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

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

April 15, 2017

Abstract

Cryptocurrencies like Bitcoin have proven to be a phenomenal success. Bitcoin-like systems use proof-of-work mechanism, 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 and proof-of-stake 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.

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

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

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

Contents

1 Introduction

Cryptocurrencies like Bitcoin [34] have proven to be a phenomenal success. The underlying techniques hold a huge promise to change the future of financial transactions, and eventually our way of computation and collaboration. At the heart of the Bitcoin system is a global public distributed ledger, called *blockchain*, that records transactions between *users* in consecutive time windows. The blockchain is maintained by a peer-to-peer network of nodes called Bitcoin *miners* via the so-called proof-of-work (PoW) mechanism: in each time window, cryptographic puzzles (also called proof-of-work puzzles [16, 2]) are generated, and all miners are encouraged/incentivized to solve the puzzles; the first miner who finds a puzzle solution is allowed to extend the blockchain with a block of transactions, and at the same time he can collect a reward. It is easy to see that the more computing power a miner invests, the better his chances are at solving a puzzle first.

Bitcoin is an **open** system; any player who invests a certain amount of computing resources is allowed to join the effort of maintaining the blockchain. This unique "easy to join/leave" feature along with a smart incentive strategy help the system "absorb"[1] a huge amount of computing resources over the past several years. Intuitively, the security of blockchain is backed up by a significant network of physical computing resources.

This intuition has recently been investigated in academia. For example, Garay et al. [19] and then Pass et al. [40] 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 [20]. 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 [32], 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 the current Bitcoin system to be more robust in the sense that the system remains stable even the adversary controls the majority of computing power. As mentioned, Bitcoin already "absorbed" a huge

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

amount of honest computing power; note that these physical resources have been converted into "virtual resources", i.e., the coins. It is fairly reasonable to assume the coins are *nicely distributed* in the system and most of them are controlled by honest users. A natural way to go is to use this huge amount of honest *virtual* resources as a buffer to defend against the adversary who can dominate the network of computing power.

The difference between physical resources and virtual resources. It is definitely desirable to utilize the power of virtual resources to secure a blockchain. If successful, the new system will be "green" in the sense that it does not require a huge amount of *physical resources*, which cannot be recycled, to back up its security. Attempts have been made. For example, proof-of-stake (PoS) mechanisms have been widely discussed in the cryptocurrency community. In a nutshell, proof-of-stake mechanisms for consensus require protocol players to prove ownership of a certain amount of virtual resources. Only those that can provide such a proof can participate in maintaining the blockchain. However, it is not clear how to construct an open blockchain which can scale to a large number of network nodes (as in Bitcoin), via any proof-of-stake mechanism. At a very intuitive level, virtual resources, which proof-of-stake mechanisms are based on, are very difficult to manage in a practical protocol. This intuition has been confirmed recently by the concurrent works of [30, 27, 6]); there, very interesting and non-trivial attempts have been made to construct provably secure, scalable blockchains via pure proof-of-stake. Unfortunately, these solutions cannot scale to a large number of network nodes in an *open* setting where participants can freely join or leave the system any time they want. In more details, a secure bootstrapping mechanism (i.e., majority voting) is required for ensuring new participants can securely join the system. However, this "bootstrapping" cannot scale to a large network. See Section 1.5 (Related Work) for more discussion. In one word, it seems it is extremely difficult to mimic Nakamoto's footprint via virtual resources *only*.

On the other hand, physical resources are relatively easier to manage. Indeed, Nakamoto demonstrates to us an amazingly practical protocol via the proof-of-work mechanism to manage physical computing resource effectively. Alternative physical resources such as a publicly available random beacon or secure hardware can also allow us to construct fast protocols. See Section 1.5 (Related Work) for more discussion about open blockchain via alternative physical resources.

The practical elegance of using random oracle for cryptocurrency. Although fast blockchain protocols can be constructed via physical resources such as random beacon or trusted hardware, there is a significant drawback in these solutions. That is, the trapdoor information of the system is possessed by a single party. Currently, it is not clear how to eliminate such trapdoor information. Interestingly, blockchain via the proof-of-work mechanism can avoid such issue in practice: the underlying proof-of-work puzzles can be based on hash functions, and the security of the system can be argued in the so-called random oracle model. Theoretical cryptographers may criticize random oracle methodology since it is not sound [11]. However, random oracles do enable an elegant solution to open blockchains in practice.[2]

Additional considerations. There are many reasons that Bitcoin has become a successful system. For example, proof-of-work puzzles Bitcoin is a *divisible* e-cash system [38, 37]. Besides the points we discussed above, to design a practical blockchain, we in general should avoid heavy cryptographic tools, and use only standard cryptographic primitives such as hash functions and digital signature schemes. Futhermore, the design should be simple. Finally, the provable security approach should be taken to develop blockchain techniques. We eventually should move these powerful blockchain techniques from an art to a science.

