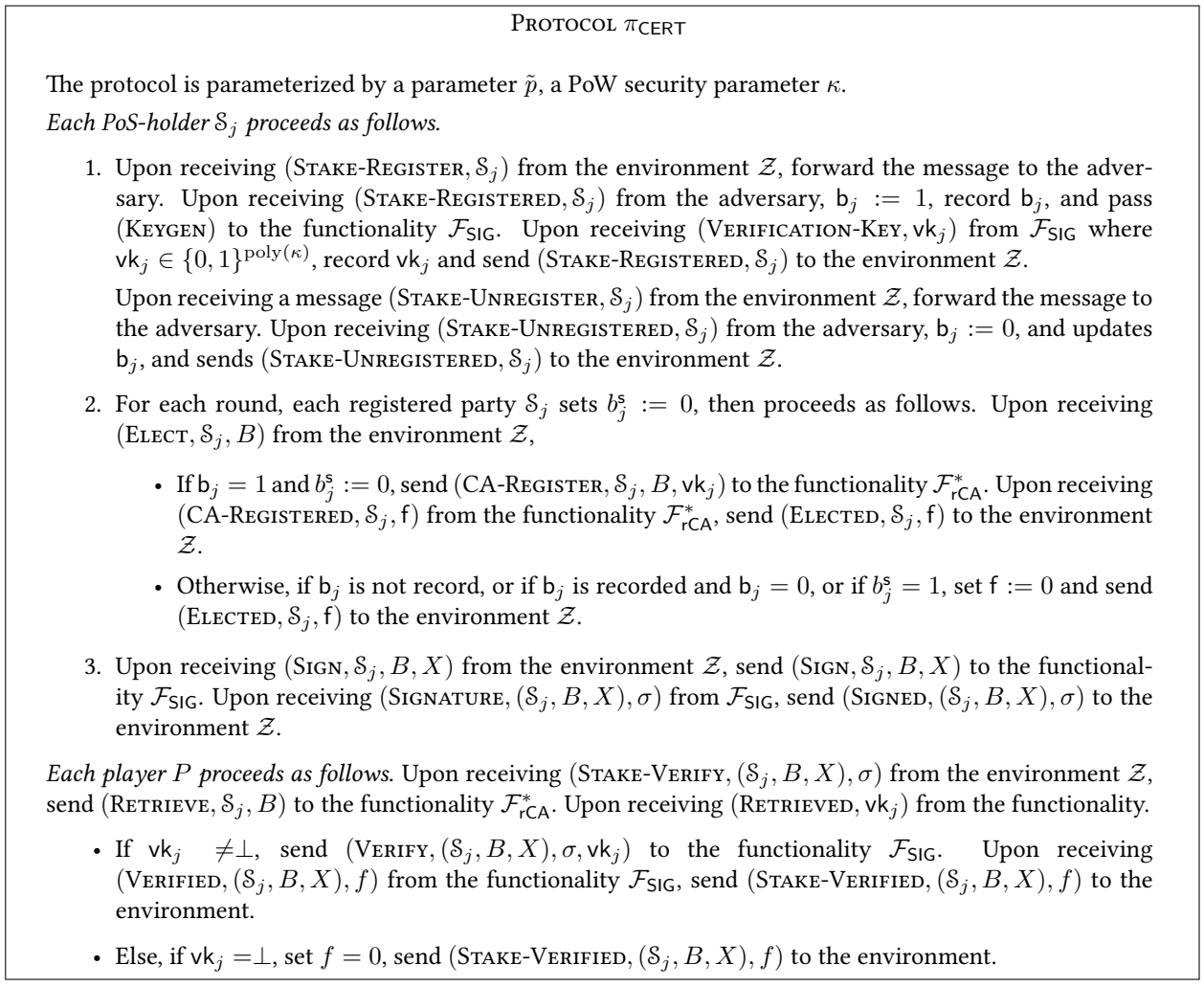
Figure 4: Resource Certification Functionality.

our functionality $\mathcal{F}^*_{\mathsf{rCERT}}$. Here, the functionality $\mathcal{F}^*_{\mathsf{rCERT}}$ can be instantiated in the $\{\mathcal{F}^*_{\mathsf{rCA}}, \mathcal{F}_{\mathsf{SIG}}\}$-hybrid model, where $\mathcal{F}^*_{\mathsf{rCA}}$ can be implemented by a "mature" blockchain and a random oracle. Please refer to Ap-

pendix A for more details about $\mathcal{F}^*_{\mathsf{rCA}}$ and $\mathcal{F}_{\mathsf{SIG}}$; note that $\mathcal{F}^*_{\mathsf{rCA}}$ can be viewed as a "resource" analog of the certificate authority functionality $\mathcal{F}_{\mathsf{CA}}$ in [9]. In Appendix A, we will further show that $\mathcal{F}^*_{\mathsf{rCA}}$ can be instantiated via a mature blockchain and digital signature scheme.

We denote ϕ^*_{rCERT} as the ideal protocol for an ideal functionality $\mathcal{F}^*_{\mathsf{rCERT}}$ and π_{CERT} as protocol in $\{\mathcal{F}^*_{\mathsf{rCA}}, \mathcal{F}_{\mathsf{SIG}}\}$-hybrid model. In the ideal protocol ϕ^*_{rCERT}, the dummy players only forward the messages received from the environment to the functionality $\mathcal{F}^*_{\mathsf{rCERT}}$, and then forward the messages received from the functionality to the environment.

Note that, in π_{CERT}, we represent the mature blockchain by a set of stake identities S which can be derived from the mature blockchain. More precisely, each stakeholder $\mathcal{S}_j$ is assigned with a unique identity vk_j where $(\mathsf{vk}_j, \mathsf{sk}_j) \leftarrow \mathsf{Gen}(1^\kappa)$ with the amount of stake s_j. We define the set of valid stakeholders as $\mathsf{S} = \{\mathcal{S}_j, \mathsf{vk}_j, s_j\}_{j\in\{0,1\}^\kappa}$. Note that, the stake identities could be changed in any round capturing the dynamic stake setting where stake changes over time. We emphasize that our mechanism for leader election is affected by changes in the stake distribution . Without loss of generality, we consider the static stake setting where there are no changes in the stake distribution, and assume all stakeholders have the same amount of stake in π_{CERT} (i.e., $s_j = 1$). The protocol π_{CERT} is described in Figure 5.



PROTOCOL π_{CERT}

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

Each PoS-holder $\mathcal{S}_j$ proceeds as follows.

1. Upon receiving $(\textsc{Stake-Register}, \mathcal{S}_j)$ from the environment $\mathcal{Z}$, forward the message to the adversary. Upon receiving $(\textsc{Stake-Registered}, \mathcal{S}_j)$ from the adversary, $\mathsf{b}_j := 1$, record b_j, and pass $(\textsc{Keygen})$ to the functionality $\mathcal{F}_{\mathsf{SIG}}$. Upon receiving $(\textsc{Verification-Key}, \mathsf{vk}_j)$ from $\mathcal{F}_{\mathsf{SIG}}$ where $\mathsf{vk}_j \in \{0,1\}^{\mathrm{poly}(\kappa)}$, record vk_j and send $(\textsc{Stake-Registered}, \mathcal{S}_j)$ to the environment $\mathcal{Z}$.

   Upon receiving a message $(\textsc{Stake-Unregister}, \mathcal{S}_j)$ from the environment $\mathcal{Z}$, forward the message to the adversary. Upon receiving $(\textsc{Stake-Unregistered}, \mathcal{S}_j)$ from the adversary, $\mathsf{b}_j := 0$, and updates b_j, and sends $(\textsc{Stake-Unregistered}, \mathcal{S}_j)$ to the environment $\mathcal{Z}$.

2. For each round, each registered party $\mathcal{S}_j$ sets $b^{\mathsf{s}}_j := 0$, then proceeds as follows. Upon receiving $(\textsc{Elect}, \mathcal{S}_j, B)$ from the environment $\mathcal{Z}$,

   - If $\mathsf{b}_j = 1$ and $b^{\mathsf{s}}_j := 0$, send $(\text{CA-}\textsc{Register}, \mathcal{S}_j, B, \mathsf{vk}_j)$ to the functionality $\mathcal{F}^*_{\mathsf{rCA}}$. Upon receiving $(\text{CA-}\textsc{Registered}, \mathcal{S}_j, \mathsf{f})$ from the functionality $\mathcal{F}^*_{\mathsf{rCA}}$, send $(\textsc{Elected}, \mathcal{S}_j, \mathsf{f})$ to the environment $\mathcal{Z}$.
   - Otherwise, if b_j is not record, or if b_j is recorded and $\mathsf{b}_j = 0$, or if $b^{\mathsf{s}}_j = 1$, set $\mathsf{f} := 0$ and send $(\textsc{Elected}, \mathcal{S}_j, \mathsf{f})$ to the environment $\mathcal{Z}$.

3. Upon receiving $(\textsc{Sign}, \mathcal{S}_j, B, X)$ from the environment $\mathcal{Z}$, send $(\textsc{Sign}, \mathcal{S}_j, B, X)$ to the functionality $\mathcal{F}_{\mathsf{SIG}}$. Upon receiving $(\textsc{Signature}, (\mathcal{S}_j, B, X), \sigma)$ from $\mathcal{F}_{\mathsf{SIG}}$, send $(\textsc{Signed}, (\mathcal{S}_j, B, X), \sigma)$ to the environment $\mathcal{Z}$.

Each player P proceeds as follows. Upon receiving $(\textsc{Stake-Verify}, (\mathcal{S}_j, B, X), \sigma)$ from the environment $\mathcal{Z}$, send $(\textsc{Retrieve}, \mathcal{S}_j, B)$ to the functionality $\mathcal{F}^*_{\mathsf{rCA}}$. Upon receiving $(\textsc{Retrieved}, \mathsf{vk}_j)$ from the functionality.

- If $\mathsf{vk}_j \neq \perp$, send $(\textsc{Verify}, (\mathcal{S}_j, B, X), \sigma, \mathsf{vk}_j)$ to the functionality $\mathcal{F}_{\mathsf{SIG}}$. Upon receiving $(\textsc{Verified}, (\mathcal{S}_j, B, X), f)$ from the functionality $\mathcal{F}_{\mathsf{SIG}}$, send $(\textsc{Stake-Verified}, (\mathcal{S}_j, B, X), f)$ to the environment.
- Else, if $\mathsf{vk}_j = \perp$, set $f = 0$, send $(\textsc{Stake-Verified}, (\mathcal{S}_j, B, X), f)$ to the environment.

Figure 5: Resource Certification Protocol.

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

of all parties in the execution of π_{CERT} with the adversary $\mathcal{A}$ and an environment $\mathcal{Z}$.

Lemma 2.2. *Consider ϕ^*_{rCERT} described above and π_{CERT} in Figure 5. It holds that the two ensembles $\mathsf{EXEC}^{\mathcal{F}^*_{\mathsf{rCERT}}}_{\phi^*_{\mathsf{rCERT}},\mathcal{S},\mathcal{Z}}$ and $\mathsf{EXEC}^{\mathcal{F}^*_{\mathsf{rCA}},\mathcal{F}_{\mathsf{SIG}}}_{\pi_{\mathsf{CERT}},\mathcal{A},\mathcal{Z}}$ are perfectly indistinguishable.*

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

1. Upon receiving $(\textsc{Stake-Register}, \mathcal{S}_j)$ from $\mathcal{A}$, pass the message to the functionality $\mathcal{F}^*_{\mathsf{rCERT}}$ and receive $(\textsc{Stake-Registered}, \mathcal{S}_j)$ from the functionality. Then output $(\textsc{Stake-Registered}, \mathcal{S}_j)$ to $\mathcal{A}$.

2. Upon receiving $(\textsc{Stake-Unregister}, \mathcal{S}_j)$ from $\mathcal{A}$, pass the message to the functionality $\mathcal{F}^*_{\mathsf{rRO}}$ and receive $(\textsc{Stake-Unregister}, \mathcal{S}_j)$ from the functionality. Then output $(\textsc{Stake-Unregister}, \mathcal{S}_j)$ to $\mathcal{A}$.

3. Upon receiving $(\textsc{Keygen})$ from the adversary $\mathcal{A}$, choose random $\mathsf{vk} \in \{0,1\}^{\mathrm{poly}(\kappa)}$, and then send $(\textsc{Verification-Key}, \mathsf{vk})$ to $\mathcal{A}$.

4. Upon receiving $(\text{CA-}\textsc{Register}, \mathcal{S}_j, B, \mathsf{vk})$ from the adversary $\mathcal{A}$, send $(\textsc{Elect}, \mathcal{S}_j, B)$ to the functionality $\mathcal{F}^*_{\mathsf{rCERT}}$ and then receive $(\textsc{Elected}, \mathcal{S}_j, \mathsf{f})$. Then if $\mathsf{f} = 1$, record $(\mathcal{S}_j, B, \mathsf{vk})$ in table T, and send $(\text{CA-}\textsc{Registered}, \mathcal{S}_j, \mathsf{f})$ to $\mathcal{A}$.

5. Upon receiving $(\textsc{Sign}, \mathcal{S}_j, B, X)$ from $\mathcal{A}$, send $(\textsc{Sign}, \mathcal{S}_j, B, X)$ to the functionality $\mathcal{F}^*_{\mathsf{rCERT}}$ and receive $(\textsc{Signed}, (\mathcal{S}_j, B, X), \sigma)$. Then send $(\textsc{Signature}, (\mathcal{S}_j, B, X), \sigma)$ to $\mathcal{A}$.

6. Upon receiving $(\textsc{Retrieve}, \mathcal{S}_j, B)$ from $\mathcal{A}$,

   - If there is no record of the form $(\mathcal{S}_j, B, \cdot)$, send $(\textsc{Retrieved}, \perp)$ to $\mathcal{A}$.
   - Otherwise, if there is a record of the form $(\mathcal{S}_j, B, \mathsf{vk})$, send $(\textsc{Retrieved}, \mathsf{vk})$ to $\mathcal{A}$.

7. Upon receiving $(\textsc{Verify}, (\mathcal{S}_j, B, X), \sigma, \mathsf{vk}_j)$ from $\mathcal{A}$, send $(\textsc{Stake-Verify}, (\mathcal{S}_j, B, X), \sigma)$ to the functionality $\mathcal{F}^*_{\mathsf{rCERT}}$. Upon receiving $(\textsc{Stake-Verified}, (\mathcal{S}_j, B, X), f)$ from the functionality $\mathcal{F}^*_{\mathsf{rCERT}}$, send $(\textsc{Verified}, (\mathcal{S}_j, B, X), f)$ to $\mathcal{A}$.

In the simulation above, the adversary $\mathcal{S}$ simulate the ideal functionalities $\mathcal{F}_{\mathsf{SIG}}$ and $\mathcal{F}^*_{\mathsf{rCA}}$ for the hybrid protocol π_{CERT}. It is clearly that the two two ensembles $\mathsf{EXEC}^{\mathcal{F}^*_{\mathsf{rCERT}}}_{\phi^*_{\mathsf{rCERT}},\mathcal{S},\mathcal{Z}}$ and $\mathsf{EXEC}^{\mathcal{F}^*_{\mathsf{rCA}},\mathcal{F}_{\mathsf{SIG}}}_{\pi_{\mathsf{CERT}},\mathcal{A},\mathcal{Z}}$ are perfectly indistinguishable since (1) the string vk is randomly choose from a set $\{0,1\}^{\mathrm{poly}(\kappa)}$, and (2) other than that, the adversary $\mathcal{S}$ only need to query the functionality $\mathcal{F}^*_{\mathsf{rCERT}}$ and forward messages (with different headers) to $\mathcal{A}$. This completes the proof.

□

2.4 Blockchain Security Properties

Previously, several fundamental security properties for 1-hop blockchain protocols have been defined: *common prefix property* [17, 33], *chain quality property* [17], and *chain growth property* [22]. Intuitively, the chain growth property for PoS-chains argues that the PoS-chains of honest players should grow linearly to the number of rounds. The common prefix property for PoS-chains indicates the consistency of any two honest PoS-chains except the last κ blocks. The chain quality property for PoS-chains, aims at expressing the number of honest PoS-holders contributions that are contained in a sufficiently long and continuous part of an honest PoS-holder's 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 PoS-holder's 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 growth at the same rate, we mainly focus on the common prefix, chain quality, and chain growth properties for the PoS-chain. Interestingly, the common prefix and chain growth for PoW-chain are explicitly implied from the PoS-chain, but the chain quality for PoW-chain cannot be shown from the PoS-chain since the adversary could control the majority of computing power in our setting. We therefore separately consider the chain growth property for PoW-chain. The definitions for these properties are formally given as follows.

Definition 2.3 (Chain Growth Property for PoS-chain)**.** *The chain growth property $\mathcal{Q}_{\text{cg}}$ states that for any honest PoS-holder $\mathcal{S}$ with the local PoS-chain $\tilde{\mathcal{C}}$ in round r and $\tilde{\mathcal{C}}'$ in round r' where $s = r' - r > 0$, in $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$. It holds that $\mathsf{len}(\tilde{\mathcal{C}}') - \mathsf{len}(\tilde{\mathcal{C}}) \geq g \cdot s$ where g is the growth rate.*

Definition 2.4 (Common Prefix Property for PoS-chain)**.** *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_i and $\mathcal{S}_j$ in round r_j with the local PoS-chains $\tilde{\mathcal{C}}_i, \tilde{\mathcal{C}}_j$, respectively, in $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$ where $i, j \in \{n+1, \ldots, n+\tilde{n}\}, r_i \leq r_j$, 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 2.5 (Chain Quality Property for PoS-chain)**.** *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 $\mathcal{S}$ 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.*

Here, we mainly consider the three properties for PoS-chains. 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.

Definition 2.6 (Chain Quality Property for PoW-chain)**.** *The chain quality property $\mathcal{Q}'_{\text{cq}}$ with parameters $\mu \in \mathbb{R}$ and $\ell \in \mathbb{N}$ states that for any honest PoW-miner $\mathcal{W}$ with PoW-chain $\mathcal{C}$ in $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$, it holds that for large enough ℓ consecutive PoW-blocks of $\mathcal{C}$ the ratio of honest blocks is at least μ'.*

3 Construction

We start by defining blocks, proof-of-work chains, and proof-of-work mechanism in Section 3.1; in Section 3.3, we design a process to choose the best valid chain-pair among a set of chain-pairs; finally, we present our main protocol in Section 3.2.

3.1 Proof-of-Work and Proof-of-Stake

As in the original Bitcoin white paper [29], a proof-of-work (PoW) blockchain is defined as a sequence of ordered blocks. This blockchain is created and maintained by a set of players called *PoW-miners*. In this subsection, for completeness, we first restate the format of a proof-of-work block and blockchain, and then show proof-of-stake related notations.

Proof-of-work. A PoW-block B is a pair of the form $B = \langle h, w \rangle$ where $h \in \{0,1\}^\kappa$ denotes the pointer to the previous block, $w \in \{0,1\}^\kappa$ is a random nonce. A PoW-chain $\mathcal{C}$ 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 $\mathcal{C} \preceq \mathcal{C}'$.

Proof-of-stake. In our cryptocurrency system, along with the proof-of-work blockchain, there is another type of chains called *Proof-of-Stake blockchain*, which is maintained by a set of stakeholders (also called

PoS-holders).

We now introduce the format of a PoS-block. In our system, each valid PoS-blocks is coupled with a valid PoW-block. Based on a given PoW-block B, a stakeholder can produce a PoS-block which is defined as a tuple of the form $\tilde{B} = \langle \mathcal{S}, B, X, \sigma \rangle$. Here, $\mathcal{S} \in \{0,1\}^\kappa$ is the pseudonym of the stakeholder who generates this block, $X \in \{0,1\}^*$ is the payload of the proof-of-stake block $\tilde{B}$ (also denoted as $\mathsf{payload}(\tilde{B})$); and σ is a signature for $\langle \mathcal{S}, B, X \rangle$.

The structure of a PoS-chain is very similar to the PoW-chain, and many notations such as head, length, and subchain can be defined in the same way. 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 $\mathsf{len}(\tilde{\mathcal{C}}) = \ell$, then we have $\mathsf{payload}(\tilde{\mathcal{C}}) = ||_{i=1}^{\ell} \mathsf{payload}(\tilde{B}_i)$.

3.2 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 resistant 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 PoW-miner; however, without loss of generality, we treat the two roles separately. In order to tie them tightly, 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 behavior of PoW-miners and PoS-holders. The PoW and PoS executions vary slightly. On one hand, in the PoW execution, PoW-miners searche for proof-of-work solutions. 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. In general, PoW-miners and PoS-holders collect blockchain information from the broadcast channel, perform some validation and generating blocks, and then share their states with the network through $\mathcal{F}_{\mathsf{NET}}$. Please see Figure 6 for a pictorial illustration of our main protocol.

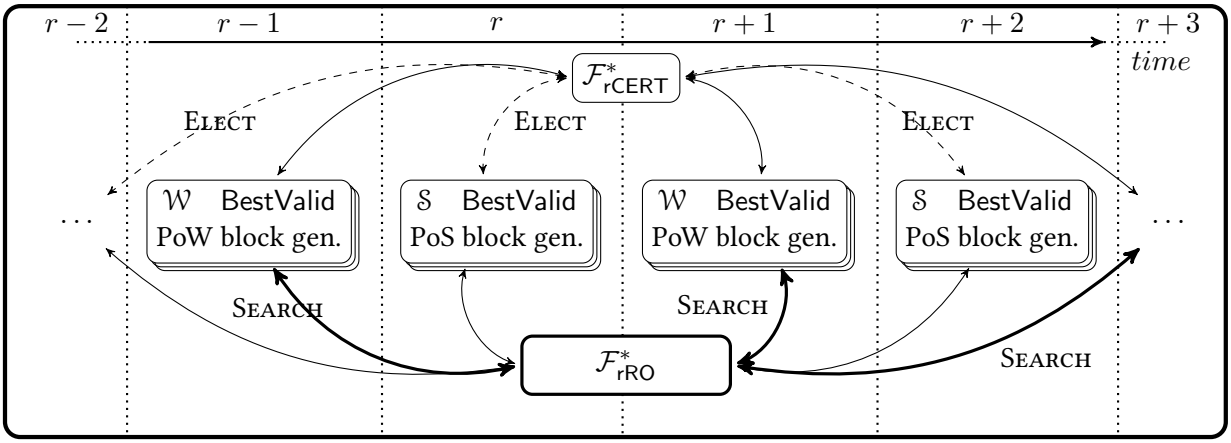


Figure 6: Round Progression in Our Protocol.
This figure depicts the progression of protocol Π through round r. We assume that in each round, each party communicates with two functionalities $\mathcal{F}^*_{\mathsf{rRO}}$ and $\mathcal{F}^*_{\mathsf{rCERT}}$. More specifically, in each round, an online player complete two tasks: (1) determining the best valid chain-pair, (2) attempting to extend either the PoS-chain or PoW-chain of the best chain-pair.

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 [17, 33]. 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 $\mathsf{len}(\mathcal{C}_{\mathsf{init}}) = \mathsf{len}(\tilde{\mathcal{C}}_{\mathsf{init}})$.

PoW-Miner Π^{w}. For each PoW-miner $\mathcal{W}_i$, with an initial local chain-pair $\langle \mathcal{C}_i, \tilde{\mathcal{C}}_i \rangle$, if he is activated by the environment on command (Input-Work, $\mathcal{W}_i$), and received a set of chains $\mathbb{C}$ from the network functionality $\mathcal{F}_{\mathsf{NET}}$. Then he chooses the best valid chain-pair over the chain set $\mathbb{C}$ including his local chain-pair $\langle \mathcal{C}_i, \tilde{\mathcal{C}}_i \rangle$ by executing BestValid (See Figure 8.) We note that, on each PoW-miner's view, the best chain-pair should have two members chains (PoW-chain and PoS-chain) of the *same* length.

After that, he updates his local chain-pair as the best valid chain-pair, and attempts to extend his updated chain-pair. Here, he needs to generate a new PoW-block by outsourcing the functionality $\mathcal{F}^*_{\mathsf{rRO}}$. Note that, he needs to register to the setup before querying. Intuitively, once he registered, he is granted the needed computing power, and the functionality then uses this computing power to mine for a PoW puzzle solution.

More concretely, for each new PoW-block is linked to the heads of $\mathcal{C}_i$ and $\tilde{\mathcal{C}}_i$ by storing a pointer h to *the head of both $\mathcal{C}_i$ and $\tilde{\mathcal{C}}_i$* (note that, $\mathsf{len}(\mathcal{C}_i) = \mathsf{len}(\tilde{\mathcal{C}}_i)$.) Thus, he first sends a regular random oracle query (Compute, $\mathsf{head}(\mathcal{C}_i), \mathsf{head}(\tilde{\mathcal{C}}_i)$) to $\mathcal{F}^*_{\mathsf{rRO}}$, and then receives the digest h. Intuitively, we treat $\mathsf{head}(\tilde{\mathcal{C}}_i)$ as a payload X in the Compute command. He further uses this digest to request $\mathcal{F}^*_{\mathsf{rRO}}$ to mine a PoW puzzle solution in the current round by message (Search, $h, \mathcal{W}_i$). If he receives a message (Searched, w) from $\mathcal{F}^*_{\mathsf{rRO}}$ such that $w \neq \perp$. This implies he found the solution. Sequentially, a new valid block B with the form $B = \langle h, w \rangle$ is attached to $\mathcal{C}_i$ generating a new PoW-chain $\mathcal{C}_i = \mathcal{C}_i \| B$. The player therefore requests $\mathcal{F}_{\mathsf{NET}}$ to broadcast his local chain-pair by message (Broadcast, $\langle \mathcal{C}_i, \tilde{\mathcal{C}}_i \rangle$).

PoS-Holder Π^{s}. This is carried out by PoS-holders. Similarly to Π^{w}s, once activated by the environment on (Input-Stake, $\mathcal{S}_j, X$), where X denotes the payload would be stored in the chain, and received a chain-pair set $\mathbb{C}$ from $\mathcal{F}_{\mathsf{NET}}$, each PoS-holder $\mathcal{S}_j$ finds the best valid chain-pair $\langle \mathcal{C}_{\mathrm{best}}, \tilde{\mathcal{C}}_{\mathrm{best}} \rangle$ through BestValid, and then updates his local chain-pair $\langle \mathcal{C}_j, \tilde{\mathcal{C}}_j \rangle$.

We remark that, we only interested in the case where there is a new PoW-block in the best chain-pair. In other word, we interest in the chain-pair in which the PoW-chain is longer than his corresponding PoS-chain by one block. Consider there is a new PoW-block B in the best PoW-chain. Here, PoS-holders have access to the functionality $\mathcal{F}^*_{\mathsf{rCERT}}$. The players need to register to the functionality $\mathcal{F}^*_{\mathsf{rCERT}}$ before requesting the functionality. Note that only the elected PoS-holders are allowed to generate new PoS-blocks where B is treated as a biased random beacon for the leader election; thus, each registered PoS-holder $\mathcal{S}_j$ can query $\mathcal{F}^*_{\mathsf{rCERT}}$ for leader election by command (Elect, $\mathcal{S}_j, B$). If this PoS-holder is the lucky stakeholder (with probability $\tilde{p}$), he would receive a message (Elected, $\mathcal{S}_j, \mathsf{f}$) from $\mathcal{F}^*_{\mathsf{rCERT}}$ such that $\mathsf{f} = 1$. He therefore requests the functionality to generate a signature for $(\mathcal{S}_j, B, X)$ by command (Sign, $\mathcal{S}_j, B, X$) where B is the new PoW-block and X is the payload from the environment. Once the party received the signature σ from the functionality, he generates a new PoS-block $\tilde{B} := \langle \mathcal{S}_j, B, X, \sigma \rangle$, updates his PoS-chain $\tilde{\mathcal{C}}$ and then broadcasts his local pair to the network.

Please refer to Figure 7 for more details of our main protocol.

3.3 Consensus: 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 [17], 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.

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 $1 \leq i \leq n + \tilde{n}$.

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

1. *Select the best local chain-pair:* Upon receiving $(\text{MESSAGE}, \mathcal{W}_i, \mathbb{C})$ from $\mathcal{F}_{\mathsf{NET}}$, compute $\langle \mathcal{C}_{\text{best}}, \tilde{\mathcal{C}}_{\text{best}} \rangle := \mathsf{BestValid}(\mathbb{C} \cup \langle \mathcal{C}_i, \tilde{\mathcal{C}}_i \rangle, \textit{PoW-miner})$; then set $\mathcal{C}_i := \mathcal{C}_{\text{best}}$ and $\tilde{\mathcal{C}}_i := \tilde{\mathcal{C}}_{\text{best}}$.
2. *Attempt to extend PoW-chain:*
   - *Compute the point to the precious block:* Send $(\text{COMPUTE}, \mathsf{head}(\mathcal{C}_i), \mathsf{head}(\tilde{\mathcal{C}}_i))$ to the ideal functionality $\mathcal{F}^*_{\mathsf{rRO}}$ and receive $(\text{COMPUTED}, h)$ from $\mathcal{F}^*_{\mathsf{rRO}}$.
   - *Search for a puzzle solution:* If $h \neq \perp$, then send $(\text{SEARCH}, \mathcal{W}_i, h)$ to the ideal functionality $\mathcal{F}^*_{\mathsf{rRO}}$, and then receive $(\text{SEARCHED}, \mathcal{W}_i, w)$ from $\mathcal{F}^*_{\mathsf{rRO}}$.
   - *Generate a new PoW-block:* If $w \neq \perp$, set $B := \langle h, w \rangle$ (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 $(\text{BROADCAST}, \langle \mathcal{C}_i, \tilde{\mathcal{C}}_i \rangle)$ to $\mathcal{F}_{\mathsf{NET}}$.