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

[2] We note that Nakamoto's design is consistent with the folklore wisdom of a "nothing up my sleeve number" [45] which has been widely used in practical cryptographic designs.

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

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 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). Once a new PoW-block is found, the PoW-round completes, and a PoS-round starts immediately; based on the proof-of-work chain and the new PoW-block, a PoS-holder is chosen (i.e., testing each stakeholder's verification key via another hash inequality), and this PoS-holder then has the privilege to extend the best valid chain-pair on its view (i.e., by signing and approving new transactions). 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.

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

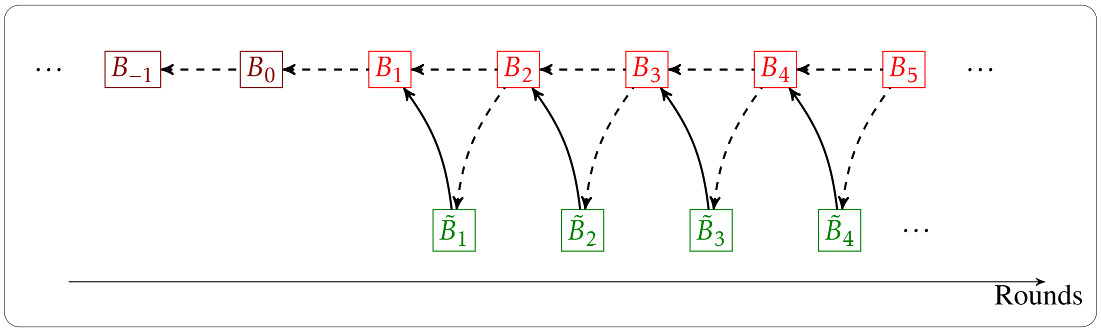


Figure 1: 2-hop blockchain structure
Here, dot arrows denote the first hops, and solid arrows denote the second hops. Red blocks B_i's denote the proof-of-work blocks, and green blocks $\tilde{B}_i$'s denote the corresponding proof-of-stake blocks. Note that the dark-red blocks are from the "mature blockchain".

our blockchain if the honest players control majority of the *collective*[3] 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 [19] and Pass et al [40], 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 se-

[3]Please see Section 4 for more discussions on collective resources.

cure 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 [19, 40] 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 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

Subsequent work. The 2-hop blockchain in this paper has been implemented [1]. We note that 2-hop blockchain design should be further extended so that it can be deployed in practice. Some efforts have been made in TwinsCoin [14, 1]. In particular, in TwinsCoin, a new mechanism has been introduced to adjust the difficulties for both PoW and PoS chains. Additional practical considerations have been made in TwinsCoin: the protocol is designed in the non-flat setting where protocol players may have different amounts of stake or computing power; light clients have made possible for the first time for PoS.

Closely related work on combining proof-of-work and proof-of-stake. The idea of combining proof-of-work and proof-of-stake has been studied in [28, 15, 5], and the latest one [5] is closest to our work. Although these studies showed that their protocols are secure against some classes of attacks, they *do not provide any formal security model and analysis relying on precise definitions*. We also highlight that the earlier proposal [5] is fundamentally different from our result on both security analysis and the construction. The proof-of-activity (PoA) proposal in [5] will not immediately give us a divisible cryptocurrency [38, 37] unless extra efforts are paid (this point will be elaborated below). This means that the PoA proposal cannot be very practical. In contrast, our proposal naturally mimics Nakamoto's footprint, and it will immediately give us a practical divisible solution. Below are more elaborations:

[4]Blockchain protocols such as PoW-based blockchains [34] that utilize physical assumptions (e.g., computational power) have a 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.

- The PoA proposal uses the follow-the-satoshi (FTS) function to choose a leader, where satoshi is the unit of stake. In order to track which satoshi is chosen in the future, transactions must keep satoshi identities in PoA. For example, if a user wants to transfer m satoshis to another, he must state all of the m identities of satoshis in the transaction. It is very inconvenient to store/retrieve the transactions. More sophisticated mechanism should be introduced to improve the performance which further complicates the design. We, on the other hand, follow Nakamoto's footprint via a hash inequality (i.e, $H(B, \mathsf{vk}) < \tilde{\mathsf{D}}$) to choose a vk (corresponding to a stakeholder). Interestingly, this helps us to avoid the issue of the FTS approach.

- We decouple single chain into PoW/PoS chain-pairs on purpose. If players believe majority of hashing power is honest, they may just validate PoW-chain. Verifying PoW-chain (which is based on hash function) is much faster than PoS-chain (which is based on signatures). In contrast, in the PoA proposal, PoW and PoS are composed together, and signatures have to be verified.

In addition, neither best chain strategy nor chain/block validation has been carefully defined for stakeholders in the PoA proposal. We remark that these protocol specifications are critical. Without the specifications, it is not clear how to implement their design idea. Furthermore, in the PoA, miners only generate empty block header and then the header will be mapped to N stakeholders, where $N > 1$ and typically N is set as 3. We do not know how to prove the basic security properties for $N > 1$.