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

PoS-Holder Π^{s}. Each PoS-holder $\mathcal{S}_j$, for $n + 1 \leq j \leq n + \tilde{n}$, proceeds as follows. Without lost of generality, we assume that the PoS-holders already registered to the functionality $\mathcal{F}^*_{\mathsf{rCERT}}$. For each even round, upon receiving message $(\text{INPUT-STAKE}, \mathcal{S}_j, X)$ from the environment $\mathcal{Z}$ where X denotes the block-payload, proceed as follows.

1. *Select the best local chain-pair:* Upon receiving $(\text{MESSAGE}, \mathcal{S}_i, \mathbb{C})$ from $\mathcal{F}_{\mathsf{NET}}$, compute $\langle \mathcal{C}_{\text{best}}, \tilde{\mathcal{C}}_{\text{best}} \rangle := \mathsf{BestValid}(\mathbb{C} \cup \langle \mathcal{C}_j, \tilde{\mathcal{C}}_j \rangle, \textit{PoS-holder})$, then set $\mathcal{C}_j := \mathcal{C}_{\text{best}}$ and $\tilde{\mathcal{C}}_j := \tilde{\mathcal{C}}_{\text{best}}$.
2. *Attempt to extend PoS-chain:* Consider there is a new PoW-block B in $\mathcal{C}_{\text{best}}$.
   - *Stake election:* Send $(\text{ELECT}, \mathcal{S}_j, B)$ to the ideal functionality $\mathcal{F}^*_{\mathsf{rCERT}}$, and receive $(\text{ELECTED}, \mathcal{S}_j, \mathsf{f})$ from $\mathcal{F}^*_{\mathsf{rCERT}}$.
   - *Generate a signature:* If $\mathsf{f} = 1$, send $(\text{SIGN}, \mathcal{S}_j, B, X)$ to the ideal functionality $\mathcal{F}^*_{\mathsf{rCERT}}$, and receive $(\text{SIGNED}, (\mathcal{S}_j, B, X), \sigma)$ from $\mathcal{F}^*_{\mathsf{rCERT}}$.
   - *Generate a new PoS-block:* Set $\tilde{B} := \langle \mathcal{S}_j, B, X, \sigma \rangle$. 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 $(\text{BROADCAST}, \langle \mathcal{C}_j, \tilde{\mathcal{C}}_j \rangle)$ to $\mathcal{F}_{\mathsf{NET}}$.

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

Figure 7: Our main protocol in the $\{\mathcal{F}^*_{\mathsf{rRO}}, \mathcal{F}^*_{\mathsf{rCERT}}, \mathcal{F}_{\mathsf{NET}}\}$-hybrid model with respect to the local process BestValid (See Figure 7).

We emphasize that since each valid PoS-block is tied to a PoW-block, and each PoW-block or PoS-block is valid if their peers are valid. 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-block; thus, PoS-blocks without corresponding PoW-blocks are not valid.

In more detail, BestValid proceeds as follows. On input a set of chains $\mathbb{C}'$ and an index $Type$ where $Type \in \{PoW\text{-}miner, PoS\text{-}holder\}$. For each chain-pair $\langle \mathcal{C}, \tilde{\mathcal{C}} \rangle$, the process validates every PoW-block together with its corresponding PoS-block. However, in every round, there may a new PoW-block without any PoS-block (this only happens in PoS-rounds.) Therefore, if $\mathsf{len}(\mathcal{C}) - 1 = \mathsf{len}(\tilde{\mathcal{C}})$ and $Type = PoS\text{-}holder$ (this means there is a new PoW-block), then it would validate this block separately and then validate every block-pair in the considered chain-pair. Let ℓ-th block of $\mathcal{C}$ is the new block, we need to verify that (1) $\mathcal{C}[\ell]$ is linked to the previous PoW-block and PoS-block correctly, (2) the PoW puzzle is solved properly. To verify the first condition, BestValid queries the resource random oracle $\mathcal{F}^*_{\mathsf{rRO}}$ by command $(\textsc{Compute}, \mathcal{C}[\ell-1], \tilde{\mathcal{C}}[\ell-1])$ to compute the correct pointer. If this pointer is not equal to the pointer stored in $\mathcal{C}[\ell]$, this chain-pair is invalid and will be removed from the chainset $\mathbb{C}'$. To verify the second condition, BestValid queries the resource random oracle $\mathcal{F}^*_{\mathsf{rRO}}$ by command $(\text{RO-}\textsc{Verify}, \mathcal{C}[\ell])$, if it receives $(\text{RO-}\textsc{Verified}, \perp)$ from the functionality, $\mathcal{C}[\ell]$ is invalid making the whole chain-pair chain-pair $\langle \mathcal{C}, \tilde{\mathcal{C}} \rangle$ invalid.

After checking the new block, the process then evaluates every block-pair of the chain-pair $\langle \mathcal{C}, \tilde{\mathcal{C}} \rangle$ sequentially by the following. It first applies the validation predicate $V(\cdot)$ to check the payload in the PoS-chain $\tilde{\mathcal{C}}$. If the payload is valid, then starting from the head of $\mathcal{C}$, for every PoW-block $\mathcal{C}[i]$, for $1 \leq i \leq \mathsf{len}(\mathcal{C})$, in the PoW-chain $\mathcal{C}$, BestValid proceeds as follows. The process first ensures that $\mathcal{C}[i]$ is linked to the previous PoW-block $\mathcal{C}[i-1]$ and PoS-block $\tilde{\mathcal{C}}[i-1]$ correctly, (2) the PoW puzzle is solved properly as in checking the new block above. It then tests if (1) the corresponding PoS-block $\tilde{\mathcal{C}}[i]$ stores the examined PoW-block, and (2) whether the stakeholder who generates PoS-block $\tilde{\mathcal{C}}[i]$ is the valid PoS-holder (by outsourcing $\mathcal{F}^*_{\mathsf{rCERT}}$), and (3) the signature generated by that stakeholder is verifiable (by outsourcing $\mathcal{F}^*_{\mathsf{rCERT}}$).

After the validation, the best valid chain-pair is the one with the longest PoW-chain. Please refer to Figure 8 for more details.

Remark 3.1 (Tie Breaking)**.** *Our protocol primarily deals with length so it makes sense to adopt a simple tie breaking strategy to choose the best chain from two chains of equal length. While there is work that show the advantages of choosing a chain randomly (viz. [16]), we follow the simple strategy considered in [17]; in which the best chain is the one that is lexicographically the smallest.*

4 Security Analysis

Our 2-hop blockchain can be viewed as a natural generalization of Nakamoto's famous 1-hop blockchain. In order to improve the readability, in general, 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., $\hat{\alpha}$.

Our security analysis of the 2-hop blockchain here is inspired and influenced by previous analysis of Nakamoto's 1-hop protocol [17, 33]. Before stating our main theorems, we first recall the important parameters in the first hop of our protocol (i.e., proof-of-work blockchain); these parameters are the same as that in the Nakamoto protocol and we borrow them from the previous analysis [17, 33]. Consider the total number of PoW-miners is n, the portion of malicious computing power is ρ, and a mining hardness parameter p.

- Let $\alpha = 1 - (1-p)^{(1-\rho)n}$ be the probability that some honest PoW-miners mine 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 expect less than one proof of work solution to be produced in each round. 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 resources.

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 $\langle h_\ell, w_\ell \rangle$, send $(\textsc{Compute}, \mathcal{C}[\ell - 1], \tilde{\mathcal{C}}[\ell - 1])$ to $\mathcal{F}^*_{\mathsf{rRO}}$ and then receive $(\textsc{Computed}, h)$. If $h \neq h_\ell$, remove this chain-pair from $\mathbb{C}'$.
   - *Verify the PoW solution in $\mathcal{C}[\ell]$:* Send message $(\text{RO-}\textsc{Verify}, \mathcal{C}[\ell])$ to the functionality $\mathcal{F}^*_{\mathsf{rRO}}$, and then receive $(\text{RO-}\textsc{Verified}, h')$. If $h' = \perp$, 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 $\langle h_i, w_i \rangle$, send $(\textsc{Compute}, \mathcal{C}[i - 1], \tilde{\mathcal{C}}[i - 1])$ to $\mathcal{F}^*_{\mathsf{rRO}}$ and then receive $(\textsc{Computed}, h)$.
     - *Verify the PoW solution in $\mathcal{C}[i]$:* Send message $(\text{RO-}\textsc{Verify}, \mathcal{C}[i])$ to the functionality $\mathcal{F}^*_{\mathsf{rRO}}$, and then receive $(\text{RO-}\textsc{Verified}, h')$.

     If $h_i \neq h$ or $h' = \perp$, 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 $\langle \mathcal{S}, B, X, \sigma \rangle$. Then send message $(\text{Stake-}\textsc{Verify}, (B, X, \mathcal{S}_j), \sigma)$ to the functionality $\mathcal{F}^*_{\mathsf{rCERT}}$. Upon receiving message $(\text{Stake-}\textsc{Verified}, (B, X, \mathcal{S}_j), f)$ from $\mathcal{F}^*_{\mathsf{rCERT}}$, if $f = 0$ or $B \neq \mathcal{C}[i]$, set $\mathbb{b}_2 := 0$. Else, set $\mathbb{b}_2 = 1$.
   - If $\mathbb{b}_1 = 0$ or $\mathbb{b}_2 = 0$, remove this chain-pair from $\mathbb{C}'$.
3. Otherwise, remove this chain-pair from $\mathbb{C}'$.

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

Figure 8: The chain set validation local functionality.

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.

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 poriton of malicious stakes is $\tilde{\rho}$, and the election parameter is $\tilde{p}$. Similarly, if $\tilde{p}\tilde{n} \ll 1$, $\tilde{\alpha}$ can be approximated by $(1 - \tilde{\rho})\tilde{n}\tilde{p}$.

- Let $\tilde{\alpha} = (1 - \tilde{\rho})\tilde{n}\tilde{p}$ be the probability that a PoW-block is mapped to an honest PoS-holder.
- Let $\tilde{\beta} = \tilde{\rho}\tilde{n}\tilde{p}$ be the probability that a PoW-block is mapped to an malicious PoS-holder.

Let $\hat{\alpha} = \alpha\tilde{\alpha}$, $\hat{\beta} = \beta\tilde{\beta}$, $\hat{\gamma} = \frac{\hat{\alpha}}{1+2\Delta\hat{\alpha}}$. Note that $\hat{\gamma}$ 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" collective resource. Note that if $np + \tilde{n}\tilde{p} \ll 1$, then $\hat{\gamma} \approx \hat{\alpha}$ and thus $\hat{\rho} = \frac{\hat{\alpha}}{\hat{\beta}} \approx \frac{(1-\rho)(1-\tilde{\rho})}{\rho\tilde{\rho}}$. We are now ready to state our main theorems.

Theorem 4.1 (Chain Growth for PoS-chain)**.** *For any $\delta > 0$, consider protocol $\Pi = (\Pi^{\mathsf{w}}, \Pi^{\mathsf{s}})$ in Section 3.2. For any honest PoS-holder $\mathcal{S}$ with the local PoS-chain $\tilde{\mathcal{C}}$ in round r and $\tilde{\mathcal{C}}'$ in round r' where $s = r' - r > 0$, in $\mathsf{EXEC}_{(\Pi^{\mathsf{w}}, \Pi^{\mathsf{s}}), \mathcal{A}, \mathcal{Z}}$, the probability that $\mathsf{len}(\tilde{\mathcal{C}}') - \mathsf{len}(\tilde{\mathcal{C}}) \geq g \cdot s$ where $g = (1 - \delta)\hat{\gamma}$ is at least $1 - e^{-\Omega(s)}$.*

Theorem 4.2 (Chain Quality for PoS-chain). *We assume $\hat{\gamma} = \hat{\lambda}(\alpha+\beta)\tilde{\beta}$ and $\hat{\lambda} > 1$. For any $\delta > 0$, consider protocol $\Pi = (\Pi^{\mathsf{w}}, \Pi^{\mathsf{s}})$ in Section 3.2. For any honest PoS-holder $\mathcal{S}$ 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 4.3 (Common Prefix for PoS-chain). *We assume $\hat{\gamma} = \hat{\lambda}\hat{\beta}$ and $\hat{\lambda} > 1$. For any $\delta > 0$, consider protocol $\Pi = (\Pi^{\mathsf{w}}, \Pi^{\mathsf{s}})$ in Section 3.2. 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 $r \le r'$, 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 $1-e^{-\Omega(\kappa)}$.*

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.

4.1 Important Terms

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

Definition 4.4 (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 4.5 (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 4.6. *As discussed in previous section, a PoS-block may be invalid because the delay of the massages on the network. Here the honest stake successful round also only consider that the real valid PoS-block is generated (without discounted).*

Definition 4.7 (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 4.8 (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 player.*

Definition 4.9 (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 4.10 (Strong Safe round). *We say round r is a strong safe round, if no players (both honest and malicious) publish any new PoS-chain in the 2Δ previous rounds of round r to any honest party.*

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 $\mathcal{F}^*_{\mathsf{rCERT}}$, the probability that an online honest stake is selected is $\tilde{\alpha}$. Therefore, the probability that a PoS-round is an honest stake successful round is $\alpha\tilde{\alpha}$ on average.

Following a similar argument, the number of valid malicious blocks that the malicious parties 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 public best chain-pair. At the same time, the malicious stakeholders can append all of the valid malicious PoS-blocks to their (hidden) chain-pair.

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

- **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 and the chosen honest PoS-holder will query the corresponding signature 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 up.

- **Case 2:** When a malicious PoW-miner finds a new PoW-block which is mapped to a malicious PoS-holder we 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 help a hidden chain-pair grow up.
- **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 as the starting point to fork a chain-pair and keep it hidden. That means the malicious PoS-holders can have one free block at the beginning of a hidden PoS-chain of a hidden chain-pair. Any hidden chain-pair can have no more than 1 such kind of block.
- **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 will receive it eventually. If it is accepted, it may help to grow the best public chain-pair.

As the discussion, $\hat{\alpha} = \alpha\tilde{\alpha}$ and $\hat{\beta} = \beta\tilde{\beta}$ is the collective probability of **Case 1** and **Case 2**. We define the collective ratio of the honest and malicious resources.

Before we do the formal analysis we will define public PoS-chain first.

Definition 4.11 (Public PoS-chain)**.** *If a PoS-chain $\tilde{C}$ is known to all honest players, it is a public PoS-chain.*

The following two lemmas show that each honest PoS-holder' local PoS-chains of its local chain-pair will be extended by one block after an honest stake successful round. There may be competition in that round, whatever the result is, the length will grow in that round.

4.2 Analysis with Bounded Delay

We assume that the malicious parties can delay the message arrival with some bounded time through the functionality $\mathcal{F}_{\mathsf{NET}}$ to capture the asynchronous network scenario. The malicious parties can delay any message on the network in no more than Δ rounds (this is guaranteed by $\mathcal{F}_{\mathsf{NET}}$) which we say it is a Δ bounded channel. When an honest 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 stake holders will keep remain divergent for this block. Following a similar analysis, 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 2Δ rounds at most. The intuition is that the delay of messages will decrease 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 introduce the "effective" honest collective resources $\hat{\gamma}$ to capture this intuition.

4.2.1 Hybrid expriments

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 $\mathsf{EXEC}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$ denote the typical execution of $(\Pi^{\mathsf{w}}, \Pi^{\mathsf{s}})$ where

1. The randomness is fixed to σ,
2. Messages of honest players are delayed by $\mathcal{F}_{\mathsf{NET}}$ in at most Δ rounds.

Without loss of generality, we assume that the messages produced in PoW-round are delayed to PoS-round, and messages produced in PoS-round are delayed to PoW-round.

Let $\mathsf{EXEC}^{\nu}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$ denote the execution such that

1. The randomness is fixed to σ,
2. $\mathcal{F}_{\mathsf{NET}}$ delays all messages generated by *honest* PoW-miners at round ν' for all $r \leq \nu' \leq \nu - \Delta$, where $r + s \geq \nu \geq r + \Delta - 1$, in *exact* Δ rounds, during Δ rounds, no *honest* PoW-miners query the functionality $\mathcal{F}^{*}_{\mathsf{rRO}}$ until the messages are delivered,
3. All messages from honest PoS-holders from r to $r + s$ are delayed as in $\mathsf{EXEC}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$ and PoS-holders executes normally as in $\mathsf{EXEC}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$ meaning that the chosen PoS-holders in each PoS-round in both $\mathsf{EXEC}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$ and $\mathsf{EXEC}^{\nu}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$ are the same from round r to $r + s$.
4. After round ν, the messages from honest players are delayed as in $\mathsf{EXEC}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$ and $\mathcal{F}^{*}_{\mathsf{rRO}}$, $\mathcal{F}^{*}_{\mathsf{rCERT}}$ execute as in $\mathsf{EXEC}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$. This means, after round ν, the chosen PoW-miners and PoS-holders in each PoW-round and PoS-round, respectively, in both $\mathsf{EXEC}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$ and $\mathsf{EXEC}^{\nu}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$ are the same.

That is, in this execution, only messages from honest successful rounds before round $\nu - \Delta$ are delayed in Δ rounds.

Let $\mathsf{EXEC}^{\nu}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),\mathcal{A},\mathcal{Z}}(\sigma)$ denote the execution such that

1. The randomness is fixed to σ,
2. $\mathcal{F}_{\mathsf{NET}}$ delays all messages generated by *honest* PoW-miners in round ν' for all $r \geq \nu' \leq r + s - \Delta$, where $r, s > 0$ to the maximum number of Δ, during these Δ rounds, no honest PoW-miners request the functionality $\mathcal{F}^{*}_{\mathsf{rRO}}$ until the messages are delivered,
3. All messages generated by honest PoS-holders, at round ν'' for all $r \geq \nu'' \geq \nu - \Delta$, where $r + s \geq \nu \geq r + \Delta - 1$, to the maximum number of Δ rounds by $\mathcal{F}_{\mathsf{NET}}$, during these Δ rounds, the *honest* PoS-holders do not query the functionality $\mathcal{F}^{*}_{\mathsf{rCERT}}$ until the messages are delivered,
4. After round ν, the messages from honest players are delayed as in $\mathsf{EXEC}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$ and $\mathcal{F}^{*}_{\mathsf{rRO}}$, $\mathcal{F}^{*}_{\mathsf{rCERT}}$ execute as in $\mathsf{EXEC}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$. This means, after round ν, the chosen PoW-miners and PoS-holders in each PoW-round and PoS-round, respectively, in both $\mathsf{EXEC}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$ and $\mathsf{EXEC}^{\nu}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$ are the same.

That is, in this execution, messages from all honest successful rounds in s consecutive rounds are delayed in Δ rounds, and only messages from honest stake successful rounds before round $\nu - \Delta$ are delayed in Δ rounds.

We prove our first lemma showing that the length of PoS-chain of $\mathcal{S}$ at round $r+s$ in the normal execution $\mathsf{EXEC}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$ and the modified execution $\mathsf{EXEC}^{r}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$ where messages of PoW-miners before rounds r are delayed in exact Δ rounds are the same.

Lemma 4.12. *For all $\sigma, r, s > 0$, and for any honest PoS-holder $\mathcal{S}$ at round $r + s$, let $\tilde{\mathcal{C}}^{r+\Delta-1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),r+s,\mathcal{S}}$ denote the PoS-chain of $\mathcal{S}$ at round $r + s$ in the execution $\mathsf{EXEC}^{r+\Delta-1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$, $\tilde{\mathcal{C}}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),r+s,\mathcal{S}}$ denote the PoS-chain of $\mathcal{S}$ at round $r + s$ in the execution $\mathsf{EXEC}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$. We then have*

$$\mathsf{len}(\tilde{\mathcal{C}}^{r+\Delta-1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),r+s,\mathcal{S}}) = \mathsf{len}(\tilde{\mathcal{C}}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),r+s,\mathcal{S}})$$

Proof. We show that $\mathsf{len}(\tilde{\mathcal{C}}^{r+\Delta-1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),r+s,\mathcal{S}}) = \mathsf{len}(\tilde{\mathcal{C}}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),r+s,\mathcal{S}})$. In the execution $\mathsf{EXEC}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$, for any round, messages are delayed in any number of round less than or equal to Δ. In the execution $\mathsf{EXEC}^{r+\Delta-1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}})}(\sigma)$,

from round r to $r+s$, the messages at round ν' for all $r \leq \nu' \leq (r+\Delta-1)-\Delta$, are delayed to the next Δ rounds. That is, there does not exists ν' such that $r \leq \nu' \leq (r+\Delta-1)-\Delta$. This implies there are no rounds from r to $r+s$ that messages are delayed to exact Δ rounds. Therefore, the two executions $\mathsf{EXEC}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$ and $\mathsf{EXEC}^{r+\Delta-1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$ are the same, and then $\mathsf{len}(\tilde{\mathcal{C}}^{r+\Delta-1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),r+s,\mathcal{S}}) = \mathsf{len}(\tilde{\mathcal{C}}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),r+s,\mathcal{S}})$.

□

We are now ready to introduce our lemma to show that the PoS-chain of every honest player will not decrease in length from the execution $\mathsf{EXEC}^{\nu}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$ to the execution $\mathsf{EXEC}^{\nu+1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$ for every fixed randomness σ where messages from honest successful rounds are delayed as described above.

Lemma 4.13. *For all $\sigma, r, s, \nu > 0$, where $r+\Delta-1 \leq \nu \leq r+s$, and for any honest PoS-holder $\mathcal{S}$ at round $r+s$, let $\tilde{\mathcal{C}}^{\nu}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),r+s,\mathcal{S}}$ be the PoS-chain of $\mathcal{S}$ at round $r+s$ in the execution $\mathsf{EXEC}^{\nu}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$, and let $\tilde{\mathcal{C}}^{\nu+1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),r+s,\mathcal{S}}$ be the PoS-chain of $\mathcal{S}$ at round $r+s$ in the execution $\mathsf{EXEC}^{\nu+1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$. We then have*

$$\mathsf{len}(\tilde{\mathcal{C}}^{\nu}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),r+s,\mathcal{S}}) \geq \mathsf{len}(\tilde{\mathcal{C}}^{\nu+1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),r+s,\mathcal{S}}) \tag{1}$$

Proof. We note that if a round is honest successful in $\mathsf{EXEC}^{\nu}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$, it also an honest successful round in $\mathsf{EXEC}^{\nu+1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$. Roughly speaking, the main difference between $\mathsf{EXEC}^{\nu}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$ and $\mathsf{EXEC}^{\nu+1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$ is that, in $\mathsf{EXEC}^{\nu}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$, the messages of PoW-miners in the rounds before $\nu-\Delta$ are delayed, while in $\mathsf{EXEC}^{\nu}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$, the delayed messages are from rounds before $\nu+1-\Delta$. This implies that, from round $\nu+1-\Delta$ to ν, in execution $\mathsf{EXEC}^{\nu+1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$, the messages sent previously from honest PoW-miners to PoS-holders *could* still be in delay, which means no PoW-miners or PoS-holders is allowed to query the functionality $\mathcal{F}^{*}_{\mathsf{rRO}}$ or $\mathcal{F}^{*}_{\mathsf{rCERT}}$, respectively, in round $\nu+1$, whereas, in $\mathsf{EXEC}^{\nu}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$, PoW-miners can query $\mathcal{F}^{*}_{\mathsf{rRO}}$, which means there *may* new PoS-blocks generated in during these rounds. We here take a more careful look at the two executions, and have the following two cases.

If $\mathcal{S}$ is not uncorrupted in any round from $\nu+1-\Delta$ to ν. If $\nu+1-\Delta$ is an *honest successful round*, then in $\mathsf{EXEC}^{\nu+1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$, (1) the messages generated at round $\nu+1-\Delta$ are delayed to round $\nu+1$, (2) all PoW-miners stop querying $\mathcal{F}^{*}_{\mathsf{rRO}}$ until round $\nu+1$. Therefore,

$$\mathsf{len}(\tilde{\mathcal{C}}^{\nu+1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),\nu+1,\mathcal{S}}) = \mathsf{len}(\tilde{\mathcal{C}}^{\nu+1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),\nu+1-\Delta,\mathcal{S}}) + 1$$

On the other hand, in $\mathsf{EXEC}^{\nu}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$, (1) the messages generated at round $\nu+1-\Delta$ are delivered normally as in $\mathsf{EXEC}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$, (2) all PoW-miners and PoS-holders run normally from round $\nu+1+\Delta$ to ν, and there may occur some honest stake successful rounds from $\nu+1-\Delta$ to ν. Therefore,

$$\mathsf{len}(\tilde{\mathcal{C}}^{\nu}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),\nu+1,\mathcal{S}}) \geq \mathsf{len}(\tilde{\mathcal{C}}^{\nu}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),\nu+1-\Delta,\mathcal{S}}) + 1$$

Since the two executions $\mathsf{EXEC}^{\nu}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$ and $\mathsf{EXEC}^{\nu+1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$ till round $\nu+1-\Delta$ and after round $\nu+1$ are the same, we have $\mathsf{len}(\tilde{\mathcal{C}}^{\nu}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),r+s,\mathcal{S}}) \geq \mathsf{len}(\tilde{\mathcal{C}}^{\nu+1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),r+s,\mathcal{S}})$.

It is easily to see that if $\nu+1-\Delta$ is not an honest successful round (e.g., an honest stake successful round, normal PoW-round, PoS-round), the the length of the PoS-chain in two executions are the same.

If $\mathcal{S}$ is uncorrupted in any round from $\nu+1-\Delta$ to ν. If $\nu+1-\Delta$ is an *honest successful round*, then in $\mathsf{EXEC}^{\nu+1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$, all honest parties stop working, while in $\mathsf{EXEC}^{\nu}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$, every party executes normally. If the PoS-holder $\mathcal{S}$ is uncorrupted in round $\nu+1-\Delta \leq \nu' \leq \nu$, he will start with an initial state and receive incoming chain-pair from other parties in the protocol, and then update his local chain-pair with the best one (i.e., the chain-pair with the longest PoW-chain). We here note the difference between $\mathsf{EXEC}^{\nu+1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$ and $\mathsf{EXEC}^{\nu}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$ in this case is that, in $\mathsf{EXEC}^{\nu+1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$, $\mathcal{S}$ will only receive messages from corrupted players since all honest players are stop working, while in $\mathsf{EXEC}^{\nu}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$, $\mathcal{S}$

will receive messages from both honest and corrupted players. Since the PoS-chain in the best chain-pair at round ν' is at least as long as the valid malicious PoS-chain at round ν', the PoS-chain in the best chain-pair at round $r+s$ in $\mathsf{EXEC}^{\nu}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$ is at least as long as the valid malicious PoS-chain at round $r+s$ in $\mathsf{EXEC}^{\nu+1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$.

If $\nu+1-\Delta$ is not an honest successful round, the messages of honest players will not be delayed. Therefore, the length of $\tilde{\mathcal{C}}^{\nu}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),r+s,\mathcal{S}}$ and $\tilde{\mathcal{C}}^{\nu+1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),r+s,\mathcal{S}}$ are the same. This completes the proof.

□

We then prove an important that the length of PoS-chain of $\mathcal{S}$ at round $r+s$ in the two executions $\mathsf{EXEC}^{r+s}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$ and $\mathsf{EXEC}^{r}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),\mathcal{A},\mathcal{Z}}(\sigma)$ are the same.

Lemma 4.14. *For all $\sigma, r, s > 0$, and for any honest PoS-holder $\mathcal{S}$ at round $r+s$, let $\tilde{\mathcal{C}}^{r+s}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),r+s,\mathcal{S}}$ denote the PoS-chain of $\mathcal{S}$ at round $r+s$ in the execution $\mathsf{EXEC}^{r+s}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$, and let $\tilde{\mathcal{C}}^{r+\Delta-1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),r+s,\mathcal{S}}$ denote the PoS-chain of $\mathcal{S}$ at round $r+s$ in the execution $\mathsf{EXEC}^{r+\Delta-1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),\mathcal{A},\mathcal{Z}}(\sigma)$. We then have*

$$\mathsf{len}(\tilde{\mathcal{C}}^{r+s}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),r+s,\mathcal{S}}) = \mathsf{len}(\tilde{\mathcal{C}}^{r+\Delta-1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),r+s,\mathcal{S}})$$

Proof. We demonstrate that $\mathsf{len}(\tilde{\mathcal{C}}^{r+s}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),r+s,\mathcal{S}}) = \mathsf{len}(\tilde{\mathcal{C}}^{r+\Delta-1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),r+s,\mathcal{S}})$. In the execution $\mathsf{EXEC}^{r+s}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$, all messages produced by PoW-miners at round $r \le \nu' < r+s-\Delta$ are delayed in exact Δ rounds, and all messages generated by PoS-holders are delayed normally. By definition of $\mathsf{EXEC}^{r+\Delta-1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),\mathcal{A},\mathcal{Z}}(\sigma)$, all messages generated by PoS-holders are delayed normally and PoS-holders also work normally from round r to $r+s$. Therefore, the two executions $\mathsf{EXEC}^{r+s}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$ and $\mathsf{EXEC}^{r+\Delta-1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),\mathcal{A},\mathcal{Z}}(\sigma)$ are the same and then $\mathsf{len}(\tilde{\mathcal{C}}^{r+s}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),r+s,\mathcal{S}}) = \mathsf{len}(\tilde{\mathcal{C}}^{r+\Delta-1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),r+s,\mathcal{S}})$. This concludes the proof.