Concurrent work on provably secure pure proof-of-stake. Very recently, concurrent works (e.g., [30, 27, 6]) have made very interesting attempts to construct provably secure, scalable blockchains via PoS only. Unfortunately, existing solutions cannot scale to a large number of network nodes in an *open* setting where participants can freely join or leave the system any time they want.

In more details, all existing provably secure, PoS-only systems need a secure bootstrapping mechanism (i.e., majority voting) for participants that newly join or rejoin late. Without this mechanism, the adversary can corrupt elected leaders at some later point of time and generate an "alternative" blockchain which could be longer than the true blockchain even if he does not control the majority of stake. This voting process requires most of existing players to participate in the protocol, and it does not scale in e.g., billion-scale networks. As pointed out in [6], such joining issue does not happen in a proof-of-work blockchain (e.g., Bitcoin). Valid blocks in a proof-of-work blockchain are not generated "for free"; therefore if a participant becomes corrupt later in time, he can overwrite history (generate an alternative blockchain) only with the minority of computing, and then new players can easily identify the true version of the blockchain. In short, these proof-of-stake only systems can be applied in a nearly closed environment.

At the moment of writing this paper, it is not clear if we can construct a provably secure, scalable open blockchain via pure PoS. Instead, we turn to combing PoW and PoS to achieve "the best of both worlds". This approach does not suffer from the issue discussed above since PoS is now tied to PoW.

Cryptocurrency and security analysis. Anonymous digital currency was introduced by Chaum [13] in the early 1980s. The first decentralized currency system, Bitcoin [34], was launched about 30 years later, by incentivizing a set of players to solve moderately-hard cryptographic puzzles (also called proofs-of-work puzzles [16, 2]). Recently, the security of Bitcoin system has been analyzed in the rational setting, e.g., [18, 17, 35, 24, 41, 42], and also in cryptographic setting [19, 40, 43, 25, 26]. 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. [19]. The chain growth property was first formally defined by Kiayias et al. [25]. The common prefix property was later strengthened by Pass et al. [40]. In our study, we adopt the stronger variant of the common prefix property by Pass et al. [40] together with the chain quality and chain growth from [25, 19].

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 [39, 31]. Between the use of space/memory and the use of time, are *proofs of space time* introduced in [33]. This is a hybrid proof system utilizing both computational and space resources. Intel proposes the use of trusted hardware for blockchain protocols in [22].

Cryptocurrency via virtual resources. We already discussed recent effort on provably secure pure proof-of-stake (PoS) proposals [30, 27, 6]. Before these recent rigorous efforts, using virtual resources (i.e., stake) to construct cryptocurrency has been intensively considered but in an ad hoc way, in the Bitcoin community. In a nutshell, the PoS mechanisms for consensus require a protocols' players to prove ownership of virtual resources. Only those that can provide such proofs can participate in maintaining the protocol's blockchain, their ability to do so is proportional to the stake owned. Since the inception of the idea in an online forum [7], several variants of PoS that have been proposed and implemented in real cryptocurrencies including [36, 29, 44, 4]. 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 [28, 15, 5], the proof of work and proof of stake can be combined together. We note that, before our work here, there is no known provably secure open blockchain via the combination of proof of work and proof of stake.

Organization. In Section 2, we present our analysis framework. In Section 3, we present the details of our construction, and then in Section 4 we give our security analysis ideas. Finally, additional supporting materials for security model are provided in Appendix A.

2 Model

2.1 2-hop Blockchain Protocol Executions

In order to study the security of Bitcoin-like protocol, Garay et al. [19] and then Pass et al. [40] set up the first cryptographic models by following Canetti's formulation of the "real world" executions [8, 9]. In this section, we borrow many ideas from their formulations. We further extend their models so that more blockchain protocols, e.g., 2-hop blockchains, are allowed.

Network communication. The underlying communication for blockchain protocols are formulated via a functionality $\mathcal{F}_{\mathsf{NET}}$ which captures the atomic unauthenticated "send-to-all" broadcast in an asynchronous communication setting. The functionality is parameterized by an upper bound Δ on the network latency, and interacts with players under the direction of the adversary. More concretely, the functionality proceeds as follows. Whenever it receives a message from a player, it would contact the adversary to ask the adversary to specify the delivery time for the message. Note that, if the specified delivery time exceeds the delay upper bound Δ, the functionality would not follow the adversary's instruction, and only delay the message to a maximum number of Δ rounds. That said, no messages are delayed more than Δ rounds. In addition, the adversary could read all messages sent by all honest players before deciding his strategy; the adversary may "spoof" the source of a message they transmit and impersonate the (honest) sender of the message. The functionality $\mathcal{F}_{\mathsf{NET}}$ is formally described in Figure 12 (See Appendix A.2.1).

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-miner or PoS-holder may have a different amount of computing power/stake; nevertheless, this idealized model does not sacrifice generality since one can imagine that real 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 include many chain-pairs from the network, each PoW-miner would choose the best valid chain-pair, and then utilize his computing power to solve the PoW puzzle in order to extend the best chain-pair in this round. On the other hand, in each PoS-round, the PoS-holder with the derived identity from the new PoW-block of the previous PoW-round is able to generate a new PoS-block, and then appends the new block to the best chain-pair on his local view. Note that, for each PoS-holder, the probability of being chosen is based on the amount of stake that party has. The detail of our blockchain execution is presented below.

The $\{\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_{\mathsf{NET}}\}$-hybrid execution of 2-hop blockchain protocol. Existing rigorous formulations (e.g., [19, 40]) 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 Canetti's formulation of the "real world" executions [8, 9], 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 PoS-holders, respectively. We consider the execution of the blockchain protocol $\Pi = (\Pi^{\mathsf{w}}, \Pi^{\mathsf{s}})$ that is directed by an environment $\mathcal{Z}(1^\kappa)$ (where κ is a security parameter), which activates an n number of PoW-miners and $\tilde{n}$ number of PoS-holders. The execution proceeds in *rounds*. Without lost of generality, we assume that odd rounds correspond to PoW-miners, and even rounds correspond to PoS-holders. The environment $\mathcal{Z}$ can "manage" protocol players thru an adversary $\mathcal{A}$ that can dynamically corrupt honest parties, but such corruption takes a while, i.e., 2Δ rounds, to be effective.

More concretely, the $\{\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_{\mathsf{NET}}\}$-hybrid execution proceeds as follows. Each party in the execution is initialized with an initial state $state_0$. This state includes all initial public information of the protocol Π, e.g., a genesis block. 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.

- In each odd round, each PoW-miner $\mathcal{W}_i$, for $1 \le i \le n$, with a local state $state_i$ (note that $state_i = state_0$ initially), proceeds as follows.
    - When PoW-miner $\mathcal{W}_i$ is activated by the environment $\mathcal{Z}$ with a message $(\textsc{Input-Work}, \mathcal{W}_i, x)$ where x is the input of the execution, and potentially receive incoming message $(\textsc{Message}, P, m)$

from $\mathcal{F}_{\mathsf{NET}}$ for any $P \in \{\mathcal{W}_1, \ldots, \mathcal{W}_n, \mathcal{S}_{n+1}, \ldots, \mathcal{S}_{n+\tilde{n}}\}$. 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 $\mathcal{Z}$.

- In each even round, each PoS-holder $\mathcal{S}_j$, for $n+1 \leq j \leq n+\tilde{n}$, with a local state $state_j$ (note that $state_j = state_0$ initially), proceeds as follows.
  - When PoS-holder $\mathcal{S}_j$ is activated by the environment $\mathcal{Z}$ by $(\textsc{Input-Stake}, \mathcal{S}_j, \tilde{x})$ where $\tilde{x}$ is the input from the environment, and potentially receive subroutine output message $(\textsc{Message}, P, m)$ for any $P \in \{\mathcal{W}_1, \ldots, \mathcal{W}_n, \mathcal{S}_{n+1}, \ldots, \mathcal{S}_{n+\tilde{n}}\}$,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 $\mathcal{Z}$.

- At any round r of the execution, $\mathcal{Z}$ can send message $(\textsc{Corrupt}, P)$, where $P \in \{\mathcal{W}_1, \ldots, \mathcal{W}_n, \mathcal{S}_{n+1}, \ldots, \mathcal{S}_{n+\tilde{n}}\}$, to adversary $\mathcal{A}$. Party P then remains honest till round $r + 2\Delta$. Then $\mathcal{A}$ will have access to the party's local state and control P from round $r + 2\Delta$.

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 utilize the computing power of the functionality, each player needs to register the computing services of $\mathcal{F}^*_{\mathsf{rRO}}$ or disconnect the services at some later points. Indeed, this captures the dynamic computing power setting where different players could consume the computing resource for different windows of time. Functionality $\mathcal{F}^*_{\mathsf{rRO}}$ abstracts out the Bitcoin like mining process; this will simplify the design and analysis of protocols based on such mining ecosystem. Here, each PoW-miner is able to request one *search* query from $\mathcal{F}^*_{\mathsf{rRO}}$ that consumes one unit of computing power granted in each execution 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 reset to 0 indicating that the resource will not be granted to this player any longer. Note that, here, we let the environment $\mathcal{Z}$ (through the adversary) specify who will receive the computing resource and who will not.

The functionality proceeds in rounds, and for each round, it sets a bit $b_i^{\mathsf{w}} = 0$ 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 the search query, the bit $b_i^{\mathsf{w}} = 0$ is reset to 1, and then this player will not be able to request any other search queries. 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 $b_i^{\mathsf{w}} = 0$ 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 search queries 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., checks 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 each regular query, and denote h' as that for each 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.