□

The following lemma is similar the Lemma 4.13. However, we consider the two executions $\mathsf{EXEC}^{\nu}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),\mathcal{A},\mathcal{Z}}(\sigma)$ and $\mathsf{EXEC}^{\nu+1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),\mathcal{A},\mathcal{Z}}(\sigma)$ where messages from all honest successful rounds in s consecutive rounds are delayed in exact Δ rounds and messages generated in some honest stake successful rounds are delayed in Δ rounds.

Lemma 4.15. *For all $\sigma, r, s, \nu > 0$, where $r+\Delta-1 \le \nu \le r+s$, and for any honest PoS-holder $\mathcal{S}$ at round $r+s$, let $\tilde{\mathcal{C}}^{\nu}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),r+s,\mathcal{S}}$ be the PoS-chain of $\mathcal{S}$ at round $r+s$, in the execution $\mathsf{EXEC}^{\nu}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),\mathcal{A},\mathcal{Z}}(\sigma)$, and let $\tilde{\mathcal{C}}^{\nu+1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),r+s,\mathcal{S}}$ be the PoS-chain of $\mathcal{S}$ at round $r+s$ in the execution $\mathsf{EXEC}^{\nu+1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),\mathcal{A},\mathcal{Z}}(\sigma)$. We then have*

$$\mathsf{len}(\tilde{\mathcal{C}}^{\nu}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),r+s,\mathcal{S}}) \ge \mathsf{len}(\tilde{\mathcal{C}}^{\nu+1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),r+s,\mathcal{S}}) \tag{2}$$

Proof. Note that if a round is an honest successful or honest stake successful round in $\mathsf{EXEC}^{\nu}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),\mathcal{A},\mathcal{Z}}(\sigma)$, it is also an honest successful or honest stake successful round in $\mathsf{EXEC}^{\nu+1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),\mathcal{A},\mathcal{Z}}(\sigma)$. The main difference between $\mathsf{EXEC}^{\nu}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),\mathcal{A},\mathcal{Z}}(\sigma)$ and $\mathsf{EXEC}^{\nu+1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),\mathcal{A},\mathcal{Z}}(\sigma)$ is the same as the difference between $\mathsf{EXEC}^{\nu}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$ and $\mathsf{EXEC}^{\nu+1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$ with an addition is that messages of PoS-holder are also delayed. We also have two cases here.

If $\mathcal{S}$ is not uncorrupted in any round from $\nu+1-\Delta$ to ν. If $\nu+1-\Delta$ is an *honest stake successful round*, then in $\mathsf{EXEC}^{\nu+1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),\mathcal{A},\mathcal{Z}}(\sigma)$, (1) the messages generated at round $\nu+1-\Delta$ are delayed to round $\nu+1$, (2) all PoW-miners and PoS-holders stop querying $\mathcal{F}^{*}_{\mathsf{rRO}}$ and $\mathcal{F}^{*}_{\mathsf{rCERT}}$, respectively, until round $\nu+1$. Therefore,

If $\mathcal{S}$ is not the PoS-holder who produced the new PoS-block at round $\nu+1-\Delta$,

$$\mathsf{len}(\tilde{\mathcal{C}}^{\nu+1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),\nu+1,\mathcal{S}}) = \mathsf{len}(\tilde{\mathcal{C}}^{\nu+1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),\nu+1-\Delta,\mathcal{S}}) + 1$$

If $\mathcal{S}$ is the PoS-holder who produced the new PoS-block at round $\nu+1-\Delta$,

$$\mathsf{len}(\tilde{\mathcal{C}}^{\nu+1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),\nu+1,\mathcal{S}}) = \mathsf{len}(\tilde{\mathcal{C}}^{\nu+1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),\nu+1-\Delta,\mathcal{S}})$$

On the other hand, in $\mathsf{EXEC}^{\nu}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$, (1) the messages generated at round $\nu+1-\Delta$ are delivered normally as in $\mathsf{EXEC}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$, (2) all PoW-miners and PoS-holders execute normally from round $\nu+1+\Delta$ to ν. Therefore,

If $\mathcal{S}$ is not the PoS-holder who produced the new PoS-block at round $\nu+1-\Delta$,

$$\mathsf{len}(\tilde{\mathcal{C}}^{\nu,(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta})}_{\nu+1,\mathcal{S}}) \geq \mathsf{len}(\tilde{\mathcal{C}}^{\nu,(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta})}_{\nu+1-\Delta,\mathcal{S}}) + 1$$

If $\mathcal{S}$ is the PoS-holder who produced the new PoS-block at round $\nu+1-\Delta$,

$$\mathsf{len}(\tilde{\mathcal{C}}^{\nu}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),\nu+1,\mathcal{S}}) \geq \mathsf{len}(\tilde{\mathcal{C}}^{\nu}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),\nu+1-\Delta,\mathcal{S}})$$

Since the two executions $\mathsf{EXEC}^{\nu}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),\mathcal{A},\mathcal{Z}}(\sigma)$ and $\mathsf{EXEC}^{\nu+1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),\mathcal{A},\mathcal{Z}}(\sigma)$ till round $\nu+1-\Delta$ and after round $\nu+1$ are the same, we have $\mathsf{len}(\tilde{\mathcal{C}}^{\nu}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),r+s,\mathcal{S}}) \geq \mathsf{len}(\tilde{\mathcal{C}}^{\nu+1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),r+s,\mathcal{S}})$.

If $\nu+1-\Delta$ is not an honest successful round or an honest stake successful round (an honest successful round, normal PoS-round, or PoW-round), the length of the PoS-chain of $\mathcal{S}$ in the two executions are the same. Thus, $\mathsf{len}(\tilde{\mathcal{C}}^{\nu}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),r+s,\mathcal{S}}) = \mathsf{len}(\tilde{\mathcal{C}}^{\nu+1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),r+s,\mathcal{S}})$.

If $\mathcal{S}$ is uncorrupted in any round from $\nu+1-\Delta$ to ν. As discussed, if $\nu+1-\Delta$ is an *honest stake successful round*, then all messages of honest PoS-holders in $\mathsf{EXEC}^{\nu+1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),\mathcal{A},\mathcal{Z}}(\sigma)$ are delayed Δ rounds and all players stop running, while messages in $\mathsf{EXEC}^{\nu}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),\mathcal{A},\mathcal{Z}}(\sigma)$ from round $\nu+1-\Delta$ to ν are delayed normally by $\mathcal{F}_{\mathsf{NET}}$ and all players work normally. Therefore, the uncorrupted PoS-holder will start with an initial chain-pair and receive incoming chain-pairs from all players in $\mathsf{EXEC}^{\nu}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),\mathcal{A},\mathcal{Z}}(\sigma)$, whereas $\mathcal{S}$ will only receive the incoming chain-pairs from the malicious players. The PoS-holder will therefore choose the best chain-pair. Since the PoS-chain in the best chain-pair at the uncorrupted round is not shorter than all other PoS-chains including the malicious PoS-chain in the execution. We therefore have $\mathsf{len}(\tilde{\mathcal{C}}^{\nu}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),r+s,\mathcal{S}}) \geq \mathsf{len}(\tilde{\mathcal{C}}^{\nu+1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),r+s,\mathcal{S}})$.

If $\nu+1-\Delta$ is an *honest successful round*, in both executions, all messages of honest PoW-miners at round $\nu+1-\Delta$ are delayed to the maximum of Δ rounds. Thus, $\mathsf{len}(\tilde{\mathcal{C}}^{\nu}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),r+s,\mathcal{S}}) = \mathsf{len}(\tilde{\mathcal{C}}^{\nu+1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),r+s,\mathcal{S}})$.

If $\nu+1-\Delta$ is not an honest successful round or honest stake successful round, no messages of honest parties are delayed from round $\nu+1-\Delta$ to ν. Therefore, $\mathsf{len}(\tilde{\mathcal{C}}^{\nu}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),r+s,\mathcal{S}}) = \mathsf{len}(\tilde{\mathcal{C}}^{\nu+1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),r+s,\mathcal{S}})$.

This concludes the proof.

□

The following lemma shows that the PoS-chain of every honest PoS-holder cannot decrease in length if we maximally delay messages from honest parties in every honest successful round or honest stake successful round during s consecutive rounds, and all honest PoW-miners, PoS-holders during this delay stop requesting the functionalities. The lemma is formally given as follows.

Lemma 4.16. *For all $\sigma, r, s > 0$, and for any honest PoS-holder $\mathcal{S}$ at round $r+s$, let $\tilde{\mathcal{C}}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),r+s,\mathcal{S}}$ be the PoS-chain of $\mathcal{S}$ at round $r+s$ in the execution $\mathsf{EXEC}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}(\sigma)$ and $\tilde{\mathcal{C}}^{r+s}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),r+s,\mathcal{S}}$ be the PoS-chain of $\mathcal{S}$ at round $r+s$ in the execution $\mathsf{EXEC}^{r+s}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),\mathcal{A},\mathcal{Z}}(\sigma)$. We then have*

$$\mathsf{len}(\tilde{\mathcal{C}}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),r+s,\mathcal{S}}) \geq \mathsf{len}(\tilde{\mathcal{C}}^{r+s}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),r+s,\mathcal{S}})$$

Proof. By Lemmas 4.12, 4.13, 4.15, and 4.14, we have

$$\begin{aligned}\mathsf{len}(\tilde{\mathcal{C}}_{\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),r+s,\mathcal{S}}) = \mathsf{len}(\tilde{\mathcal{C}}^{r+\Delta-1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),r+s,\mathcal{S}}) &\geq \mathsf{len}(\tilde{\mathcal{C}}^{r+\Delta}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),r+s,\mathcal{S}}) \geq \ldots \geq \mathsf{len}(\tilde{\mathcal{C}}^{r+s}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}),r+s,\mathcal{S}}) = \mathsf{len}(\tilde{\mathcal{C}}^{r+\Delta-1}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),r+s,\mathcal{S}}) \\ &\geq \mathsf{len}(\tilde{\mathcal{C}}^{r+\Delta}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),r+s,\mathcal{S}}) \geq \ldots \geq \mathsf{len}(\tilde{\mathcal{C}}^{r+s}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),r+s,\mathcal{S}})\end{aligned} \tag{3}$$

This completes the proof. □

4.2.2 Analysis in the worst delay setting

Let view_r denote the view at round r for any execution of $\Pi = (\Pi^{\mathsf{w}}, \Pi^{\mathsf{s}})$ for any randomness σ and $s, r > 0$. Then, let $\mathsf{len}(\mathsf{view}_r)$ denote the length of the best public PoS-chain chain at round r in view_r for any randomness σ and $s, r > 0$.

Note that, we denote $\mathsf{len}(\tilde{\mathcal{C}}^{r+s}_{(\Pi^{\mathsf{w}}_\Delta, \Pi^{\mathsf{s}}_\Delta), r+s, \mathcal{S}})$ as the length of the PoS-chain of $\mathcal{S}$ at round $r+s$ in the execution $\mathsf{EXEC}^{r+s}_{(\Pi^{\mathsf{w}}_\Delta, \Pi^{\mathsf{s}}_\Delta), \mathcal{A}, \mathcal{Z}}$. If the context (i.e., execution) is clear, we write $\mathsf{len}(\tilde{\mathcal{C}}_{r+s, \mathcal{S}})$ instead of $\mathsf{len}(\tilde{\mathcal{C}}^{r+s}_{(\Pi^{\mathsf{w}}_\Delta, \Pi^{\mathsf{s}}_\Delta), r+s, \mathcal{S}})$.

Note that the messages from PoS-holders to PoW-miners are delayed exactly Δ rounds and no PoS-holders work, and then message from PoW-miners are delayed exactly Δ rounds no PoS-holders execute in these Δ rounds. Thus, for a message to be delivered from a PoS-holder to another PoS-holder, it takes exactly 2Δ rounds in $\mathsf{EXEC}^{r+s}_{(\Pi^{\mathsf{w}}_\Delta, \Pi^{\mathsf{s}}_\Delta), \mathcal{A}, \mathcal{Z}}$. The following lemma captures this.

Lemma 4.17. *Consider* $\mathsf{EXEC}^{r+s}_{(\Pi^{\mathsf{w}}_\Delta, \Pi^{\mathsf{s}}_\Delta), \mathcal{A}, \mathcal{Z}}(\sigma)$. *Let* $\alpha, \beta, \tilde{\alpha}, \tilde{\beta} > 0$, $\alpha + \beta < 1$, $\tilde{\alpha} + \tilde{\beta} < 1$, *and* $\hat{\alpha} = \hat{\lambda}\hat{\beta}$ *where* $\hat{\lambda} \in (1, \infty)$. *We assume that the malicious parties can delay any message for* Δ *rounds at most. Let* $\hat{\gamma}$ *be the actual probability that a round is honest successful stake round, we have* $\hat{\gamma} = \frac{\hat{\alpha}}{1+2\Delta\hat{\alpha}}$.

Proof. Consider the case $\mathsf{EXEC}_{(\Pi^{\mathsf{w}}, \Pi^{\mathsf{s}}), \mathcal{A}, \mathcal{Z}}(\sigma)$ without any delay, the expected number of honest stake successful rounds from round r to $r+s$ is $\hat{\alpha}s$. If we assume the malicious parties can delay any message for Δ at most, each *honest stake successful round* would lead to 2Δ rounds delay where the honest player will wast computing power because they don't get the recent successful block. Now, assume there are actual c honest stake successful rounds from round r to $r+s$ in $\mathsf{EXEC}^{r+s}_{(\Pi^{\mathsf{w}}_\Delta, \Pi^{\mathsf{s}}_\Delta), \mathcal{A}, \mathcal{Z}}(\sigma)$ with Δ assumption. We then have, $\hat{\alpha}(s - 2\Delta c) = c$. This implies that $c = \frac{\hat{\alpha}s}{1+2\Delta\hat{\alpha}} = \hat{\gamma}s$ where $\hat{\gamma} = \frac{\hat{\alpha}}{1+2\Delta\hat{\alpha}}$. □

The following lemma demonstrates that each stake successful round would contribute one PoS-block to the best chain-pair after Δ rounds in an execution of Π.

Lemma 4.18. *For any honest stake successful round* ν, $\mathsf{len}(\mathsf{view}_\nu) - \mathsf{len}(\mathsf{view}_{\nu-\Delta}) \geq 1$.