We remark that we are not the first to formulate the setup of computing resources. Earlier efforts can be found in [23, 40]. We argue that, our $\mathcal{F}^*_{\mathsf{rRO}}$ formulation is more rigorous than the previous efforts; we explicitly model how the computing resource is managed and distributed from the environment to parties. In our model, each party can 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.

Furthermore, our $\mathcal{F}^*_{\mathsf{rRO}}$ is closely related to, but different from $\mathcal{F}_{\mathsf{Tree}}$ in [40]. In [40], a "per protocol" approach is taken. That is, for different blockchain protocol, say GHOST protocol [43], 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 setup *per protocol*.

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 $\mathcal{F}_{\mathsf{RO}}$ [21]. We denote ϕ^*_{rRO} as the ideal protocol for an ideal functionality $\mathcal{F}^*_{\mathsf{rRO}}$ and π_{rRO} as the protocol in the

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

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

1. Upon receiving a message (WORK-REGISTER, $\mathcal{W}_i$) from party $\mathcal{W}_i \in \{\mathcal{W}_1, \ldots, \mathcal{W}_n\}$, if there is an entry $(\mathcal{W}_i, 1)$, then ignore the message. Otherwise, pass the message to the adversary. Upon receiving a message (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 (WORK-UNREGISTER, $\mathcal{W}_i$) from party $\mathcal{W}_i \in \{\mathcal{W}_1, \ldots, \mathcal{W}_n\}$, if no entry $(\mathcal{W}_i, 1)$ is recorded, then return (ERROR) to $\mathcal{W}_i$ and halt. If there is an entry $(\mathcal{W}_i, 1)$ recorded, set $\mathsf{b}_i := 0$, and update $(\mathcal{W}_i, \mathsf{b}_i)$, and then send (WORK-UNREGISTERED, $\mathcal{W}_i$) to the party $\mathcal{W}_i$ (the party $\mathcal{W}_i$ unregistered.)

For each round, set $b_i^{\mathsf{w}} := 0$ for every registered party $\mathcal{W}_i$ $(1 \le i \le n)$, then proceed as follows.
Regular Query. Upon receiving (COMPUTE, B, X) from a party P, if there is record of the form (B, X, h), send (COMPUTED, h) to the player P. If not, choose random $h \in \{0,1\}^\kappa$, send (COMPUTED, h) to the player P and record (B, X, h).
Work Query. Upon receiving (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 $b_i^{\mathsf{w}} = 0$ (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 $b_i^{\mathsf{w}} := 1$, and record $(\mathcal{W}_i, \langle h, w\rangle, h')$. Then send (SEARCHED, $\mathcal{W}_i, w$) to the player $\mathcal{W}_i$ (the party $\mathcal{W}_i$ discovers the solution,)
   - with probability $1 - p$, set $b_i^{\mathsf{w}} := 1$, and send (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 $b_i^{\mathsf{w}} = 1$ (the party $\mathcal{W}_i$ already used the granted computational resource in this round),

   Then send (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 where $P \in \{\mathcal{W}_1, \ldots, \mathcal{W}_n, \mathcal{S}_{n+1}, \ldots, \mathcal{S}_{n+\tilde{n}}\}$, 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 $\mathcal{F}^*_{\mathsf{rRO}}$.

$\mathcal{F}_{\mathsf{RO}}$-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_{rRO} execute the protocol and then pass the outputs to the environment. Note that, we allow each PoW-miner to receive only *one unit of computing power* (one chance of querying the random oracle) per round.

Essentially, protocol π_{rRO} captures the core of PoW-based blockchain (e.g., Bitcoin). Informally, π_{rRO} carries out the following two main steps: first, each PoW-miner is able to“mine” for a puzzle solution of a hash inequality; after that, any other players (PoW-miners or PoS-holders) can verify the found solution. The formal description of protocol π_{rRO} is given in Figure 3.

Let $\mathcal{S}$ be the adversary against the ideal protocol ϕ^*_{rRO}, and $\mathcal{A}$ be the adversary against the hybrid protocol π_{rRO}. We now show that π_{rRO} is as “ secure” as ϕ^*_{rRO} with respect to the adversary $\mathcal{S}$. Let

PROTOCOL π_{rRO}

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

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

1. Upon receiving (WORK-REGISTER, $\mathcal{W}_i$) from the environment $\mathcal{Z}$, if $\mathcal{W}_i$ already recorded $\mathsf{b}_i = 1$, then ignore the message. Otherwise, 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 $\mathcal{Z}$, if $\mathcal{W}_i$ has not recorded $\mathsf{b}_i = 1$, then return (ERROR) to $\mathcal{Z}$ and halt. Otherwise, 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 $B = (h, w)$ and then obtain output h'. If $h' \le \mathtt{D}$ where $\mathtt{D} = p \cdot 2^\kappa$, send (SEARCHED, $\mathcal{W}_i, w$) to the environment. Otherwise, if $h' > \mathtt{D}$, send (SEARCHED, $\mathcal{W}_i, \perp$) 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 $P \in \{\mathcal{W}_1, \dots, \mathcal{W}_n, \mathcal{S}_{n+1}, \dots, \mathcal{S}_{n+\tilde{n}}\}$ 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 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 $h' \le \mathtt{D}$, send (RO-VERIFIED, h') to the environment. Otherwise, if $h' > \mathtt{D}$, send (RO-VERIFIED, $\perp$) to the environment.

Figure 3: Resource random oracle protocol π_{rRO}.

$\mathsf{EXEC}^{\mathcal{F}_{\mathsf{RO}}}_{\pi_{\mathsf{rRO}},\mathcal{A},\mathcal{Z}}$ be the random variable denoting the joint view of all parties in the execution of π_{rRO} with the adversary $\mathcal{A}$ and an environment $\mathcal{Z}$. In addition, let $\mathsf{EXEC}^{\mathcal{F}^*_{\mathsf{rRO}}}_{\phi^*_{\mathsf{rRO}},\mathcal{S},\mathcal{Z}}$ be the random variable denoting the joint view of all parties in the execution of ϕ^*_{rRO} with the adversary $\mathcal{S}$ and an environment $\mathcal{Z}$.

We prove the following lemma in Appendix A.3.1.

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

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. Please refer to Figure 4 for more details.

Essentially, at any time step, a PoS-holder $\mathcal{S}_j$ can send a register command (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 (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

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

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

Stake Resource Registration.

1. Upon receiving a message (STAKE-REGISTER, $\mathcal{S}_j$) from party $\mathcal{S}_j \in \{\mathcal{S}_{n+1}, \ldots, \mathcal{S}_{n+\tilde{n}}\}$, if there is an entry $(\mathcal{S}_j, 1)$, then ignore the message. Otherwise, pass the message to the adversary. Upon receiving a message (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 $\mathcal{S}_j$ (the party $\mathcal{S}_j$ registered.)
2. Upon receiving a message (STAKE-UNREGISTER, $\mathcal{S}_j$) from party $\mathcal{S}_j \in \{\mathcal{S}_{n+1}, \ldots, \mathcal{S}_{n+\tilde{n}}\}$, if no entry $(\mathcal{S}_j, 1)$ is recorded, then return (ERROR) to $\mathcal{S}_j$ and halt. If there is an entry $(\mathcal{S}_j, 1)$ recorded, then set $\mathsf{b}_j := 0$, and update $(\mathcal{S}_j, \mathsf{b}_j)$, and send (STAKE-UNREGISTERED, $\mathcal{S}_j$) to the party $\mathcal{S}_j$ (the party $\mathcal{S}_j$ unregistered.)

For each round, set $b^{\mathsf{s}}_j := 0$ for every registered party $\mathcal{S}_j$ $(n+1 \leq j \leq n+\tilde{n})$, then proceed as follows.

Stake Election: Upon receiving (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 (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 (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 (ELECTED, $\mathcal{S}_j, \mathsf{f}$) to $\mathcal{S}_j$ (the party $\mathcal{S}_j$ is not elected.)

Signature Generation: Upon receiving (SIGN, $\mathcal{S}_j, B, X$) from a party $\mathcal{S}_j$, send (SIGN, $(\mathcal{S}_j, B, X)$) to the adversary. Upon receiving (SIGNATURE, $(\mathcal{S}_j, B, X), \sigma$) from the adversary, verify that no entry $((\mathcal{S}_j, B, X), \sigma, 0)$ is recorded. If it is, then output an error message (ERROR) to $\mathcal{S}_j$ and halt. Else, output (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 (STAKE-VERIFY, $(\mathcal{S}_j, B, X), \sigma$) from a party $P \in \{\mathcal{W}_1, \ldots, \mathcal{W}_n, \mathcal{S}_{n+1}, \ldots, \mathcal{S}_{n+\tilde{n}}\}$,

1. If there exists a record of the form $(B, \mathcal{S}_j, \cdot)$ (the party $\mathcal{S}_j$ is elected), hand (STAKE-VERIFY, $(\mathcal{S}_j, B, X), \sigma$) to the adversary. Upon receiving (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 (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 (STAKE-VERIFIED, $(\mathcal{S}_j, B, X), f$) to the party P.

Figure 4: Resource certification functionality $\mathcal{F}^*_{\mathsf{rCERT}}$.

command $(\text{SIGN}, (\mathcal{S}_j, B, X))$, 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 to 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.