Proof. As the definition in 4.5. There is at least one PoS-holder produces a PoS-block in round ν. Let $\tilde{\mathcal{C}}_{\nu-\Delta, \mathcal{S}}$ be the PoS-chain that is extended by the PoS-holder at round $\nu - \Delta$. We have $\mathsf{len}(\tilde{\mathcal{C}}_{\nu-\Delta, \mathcal{S}}) \geq \mathsf{len}(\mathsf{view}_{\nu-\Delta})$. Let $\tilde{\mathcal{C}}_{\nu, \mathcal{S}}$ be the PoS-chain that is extended by the PoS-holder at round ν. At the end of round ν we have $\mathsf{len}(\tilde{\mathcal{C}}_{\nu, \mathcal{S}}) = \mathsf{len}(\tilde{\mathcal{C}}_{\nu-\Delta, \mathcal{S}}) + 1$. The honest miner will broad cast $\tilde{\mathcal{C}}_{\nu, \mathcal{S}}$ to all of the parties, so we have $\mathsf{len}(\mathsf{view}_\nu) \geq \mathsf{len}(\tilde{\mathcal{C}}_{\nu, \mathcal{S}})$. Put them together, we have $\mathsf{len}(\mathsf{view}_\nu) - \mathsf{len}(\mathsf{view}_{\nu-\Delta}) \geq 1$.

□

Corollary 4.19. *Suppose there are* t *honest stake successful rounds from round* r *to round* $r+s$, *it holds that* $\mathsf{len}(\mathsf{view}_{r+s}) - \mathsf{len}(\mathsf{view}_r) \geq t$.

Proof. Let r_i be honest stake successful round where $r < \nu_i < r+s$ and $1 \leq i \leq t$. From lemma 4.18, we have $\mathsf{len}(\mathsf{view}_{\nu_i}) - \mathsf{len}(\mathsf{view}_{\nu_i-\Delta}) \geq 1$.

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

We also need to measure the ability that how fast the malicious parties can extend a hidden chain-pair. Let $\mathsf{hid}(\mathsf{view}_r)$ denote the length of the longest hidden PoS-chain chain which is controlled by malicious parties at round r in view.

Lemma 4.20. *Let* h *be the number of valid malicious PoW-blocks that are generated in round* r, $\mathsf{hid}(\mathsf{view}_r) - \mathsf{hid}(\mathsf{view}_{r-1}) \leq h$.

Proof. No honest PoW-miner will extend a hidden chain-pair with a PoW-block. Therefore, the PoW-chain of this chain-pair will extend by one new PoW-block B only by a malicious PoW-miner. If the mapped stake-holder is honest, then the PoS-holder will not extend the corresponding PoS-chain unless that the chain-pair is public.

Put them together, only valid malicious PoW-blocks will extend hidden chain-pair, and each will contribute a new PoS-block which makes the length of the PoS-chain increase by at most 1. Hence, $\mathsf{hid}(\mathsf{view}_r) - \mathsf{hid}(\mathsf{view}_{r-1}) \leq h$. □

4.3 Achieving the Chain Growth Property

We here demonstrate that our protocol satisfies the chain growth property (Definition 2.3). The concrete statement to be proved can be found in Theorem 4.1.

Lemma 4.21. *Consider* $\mathsf{EXEC}^{r+s}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),\mathcal{A},\mathcal{Z}}$. *For any* $\delta > 0$, *during any consecutive* s *rounds, the number of honest successful rounds is* $(1-\delta)\hat{\gamma}s$ *with probability at least* $1 - e^{-\Omega(s)}$.

Proof. In any consecutive s rounds, the number of honest stake successful rounds is $\hat{\gamma}s$ on average. Let X'' be the number of honest stake successful rounds in the s consecutive rounds in the execution $\mathsf{EXEC}^{r+s}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),\mathcal{A},\mathcal{Z}}$. By Chernoff bound, we have $\Pr[X'' \leq (1-\delta)\hat{\gamma}s] \leq e^{-\delta^2\hat{\gamma}s/2}$.

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

We first prove that our blockchain protocol achieves the chain growth property in the execution $\mathsf{EXEC}^{r+s}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),\mathcal{A},\mathcal{Z}}$ before moving to the main theorem.

Lemma 4.22. *Consider* $\mathsf{EXEC}^{r+s}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),\mathcal{A},\mathcal{Z}}$. *For any* $\delta > 0$, *and for any honest PoS-holder* $\mathcal{S}$ *with the best PoS-chain* $\tilde{\mathcal{C}}_{r,\mathcal{S}}$ *and* $\tilde{\mathcal{C}}_{r',\mathcal{S}}$ *in round* r' *and* r' *respectively where* $s = r' - r > 0$, *the probability that* $\mathsf{len}(\tilde{\mathcal{C}}_{r',\mathcal{S}}) - \mathsf{len}(\tilde{\mathcal{C}}_{r,\mathcal{S}}) \geq g \cdot s$ *where* $g = (1-\delta)\hat{\gamma}$ *is at least* $1 - e^{-\Omega(s)}$.

Proof. Let PoS-chain $\tilde{\mathcal{C}}$ be the best valid PoS-chain of $\mathcal{S}$ in round r. We have $\mathsf{len}(\tilde{\mathcal{C}}_{r,\mathcal{S}}) \leq \mathsf{len}(\mathsf{view}_r)$. From Lemma 4.21, in any consecutive s rounds the number of honest successful round is more than $(1-\delta)\gamma s$ with low error probability. Together with Lemma 4.18 and Corollary 4.19, we have $\mathsf{len}(\mathsf{view}_{r'}) - \mathsf{len}(\mathsf{view}_r) \geq (1-\delta)\hat{\gamma}s$.

$\tilde{\mathcal{C}}_{r',\mathcal{S}}$ is the best valid PoS-chain accepted by an honest PoS-holder $\mathcal{S}$ at the end of round r'. We have $\mathsf{len}(\tilde{\mathcal{C}}_{r',\mathcal{S}}) \geq \mathsf{len}(\mathsf{view}_{r'})$. Put them together, $\mathsf{len}(\tilde{\mathcal{C}}_{r',\mathcal{S}}) - \mathsf{len}(\tilde{\mathcal{C}}_{r,\mathcal{S}}) \geq \mathsf{len}(\mathsf{view}_{r+s}) - \mathsf{len}(\mathsf{view}_r) \geq (1-\delta)\hat{\gamma}s$ with probability $1 - e^{-\Omega(s)}$. The corresponding growth rate is $g = (1-\delta)\hat{\gamma}$. □

Reminder of Theorem 4.1. *For any* $\delta > 0$, *consider protocol* $\Pi = (\Pi^{\mathsf{w}}, \Pi^{\mathsf{s}})$ *in Section 3.2. For any honest PoS-holder* $\mathcal{S}$ *with the local PoS-chain* $\tilde{\mathcal{C}}$ *in round* r *and* $\tilde{\mathcal{C}}'$ *in round* r' *where* $s = r' - r > 0$, *in* $\mathsf{EXEC}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}$, *the probability that* $\mathsf{len}(\tilde{\mathcal{C}}') - \mathsf{len}(\tilde{\mathcal{C}}) \geq g \cdot s$ *where* $g = (1-\delta)\hat{\gamma}$ *is at least* $1 - e^{-\Omega(s)}$.

Proof. From Lemma 4.22, we have $\Pr[\mathsf{len}(\tilde{\mathcal{C}}_{r',\mathcal{S}}) \geq \mathsf{len}(\tilde{\mathcal{C}}_{r,\mathcal{S}}) + g \cdot s] \geq 1 - e^{-\Omega(s)}$ where honest PoS-holder $\mathcal{S}$ has the local PoS-chain $\tilde{\mathcal{C}}_{r,\mathcal{S}}$ in round r and PoS-chain $\tilde{\mathcal{C}}_{r',\mathcal{S}}$ in round r' where $s = r' - r > 0$, in $\mathsf{EXEC}^{r+s}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),\mathcal{A},\mathcal{Z}}$.

We now turn to the chain growth property in $\mathsf{EXEC}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}$. Let $\tilde{\mathcal{C}}$ and $\tilde{\mathcal{C}}'$ be the local PoS-chain of $\mathcal{S}$ in round r and r', respectively, where $s = r' - r > 0$ in the execution $\mathsf{EXEC}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}$. By Lemma 4.16, it follows that,

$$\Pr[\mathsf{len}(\tilde{\mathcal{C}}) \geq \mathsf{len}(\tilde{\mathcal{C}}') + g \cdot s] \geq \Pr[\mathsf{len}(\tilde{\mathcal{C}}_{r',\mathcal{S}}) \geq \mathsf{len}(\tilde{\mathcal{C}}_{r,\mathcal{S}}) + g \cdot s] \geq 1 - e^{-\Omega(s)} \tag{4}$$

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

4.4 Achieving the Chain Quality Property

The chain-quality property (Definition 2.5) 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 resource. Finally, we demonstrate that the ratio of honest PoS-blocks in an honest player's PoS-chain are not 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 4.23. *For any $\delta > 0$, given $\Pi = (\Pi_{\mathsf{w}}, \Pi_{\mathsf{s}})$, let Z be the number of rounds which generate ℓ blocks, we then have $\Pr[Z = \Omega(\ell)] > 1 - e^{-\Omega(l)}$.*

Proof. Put 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 Z be the number of rounds which generate ℓ PoS-blocks. For any $\delta > 0$, by using Chernoff bounds, $\Pr[Z \leq (1-\delta)\frac{\ell}{\hat{\alpha}+\hat{\beta}}] \leq e^{-\delta^2 \frac{\ell}{\hat{\alpha}+\hat{\beta}}/3}$

That is, ℓ blocks will consume $\Omega(\ell)$ rounds to extend the blocks and $\Pr[Z = \Omega(\ell)] > 1 - e^{-\Omega(\ell)}$. This completes the proof. □

Next we will prove that our blockchain protocol achieves the chain quality property in the execution $\mathsf{EXEC}^{r+s}_{(\Pi^{\mathsf{w}}_{\Delta}, \Pi^{\mathsf{s}}_{\Delta}), \mathcal{A}, \mathcal{Z}}$ before moving to the main theorem.

Lemma 4.24. *Consider $\mathsf{EXEC}^{r+s}_{(\Pi^{\mathsf{w}}_{\Delta}, \Pi^{\mathsf{s}}_{\Delta}), \mathcal{A}, \mathcal{Z}}$. We assume $\hat{\gamma} = \hat{\lambda}(\alpha + \beta)\tilde{\beta}$ and $\hat{\lambda} > 1$. For any $\delta > 0$, consider protocol $\Pi = (\Pi^{\mathsf{w}}_{\Delta}, \Pi^{\mathsf{s}}_{\Delta})$ in Section 3.2. For any honest PoS-holder $\mathcal{S}$ with PoS-chain $\tilde{\mathcal{C}}$, the probability that, for large enough ℓ consecutive PoS-blocks of $\tilde{\mathcal{C}}_{r+s,\mathcal{S}}$ which are generated from round r to $r+s$, 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 r is the round where the first PoS-block in the ℓ consecutive blocks of $\tilde{\mathcal{C}}$ is produced. We will show the chain quality property in $\mathsf{EXEC}^{r+s}_{(\Pi^{\mathsf{w}}_{\Delta}, \Pi^{\mathsf{s}}_{\Delta}), \mathcal{A}, \mathcal{Z}}$. As the proof in Lemma 4.23, given any concussive l blocks for the view of the stakeholder $\mathcal{S}$, ℓ blocks will consume $\Omega(\ell)$ rounds with probability $1 - e^{-\Omega(\ell)}$. We use s to denote the number of rounds that the l blocks consumed. We have $s = \Omega(l)$ with probability $1 - e^{-\Omega(l)}$.

Let X'' be the number of honest stake successful rounds in s rounds in $\mathsf{EXEC}^{r+s}_{(\Pi^{\mathsf{w}}_{\Delta}, \Pi^{\mathsf{s}}_{\Delta}), \mathcal{A}, \mathcal{Z}}$. By Chernoff bound, we have

$$\Pr[X'' > (1-\delta)\hat{\gamma} \cdot s] > 1 - e^{-\Omega(s)} > 1 - e^{-\Omega(\ell)}$$

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

$$\Pr[Y < (1+\delta)\beta\tilde{\beta} \cdot s] > 1 - e^{-\Omega(s)} > 1 - e^{-\Omega(\ell)}$$

Further more, let Z be the number of blocks of case 3 in s round. By Chernoff bound, we have

$$\Pr[Z < (1+\delta)\alpha\tilde{\beta} \cdot s] > 1 - e^{-\Omega(s)} > 1 - e^{-\Omega(\ell)}$$

From Lemma 4.18 and Corollary 4.19, we have $\ell \geq X''$. From Lemma 4.20, the malicious parties will contribute $Y + Z$ PoS-blocks at most in the s rounds. Even if the malicious parties win all of the competitions in the block comparisons for each of his blocks, the ℓ blocks will remain $\ell - Y - Z$ blocks from the honest parties at least. We then have, in $\mathsf{EXEC}^{r+s}_{(\Pi^{\mathsf{w}}_{\Delta}, \Pi^{\mathsf{s}}_{\Delta}), \mathcal{A}, \mathcal{Z}}$, $\mu \geq \frac{\ell - Y - Z}{\ell} = 1 - \frac{Y+Z}{\ell} \geq 1 - (1+\delta)\frac{(\alpha+\beta)\tilde{\beta}}{\hat{\gamma}}$. This completes the proof. □

Reminder of Theorem 4.2. *We assume $\hat{\gamma} = \hat{\lambda}(\alpha+\beta)\tilde{\beta}$ and $\hat{\lambda} > 1$. For any $\delta > 0$, consider protocol $\Pi = (\Pi^{\mathsf{w}}, \Pi^{\mathsf{s}})$ in Section 3.2. For any honest PoS-holder $\mathcal{S}$ 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. From Lemma 4.24, in $\mathsf{EXEC}^{r+s}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),\mathcal{A},\mathcal{Z}}$, the ratio of honest PoS-blocks in s consecutive with ℓ PoS-blocks is $\mu \geq 1-(1+\delta)\frac{(\alpha+\beta)\tilde{\beta}}{\hat{\gamma}}$.

From Lemma 4.16, the number of honest PoS-blocks in $\mathsf{EXEC}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}$ is greater than or equal to that in $\mathsf{EXEC}^{r+s}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),\mathcal{A},\mathcal{Z}}$ and because messages of corrupted player would not be delayed. This implies that, in $\mathsf{EXEC}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}$, the ratio of honest PoS-blocks is no less than $1-(1+\delta)\frac{(\alpha+\beta)\tilde{\beta}}{\hat{\gamma}}$ in ℓ consecutive blocks of $\tilde{\mathcal{C}}$. This completes the proof. □

4.5 Achieving the Common Prefix Property

We now turn our attention to proving the common prefix property (Definition 2.4) for the proposed protocol. The concrete statement can be found in Theorem 4.3.

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, once the malicious parties send the hidden chain-pair to any honest party to get help then all of the parties would know the chain, and it is a public chain-pair now. Over all public chain-pairs, only the best one will be extended in the future rounds.