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

Our functionality $\mathcal{F}^*_{\mathsf{rCERT}}$ can be a "resource" analog of the *multi-session* version of certificate functionality $\mathcal{F}_{\mathsf{CERT}}$ in [10]. Note that $\mathcal{F}_{\mathsf{CERT}}$ can be implemented in the $\{\mathcal{F}_{\mathsf{CA}}, \mathcal{F}_{\mathsf{SIG}}\}$-hybrid model [10]. We can follow the approach to implement our functionality $\mathcal{F}^*_{\mathsf{rCERT}}$. Here, the functionality $\mathcal{F}^*_{\mathsf{rCERT}}$ can be instantiated in the $\{\mathcal{F}^*_{\mathsf{rCA}}, \hat{\mathcal{F}}_{\mathsf{SIG}}\}$-hybrid model, where $\hat{\mathcal{F}}_{\mathsf{SIG}}$ is a *multi-session* signature functionality [12] and the resource certificate authority functionality $\mathcal{F}^*_{\mathsf{rCA}}$ can be implemented in the $\{\hat{\mathcal{F}}_{\mathsf{CA}}, \mathcal{F}_{\mathsf{RO}}\}$-hybrid model. In addition, the multi-session certificate authority functionality $\hat{\mathcal{F}}_{\mathsf{CA}}$ can be implemented by a "mature" blockchain. Please refer to Appendix A.1 for more details about resource certificate authority functionality $\mathcal{F}^*_{\mathsf{rCA}}$ and multi-session signature functionality$\hat{\mathcal{F}}_{\mathsf{SIG}}$; note that, $\mathcal{F}^*_{\mathsf{rCA}}$ can be viewed as a "resource" analogy of the multi-session version of certificate authority functionality $\mathcal{F}_{\mathsf{CA}}$ in [10]. In Appendix A.1, we will show that $\mathcal{F}^*_{\mathsf{rCA}}$ can be instantiated in the $\{\hat{\mathcal{F}}_{\mathsf{CA}}, \mathcal{F}_{\mathsf{RO}}\}$-hybrid model. Then $\hat{\mathcal{F}}_{\mathsf{CA}}$ can be implemented by a mature blockchain. Please refer to Figure 5 for a hierarchical implementation of our $\mathcal{F}^*_{\mathsf{rCERT}}$.

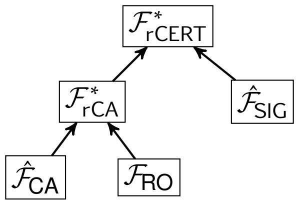


Figure 5: A hierarchical implementation of $\mathcal{F}^*_{\mathsf{rCERT}}$.

We denote ϕ^*_{rCERT} as the ideal protocol for an ideal functionality $\mathcal{F}^*_{\mathsf{rCERT}}$ and π_{rCERT} as protocol in $\{\mathcal{F}^*_{\mathsf{rCA}}, \hat{\mathcal{F}}_{\mathsf{SIG}}\}$-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. Informally, each PoS-holder through his stake determines whether he is the elected leader in the current round or not; then he is able to generate a valid signature, which can later be verified by any other players (PoW-miners or PoS-holders). To present our protocol clearly, we adopt the sid and ssid from [12] in π_{rCERT}. The protocol π_{rCERT} is formally described in Figure 6.

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

We prove the following lemma in Appendix A.3.2.

Protocol π_{rCERT}

The protocol is parameterized by a PoS parameter $\tilde{p}$ and a security parameter κ.
Each PoS-holder $\mathcal{S}_j$, where $n+1 \leq j \leq n+\tilde{n}$, proceeds as follows.

1. Upon receiving $(\textsc{Stake-Register}, \mathcal{S}_j)$ from the environment $\mathcal{Z}$, if $\mathcal{S}_j$ already recorded $\mathsf{b}_j = 1$, then ignore the message. Otherwise, set $\mathsf{b}_j := 1$, record b_j, and pass $(\textsc{Keygen}, \mathsf{sid}, \mathsf{ssid})$ for some $\mathsf{sid}, \mathsf{ssid}$ to the functionality $\hat{\mathcal{F}}_{\mathsf{SIG}}$. Upon receiving $(\textsc{Verification-Key}, \mathsf{sid}, \mathsf{ssid}, \mathsf{vk}_j)$ from $\hat{\mathcal{F}}_{\mathsf{SIG}}$ 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 $\mathcal{Z}$, if $\mathcal{S}_j$ has not recorded $\mathsf{b}_j = 1$, then return $(\textsc{Error})$ to $\mathcal{Z}$ and halt. Otherwise, set $\mathsf{b}_j := 0$, 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 $b_j^{\mathsf{s}} := 0$, then proceeds as follows. Upon receiving $(\textsc{Elect}, \mathcal{S}_j, B)$ from the environment $\mathcal{Z}$,