Putting them together, if any two chain-pairs are accepted by different honest parties in some round, that means there is at least one chain-pair is hidden to some public parties. Otherwise, the parties will make the same choice. If the hidden part of the PoW chain of this chain-pair is long, that means this part is generated only by malicious parties. After a long time it would be shorter than the PoW-chain of the public chain-pair which then cannot be accepted by any honest parties.

We remark that adversarial starting point is not better than honest players'starting point except one free block. More precisely, 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 as the starting point to fork a chain-pair and keep it hidden. That means the malicious PoS-holders can have one free block at the beginning of a hidden chain. Any hidden chain can have no more than such kind of block advantage.

We now prove an essential lemma that will be helpful in our analysis. Intuitively, the lemma shows that, in s consecutive rounds, the honest parties will extend the chain-pair faster than the malicious parties if they do not help each other. Let X denote the total number of honest stake successful rounds, in $\mathsf{EXEC}^{r+s}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}$ and X'' denote the total number of honest stake successful rounds in $\mathsf{EXEC}^{r+s}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),\mathcal{A},\mathcal{Z}}$. Y be the number of valid malicious PoW-blocks which are actually generated in s rounds, it is valid for both of the two executions.

Lemma 4.25. *Let $\hat{\gamma} = \hat{\lambda}\hat{\beta}$ and $\hat{\lambda} > 1$. Consider in $\mathsf{EXEC}^{r+s}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),\mathcal{A},\mathcal{Z}}$. For any $\delta > 0$ and $s > \frac{1}{\hat{\gamma}-\hat{\beta}}$, let X'' denote the total number of honest stake successful rounds and Y be the number of valid malicious PoW-blocks during any consecutive s rounds, we have $\Pr[X''-Y>1] \geq 1-e^{-\Omega(s)}$.*

Proof. Consider X'' number of honest stake successful rounds in $\mathsf{EXEC}^{r+s}_{(\Pi^{\mathsf{w}}_{\Delta},\Pi^{\mathsf{s}}_{\Delta}),\mathcal{A},\mathcal{Z}}$. First, from Lemma 4.21, we have $\Pr[X'' > (1-\delta)\hat{\gamma}s] > 1-e^{-\Omega(s)}$.

At the same time, the number of new valid malicious PoW-blocks is $\hat{\beta} \cdot s$ on average in s rounds. For any $\delta \in (0, 1]$, using the Chernoff bound analysis, we have $\Pr[Y \geq (1+\delta)\hat{\beta} \cdot s] \leq e^{-\delta^2 \hat{\beta} \cdot s/3}$. Thus, $\Pr[Y < (1+\delta)\hat{\beta} \cdot s] > 1 - e^{-\delta^2 \hat{\beta} \cdot s/3} = 1 - e^{-\Omega(s)}$.

If $s > \frac{1}{\hat{\gamma} - \hat{\beta}}$, for small δ and $\hat{\lambda} > 1$ we have $\Pr[X'' - Y > 1] \geq 1 - e^{-\Omega(s)}$.

□

We then argue the total number of honest successful rounds against the number valid malicious PoW-blocks in the typical execution $\mathsf{EXEC}_{(\Pi^{\mathsf{w}}, \Pi^{\mathsf{s}}), \mathcal{A}, \mathcal{Z}}$.

Lemma 4.26. *Let $\hat{\gamma} = \hat{\lambda}\hat{\beta}$ and $\hat{\lambda} > 1$. For any $\delta > 0$, if s is large enough, in protocol execution $\mathsf{EXEC}_{(\Pi^{\mathsf{w}}, \Pi^{\mathsf{s}}), \mathcal{A}, \mathcal{Z}}$ where Π is described in Section 3.2, let X denote the total number of honest stake successful rounds and Y be the number of valid malicious PoW-blocks during any consecutive s rounds, we have $\Pr[X - Y > 1] \geq 1 - e^{-\Omega(s)}$.*

Proof. From Lemma 4.25, we have $\Pr[X'' - Y > 1] \geq 1 - e^{-\Omega(s)}$. where X'' denote the total number of honest stake successful rounds and Y be the number of valid malicious PoW-blocks during any consecutive s rounds in $\mathsf{EXEC}^{r+s}_{(\Pi^{\mathsf{w}}_{\Delta}, \Pi^{\mathsf{s}}_{\Delta}), \mathcal{A}, \mathcal{Z}}$.

From Lemma 4.16, the number of honest stake successful rounds X in $\mathsf{EXEC}_{(\Pi^{\mathsf{w}}, \Pi^{\mathsf{s}}), \mathcal{A}, \mathcal{Z}}$ is greater than or equal to the number of honest stake successful rounds X'' in $\mathsf{EXEC}^{r+s}_{(\Pi^{\mathsf{w}}_{\Delta}, \Pi^{\mathsf{s}}_{\Delta}), \mathcal{A}, \mathcal{Z}}$ that is $X \geq X''$.

We also have Y is same in the two executions. We get $\Pr[X - Y > 1] \geq 1 - e^{-\Omega(s)}$.

□

If the malicious PoS-holder helps the honest PoW-miner to sign PoS-block, it would increase the longest valid PoS-chain, and this is not harmful. If the honest PoS-holder helps the malicious PoW-miner to sign PoS-block, it means this block is public and belongs to the longest valid PoS-chain which is also not harmful. Our useful lemma below states that, under the main assumption that $\hat{\gamma} = \hat{\lambda}\hat{\beta}$ and $\hat{\lambda} > 1$, the best public PoS-chain is longer than any public or hidden PoS-chain. This implies that the adversarial PoS-chain would not be better than the best public PoS-chain.

Lemma 4.27. *Let $\hat{\gamma} = \hat{\lambda}\hat{\beta}$ and $\hat{\lambda} > 1$. For any $\delta > 0$, consider the execution $\mathsf{EXEC}_{(\Pi^{\mathsf{w}}, \Pi^{\mathsf{s}}), \mathcal{A}, \mathcal{Z}}$, suppose there are two divergent valid PoS-chains $\tilde{\mathcal{C}}_i$ and $\tilde{\mathcal{C}}_j$ of two honest PoS-holders $\mathcal{S}_i$ and $\mathcal{S}_j$ in round r. We assume the last public PoS-block of $\tilde{\mathcal{C}}_j$ was public from round $r - s$. Let ℓ be the number of PoS-blocks generated on $\tilde{\mathcal{C}}_i$ from round $r - s$ to r. If $\tilde{\mathcal{C}}_i$ is the best public valid PoS-chain and s is large enough, we have $\Pr[\mathsf{len}(\tilde{\mathcal{C}}_i) \geq \mathsf{len}(\tilde{\mathcal{C}}_j)] \geq 1 - e^{-\Omega(\ell)}$*

Proof. We have the following two cases:

- If $\tilde{\mathcal{C}}_j$ is also a public chain and $\tilde{\mathcal{C}}_i$ is the best public PoS-chain. For $\tilde{\mathcal{C}}_i$ is the best public valid PoS chain, $\mathsf{len}(\tilde{\mathcal{C}}_i) \geq \mathsf{len}(\tilde{\mathcal{C}}_j)$ directly comes from the definition of best chain Process in Figure 8.
- If $\tilde{\mathcal{C}}_j$ is a hidden chain from the malicious parties. Let $\tilde{\mathcal{C}}'_i$ be the best public PoS-chain in round $r - s$ and $\tilde{\mathcal{C}}'_j$ denote the status of $\tilde{\mathcal{C}}_j$ in round $r - s$. We have $\mathsf{len}(\tilde{\mathcal{C}}'_i) \geq \mathsf{len}(\tilde{\mathcal{C}}'_j)$.

  From Lemma 4.26, we have that the number of honest successful stake rounds is greater than the number of valid malicious PoS-blocks by more than 1 with probability at least $1 - e^{-\Omega(s)}$ in s rounds. From Corollary 4.19 and Lemma 4.20, each honest successful stake round will increase the length of the public PoS-chain by 1 block and each valid malicious PoS-block will extend the hidden chain-pair by 1 block at most. Further more, the malicious hidden chain may have 1 more block advantage as in **Case 3**. Putting them together, we have, with the probability at least $1 - e^{-\Omega(s)}$,

$$\mathsf{len}(\tilde{\mathcal{C}}_i) \geq \mathsf{len}(\tilde{\mathcal{C}}'_i) + X \geq \mathsf{len}(\tilde{\mathcal{C}}'_j) + Y + 1 \geq \mathsf{len}(\tilde{\mathcal{C}}_j) \tag{5}$$

From Lemma 4.23, it follows that $\Pr[\mathsf{len}(\tilde{\mathcal{C}}_i) \geq \mathsf{len}(\tilde{\mathcal{C}}_j)] \geq 1 - e^{-\Omega(\ell)}$. □

Next, we will give an important lemma in our analysis. Intuitively, the lemma shows that the PoS-chains which were not chosen as the best PoS-chain (the PoS-chain in the best chain-pair) will never be accepted by any honest player in future.

Lemma 4.28. *We assume $\hat{\gamma} = \hat{\lambda}\hat{\beta}$ and $\hat{\lambda} > 1$. For any $\delta > 0$, consider the execution $\mathsf{EXEC}^{r+s}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}$. Let $\tilde{\mathcal{C}}$ be the best public PoS-chain of the system at round r. Let $\tilde{\mathcal{C}}'$ be a PoS-chain of an honest PoS-holder at round r. We assume $\tilde{\mathcal{C}}'$ and $\tilde{\mathcal{C}}$ diverge at round r' where there are ℓ PoS-blocks generated on $\tilde{\mathcal{C}}$ from round r' to r. It holds that the probability that $\tilde{\mathcal{C}}'$ is accepted by an honest player in the future is at most $e^{-\Omega(\ell)}$.*

Proof. Note that, during the rounds $s = r - r'$ there are at least ℓ blocks are generated to extend the best public PoS-chain.

Let X denote the total number of honest stake successful rounds where a single honest PoS-holder produces a block in that honest stake successful round in $\mathsf{EXEC}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}$. By Lemmas 4.26, we have

$$\Pr[X > (1-\delta)\hat{\gamma} \cdot s] > 1 - e^{-\Omega(s)} > 1 - e^{-\Omega(\ell)}$$

Let Y be the number of valid malicious PoW-blocks which are actually generated in s rounds. By Lemmas 4.26, we have

$$\Pr[Y < (1+\delta)\hat{\beta} \cdot s] > 1 - e^{-\Omega(s)} > 1 - e^{-\Omega(\ell)}$$

We then have $\Pr[X - Y > (\lambda - 1 - \delta)\hat{\beta}s] > 1 - e^{-\Omega(\ell)}$. This implies the best public PoS-chain is longer than PoS-chain $(\lambda - 1 - \delta)\hat{\beta}s$ blocks with probability $1 - e^{-\Omega(\ell)}$ where $s = \Omega(\ell)$.

If $\tilde{\mathcal{C}}'$ is accepted by an honest party in a future round r'' and let $s' = r'' - r$, we will prove the probability that $\tilde{\mathcal{C}}'$ is negligible in ℓ. Let Z be the number of honest stake successful rounds subtracts the number of valid malicious PoW-blocks in the s' rounds. We have $Z = (\hat{\gamma} - \hat{\beta})s' = (\lambda - 1)\hat{\beta}s'$ on average.

If $\tilde{\mathcal{C}}'$ is accepted by an honest party that means $(\lambda - 1 - \delta)\hat{\beta}s + Z \leq 0$. Let $Z' = (\lambda - 1 - \delta)\hat{\beta}s + Z$, by Chernoff bound, we have $\Pr[Z' < 0] < e^{-\Omega(s+s')} < e^{-\Omega(\ell)}$. □

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 4.3. *We assume $\hat{\gamma} = \hat{\lambda}\hat{\beta}$ and $\hat{\lambda} > 1$. For any $\delta > 0$, consider protocol $\Pi = (\Pi^{\mathsf{w}}, \Pi^{\mathsf{s}})$ in Section 3.2. 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 $r \leq r'$, 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. Let $\tilde{\mathcal{C}}$ be the best public PoS-chain in round r. From Lemma 4.27, we have $\tilde{\mathcal{C}}_i$ diverges with $\tilde{\mathcal{C}}$ no more than $\Theta(\kappa)$, otherwise it can not be the best chain for any honest party in round r.

Let $\tilde{\mathcal{C}}'_j$ be the status of $\tilde{\mathcal{C}}_j$ in round r. From lemma 4.28, we know that $\tilde{\mathcal{C}}'_j$ diverges with $\tilde{\mathcal{C}}$ no more than $\Theta(\kappa)$, otherwise it can not be the best chain for any honest party in round r'.

Put them together, we get $\tilde{\mathcal{C}}_i$ diverges with $\tilde{\mathcal{C}}'_j$ no more than $\Theta(\kappa)$. $\tilde{\mathcal{C}}'_j$ is prefix of $\tilde{\mathcal{C}}_j$, we have $\tilde{\mathcal{C}}_i$ diverges with $\tilde{\mathcal{C}}_j$ no more than $\Theta(\kappa)$.

□

Acknowledgement

We thank Thomas Veale for making the pictures for our paper, and for his helpful discussions.

The last author Hong-Sheng Zhou would like to thank Alexander Chepurnoy for his valuable discussions and feedback, and his encouragement for pursuing this work. Hong-Sheng would also like to thank Juan Garay and Aggelos Kiayias for answering his questions on the Bitcoin backbone paper, and Jeremiah Blocki, Jonathan Katz, Babis Papamanthou, and Vassilis Zikas for their helpful discussions at early stage of this project.

References

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

[2] I. Bentov, A. Gabizon, and A. Mizrahi. Currencies without proof of work. In *Financial Cryptography and Data Security (FC)*, 2016.

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

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

[5] V. Buterin. Understanding serenity, part 2: Casper. 2015. https://blog.ethereum.org/2015/12/28/understanding-serenity-part-2-casper/.

[6] V. Buterin. Proof of stake: How i learned to love weak subjectivity. November 2014. https://blog.ethereum.org/2014/11/25/proof-stake-learned-love-weak-subjectivity/.

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

[8] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000. http://eprint.iacr.org/2000/067.

[9] R. Canetti. Universally composable signatures, certification and authentication. Cryptology ePrint Archive, Report 2003/239, 2003. http://eprint.iacr.org/2003/239.

[10] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited (preliminary version). In *30th ACM STOC*, pages 209–218. ACM Press, May 1998.

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

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

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

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

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

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

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

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

[19] M. D. Green, J. Katz, A. J. Malozemoff, and H.-S. Zhou. A unified approach to idealized model separations via indistinguishability obfuscation. Cryptology ePrint Archive, Report 2014/863, 2014. http://eprint.iacr.org/2014/863.

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

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

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

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

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

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

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

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

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

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

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

[31] R. Ostrovsky and M. Yung. How to withstand mobile virus attacks (extended abstract). In L. Logrippo, editor, *10th ACM PODC*, pages 51–59. ACM, Aug. 1991.

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

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

[34] A. Sapirstein, Y. Sompolinsky, and A. Zohar. Optimal selfish mining strategies in bitcoin. In *Financial Cryptography and Data Security (FC)*, 2016.

[35] O. Schrijvers, J. Bonneau, D. Boneh, and T. Roughgarden. Incentive compatibility of bitcoin mining pool reward functions. In *Financial Cryptography and Data Security (FC)*, 2016.

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

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

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

A Resource Certificate Authority Functionality $\mathcal{F}^*_{\mathsf{rCA}}$

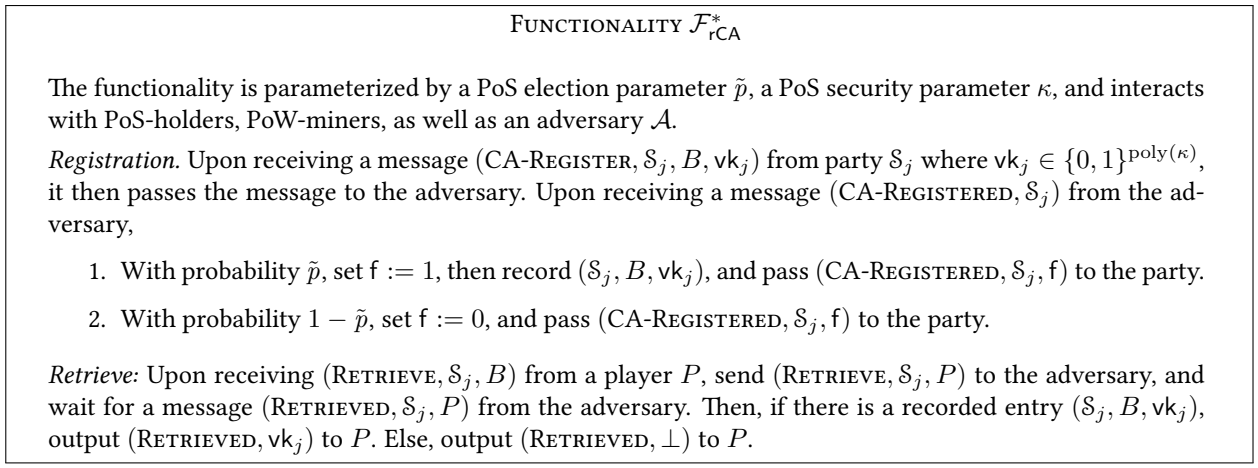
In this work, we consider the following scenario where a cryptocurrency system (e.g., Bitcoin) has "grown up" which means all players keep roughly the same blockchain; meaning that stake stored in this blockchain is distributed among all players. This mature blockchain can be used to implement a resource certification functionality $\mathcal{F}^*_{\mathsf{rCA}}$ described in Figure 9. In [9], Canetti introduce the certificate authority functionality $\mathcal{F}_{\mathsf{CA}}$; here, we introduce the resource certificate authority functionality $\mathcal{F}^*_{\mathsf{rCA}}$ without any trapdoor information and can be implemented by real world resource (e.g., computing power).

Note that, proof-of-stake is introduced to strengthen the proof-of-work blockchain. Specifically, our goal is to use proof-of-stake effectively to secure the proof-of-work blockchain if the adversary controls the majority of computing power but the majority of collective online resources (stake and computing) overall. We will later formally show that, under the assumption that the majority of collective online resources belongs to the honest players, even if the adversary dominates the proof-of-work chain (meaning that the adversary controls the majority of computing power), our protocol is still secure. Here, the honest stakeholders have an important role to protect the proof-of-work blockchain from the domination of the malicious players.

We argue that the scenario we consider here is realistic. Currently, the PoW-based cryptocurrency system such as Bitcoin is stable where the honest players have the majority of computing power, which means the majority of stake is also under the control of honest players. Then, the adversary may develop novel mining techniques and attempt to dominate the Bitcoin system. However, by our effective protocol design and under the plausible assumption that the majority of collective online resources is honest, these PoW-based cryptocurrency systems are protected.

Similarly to $\mathcal{F}^*_{\mathsf{rRO}}$, at any time step, a PoS-holder $\mathcal{S}_j$ could send a register command $(\text{CA-}\textsc{Register}, \mathcal{S}_j, B, \mathsf{vk}_j)$ to ask for registration. The functionality then records $(\mathcal{S}_j, B, \mathsf{vk}_j)$ (if permitted by the adversary), with probability $\tilde{p}$. Then, for each execution round, a different player P could request the functionality retrieving the message registered by $\mathcal{S}_j$, the functionality then returns the record of $\mathcal{S}_j$ if it permitted by the adversary. Otherwise, the player $\mathcal{S}_{j'}$ will not receive vk_j.

The formal description of $\mathcal{F}^*_{\mathsf{rCA}}$ is given in Figure 9.



Functionality $\mathcal{F}^*_{\mathsf{rCA}}$

The functionality is parameterized by a PoS election parameter $\tilde{p}$, a PoS security parameter κ, and interacts with PoS-holders, PoW-miners, as well as an adversary $\mathcal{A}$.

Registration. Upon receiving a message $(\text{CA-}\textsc{Register}, \mathcal{S}_j, B, \mathsf{vk}_j)$ from party $\mathcal{S}_j$ where $\mathsf{vk}_j \in \{0,1\}^{\mathrm{poly}(\kappa)}$, it then passes the message to the adversary. Upon receiving a message $(\text{CA-}\textsc{Registered}, \mathcal{S}_j)$ from the adversary,

1. With probability $\tilde{p}$, set $\mathsf{f} := 1$, then record $(\mathcal{S}_j, B, \mathsf{vk}_j)$, and pass $(\text{CA-}\textsc{Registered}, \mathcal{S}_j, \mathsf{f})$ to the party.
2. With probability $1 - \tilde{p}$, set $\mathsf{f} := 0$, and pass $(\text{CA-}\textsc{Registered}, \mathcal{S}_j, \mathsf{f})$ to the party.

Retrieve: Upon receiving $(\textsc{Retrieve}, \mathcal{S}_j, B)$ from a player P, send $(\textsc{Retrieve}, \mathcal{S}_j, P)$ to the adversary, and wait for a message $(\textsc{Retrieved}, \mathcal{S}_j, P)$ from the adversary. Then, if there is a recorded entry $(\mathcal{S}_j, B, \mathsf{vk}_j)$, output $(\textsc{Retrieved}, \mathsf{vk}_j)$ to P. Else, output $(\textsc{Retrieved}, \perp)$ to P.

Figure 9: Resource Certificate Authority Functionality.

There are multiple ways to instantiate $\mathcal{F}^*_{\mathsf{rCA}}$. Intuitively, in our main application scenario, $\mathcal{F}^*_{\mathsf{rCA}}$ is implemented by an already "mature" blockchain (i.e., Bitcoin) and a random oracle; at a specified point, this already mature blockchain changes its gear, and switch to a new mode (i.e., hybrid PoW/PoS protocol).

We denote ϕ_{rCA} as the ideal protocol for an ideal functionality $\mathcal{F}^*_{\mathsf{rCA}}$ and π_{CA} as the protocol in the $\mathcal{F}_{\mathsf{CA}}, \mathcal{F}_{\mathsf{RO}}$-hybrid model. In the ideal protocol ϕ_{rCA}, players are dummy, they just forward the messages received from the environment $\mathcal{F}^*_{\mathsf{rCA}}$ to the functionality $\mathcal{F}^*_{\mathsf{rRO}}$, and then forward the messages received from the functionality to the environment. In contrast, upon receiving messages from the environment, the players

in π_{CA} execute the protocol and then pass the outputs to the environment. The protocol π_{CA} is described in Figure 10.

PROTOCOL π_{CA}

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

1. Upon receiving $(\text{CA-REGISTER}, \mathcal{S}_j, B, \mathsf{vk}_j)$ from the environment $\mathcal{Z}$, send (B, vk_j) to the functionality $\mathcal{F}_{\mathsf{RO}}$ and receive h.
   - If $h > \tilde{\mathsf{D}}$ where $\tilde{\mathsf{D}} = \tilde{p} \cdot 2^{\kappa}$, then set $\mathsf{f} := 0$, and pass $(\text{CA-REGISTERED}, \mathcal{S}_j, \mathsf{f})$ to the environment.
   - Else, if $h \leq \tilde{\mathsf{D}}$, send $(\text{REGISTER}, \mathcal{S}_j, B, \mathsf{vk}_j)$ to the functionality $\mathcal{F}_{\mathsf{CA}}$. Upon receiving $(\text{REGISTERED}, \mathcal{S}_j, B, \mathsf{vk}_j)$ from $\mathcal{F}_{\mathsf{CA}}$, set $\mathsf{f} := 1$ and send $(\text{CA-REGISTERED}, \mathcal{S}_j, \mathsf{f})$ to the environment.
2. Upon receiving $(\text{RETRIEVE}, \mathcal{S}_j, B)$ from the environment, send $(\text{RETRIEVE}, \mathcal{S}_j, B)$ to the functionality $\mathcal{F}_{\mathsf{CA}}$ and then receive the output $(\text{RETRIEVED}, \mathsf{vk}_j)$. Then pass the message to the environment.

Figure 10: Resource Certificate Authority Protocol.

Let $\mathcal{S}$ be the adversary in the ideal protocol ϕ_{rCA}, and $\mathcal{A}$ be the adversary in protocol π_{CA}. Let $\mathsf{EXEC}^{\mathcal{F}^*_{\mathsf{rCA}}}_{\phi_{\mathsf{rCA}},\mathcal{S},\mathcal{Z}}$ denote random variable denoting the joint view of all parties in the execution of ϕ_{rCA} with the adversary $\mathcal{S}$ and an environment $\mathcal{Z}$. Let $\mathsf{EXEC}^{\mathcal{F}_{\mathsf{CA}},\mathcal{F}_{\mathsf{RO}}}_{\pi_{\mathsf{CA}},\mathcal{A},\mathcal{Z}}$ denote random variable denoting the joint view of all parties in the execution of π_{CA} with the adversary $\mathcal{A}$ and an environment $\mathcal{Z}$.

Lemma A.1. *Consider ϕ_{rCA} described above and described above and π_{CA} in Figure 10. It holds that the two ensembles $\mathsf{EXEC}^{\mathcal{F}^*_{\mathsf{rCA}}}_{\phi_{\mathsf{rCA}},\mathcal{S},\mathcal{Z}}$ and $\mathsf{EXEC}^{\mathcal{F}_{\mathsf{CA}},\mathcal{F}_{\mathsf{RO}}}_{\pi_{\mathsf{CA}},\mathcal{A},\mathcal{Z}}$ are perfectly indistinguishable.*

Proof. The adversary $\mathcal{S}$ on input 1^{κ} and $\tilde{p}$ operates as follows. Note that, $\mathcal{S}$ stores a table T.

1. Upon receiving (B, vk_j) from $\mathcal{A}$ in the name of $\mathcal{S}_j$, send $(\text{CA-REGISTER}, \mathcal{S}_j, B, \mathsf{vk}_j)$ to the functionality $\mathcal{F}^*_{\mathsf{rCA}}$ and receive $(\text{CA-REGISTERED}, \mathcal{S}_j, \mathsf{f})$. If $\mathsf{f} = 0$, choose random $h \in \{0,1\}^{\kappa}$ such that $h > \tilde{\mathsf{D}}$ where $\tilde{\mathsf{D}} = \tilde{p} \cdot 2^{\kappa}$, then store $((B, \mathsf{vk}_j), h))$ in the table T and send h to $\mathcal{A}$. If $\mathsf{f} = 1$, choose $h \in \{0,1\}^{\kappa}$ such that $h \leq \tilde{\mathsf{D}}$, then store $((B, \mathsf{vk}_j), h)$ in the table T and send h to $\mathcal{A}$.
2. Upon receiving $(\text{REGISTER}, \mathcal{S}_j, B, \mathsf{vk}_j)$ from $\mathcal{A}$, send $(\text{REGISTERED}, \mathcal{S}_j, B, \mathsf{vk}_j)$ to $\mathcal{A}$.
3. Upon receiving $(\text{RETRIEVE}, \mathcal{S}_j, B)$ from $\mathcal{A}$, send $(\text{RETRIEVE}, \mathcal{S}_j, B)$ to the functionality $\mathcal{F}^*_{\mathsf{rCA}}$ and obtain $(\text{RETRIEVED}, \mathsf{vk}_j)$. Then pass the message to the environment.

We now show that the two ensembles $\mathsf{EXEC}^{\mathcal{F}^*_{\mathsf{rCA}}}_{\phi_{\mathsf{rCA}},\mathcal{S},\mathcal{Z}}$ and $\mathsf{EXEC}^{\mathcal{F}_{\mathsf{CA}},\mathcal{F}_{\mathsf{RO}}}_{\pi_{\mathsf{CA}},\mathcal{A},\mathcal{Z}}$ are perfectly close. Notice that for each random oracle query from $\mathcal{A}$, the adversary $\mathcal{S}$ asks the functionality $\mathcal{F}^*_{\mathsf{rCA}}$ to decide whether this random oracle query is successful or not, then it samples the output randomly from a set $\{0,1\}^{\kappa}$. Moreover, for every register query to the functionality $\mathcal{F}_{\mathsf{CA}}$, $\mathcal{S}$ would accept if it the random oracle query is successful. Putting them together, the views of players in the two executions are perfectly indistinguishable. □

Remark A.2. *The functionality $\mathcal{F}_{\mathsf{CA}}$ can be instantiated by a mature and continuing blockchain. Intuitively, the blockchain is public and all participants agree on the chain except that last several blocks. Therefore, everyone can derive the latest set of valid stakeholders from the common prefix of the blockchain with their corresponding stake.*

B Signature Functionality $\mathcal{F}_{\mathsf{SIG}}$

We present the signature functionality following the modeling of [9].

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

Key Gerneration: Upon receiving a message (KEYGEN) from some party S, hand (KEYGEN) to the adversary. Upon receiving $(\text{VERIFICATION-KEY}, \mathsf{vk})$ from the adversary, output $(\text{VERIFICATION-KEY}, \mathsf{vk})$ to the party S.

Signature Generation: Upon receiving a message (SIGN, m) from S, send (SIGN, m) to the adversary. Upon receiving $(\text{SIGNATURE}, m, \sigma)$ from the adversary, verify that no entry $(m, \sigma, \mathsf{vk}, 0)$ is recorded. If it is, then output an error message to S and halt. Otherwise, output $(\text{SIGNATURE}, m, \sigma)$ to S, and record the entry $(m, \sigma, \mathsf{vk}, 1)$.

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

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

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

Figure 11: Signature Functionality.