   - If $\mathsf{b}_j = 1$ and $b_j^{\mathsf{s}} := 0$, send $(\textsc{CA-Register}, \mathcal{S}_j, B, \mathsf{vk}_j)$ to the functionality $\mathcal{F}^*_{\mathsf{rCA}}$. Upon receiving $(\textsc{CA-Registered}, \mathcal{S}_j, \mathsf{f})$ 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 $b_j^{\mathsf{s}} = 1$, 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 $(\textsc{Sign}, \mathsf{sid}, \mathsf{ssid}, \mathcal{S}_j, B, X)$ to the functionality $\hat{\mathcal{F}}_{\mathsf{SIG}}$. Upon receiving $(\textsc{Signature}, \mathsf{sid}, \mathsf{ssid}, (\mathcal{S}_j, B, X), \sigma)$ from $\hat{\mathcal{F}}_{\mathsf{SIG}}$, send $(\textsc{Signed}, (\mathcal{S}_j, B, X), \sigma)$ to the environment $\mathcal{Z}$.

Each player $P \in \{\mathcal{W}_1, \ldots, \mathcal{W}_n, \mathcal{S}_{n+1}, \ldots, \mathcal{S}_{n+\tilde{n}}\}$ proceeds as follows. Upon receiving $(\textsc{Stake-Verify}, (\mathcal{S}_j, B, X), \sigma)$ 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 $(\textsc{Verify}, \mathsf{sid}, \mathsf{ssid}, (\mathcal{S}_j, B, X), \sigma, \mathsf{vk}_j)$ to the functionality $\hat{\mathcal{F}}_{\mathsf{SIG}}$. Upon receiving $(\textsc{Verified}, \mathsf{sid}, \mathsf{ssid}, (\mathcal{S}_j, B, X), f)$ from the functionality $\hat{\mathcal{F}}_{\mathsf{SIG}}$, 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 6: Resource certification protocol π_{rCERT}.

Lemma 2.2. *Consider ϕ^*_{rCERT} described above and π_{rCERT} in Figure 6. It holds that the two ensembles $\mathsf{EXEC}^{\mathcal{F}^*_{\mathsf{rCERT}}}_{\phi^*_{\mathsf{rCERT}}, \mathcal{S}, \mathcal{Z}}$ and $\mathsf{EXEC}^{\mathcal{F}^*_{\mathsf{rCA}}, \hat{\mathcal{F}}_{\mathsf{SIG}}}_{\pi_{\mathsf{rCERT}}, \mathcal{A}, \mathcal{Z}}$ are perfectly indistinguishable.*

2.4 Blockchain Security Properties

Previously, several fundamental security properties for 1-hop blockchain protocols have been defined: *common prefix property* [19, 40], *chain quality property* [19], and *chain growth property* [25]. Intuitively, the chain growth property argues that the chains of honest players should grow linearly to the number of rounds. The common prefix property indicates the consistency of any two honest chains except the last κ blocks. The chain quality property, aims at expressing the number of honest blocks'contributions that are contained in a sufficiently long and continuous part of an honest chain. Specifically, for parameters $\ell \in \mathbb{N}$ and $\mu \in (0,1)$, the ratio of honest input contributions in a continuous part of an honest chain has a lower bounded μ.

We follow the same spirit to define the security properties for our 2-hop blockchain protocol. Since each valid PoS-chain and PoW-chain exist in our system as a pair having the same structure and grow at the same rate, we 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)**.** *Consider 2-hop blockchain protocol Π. The chain growth property $\mathcal{Q}_{\mathrm{cg}}$ states that for any honest PoS-holder $\mathcal{S} \in \{\mathcal{S}_{n+1}, \ldots, \mathcal{S}_{n+\tilde{n}}\}$ with the local PoS-chain $\tilde{\mathcal{C}}$ in round r and $\tilde{\mathcal{C}}'$ in round r' where $s = r' - r > 0$, 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)**.** *Consider 2-hop blockchain protocol Π. The common prefix property $\mathcal{Q}_{\mathrm{cp}}$ with parameter $\kappa \in \mathbb{N}$ states that for any two honest PoS-holders $\mathcal{S}_i$ in round r and $\mathcal{S}_j$ in round r' with the local PoS-chains $\tilde{\mathcal{C}}_i$, $\tilde{\mathcal{C}}_j$, respectively, in $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$ where $i, j \in \{n+1, \ldots, n+\tilde{n}\}, r \leq r'$, it holds that $\tilde{\mathcal{C}}_i[1, \ell_i] \preceq \tilde{\mathcal{C}}_j$ where $\ell_i = \mathsf{len}(\tilde{\mathcal{C}}_i) - \Theta(\kappa)$.*

Definition 2.5 (Chain Quality Property for PoS-chain)**.** *Consider 2-hop blockchain protocol Π. The chain quality property $\mathcal{Q}_{\mathrm{cq}}$ with parameters $\mu \in \mathbb{R}$ and $\ell \in \mathbb{N}$ states that for any honest PoS-holder $\mathcal{S} \in \{\mathcal{S}_{n+1}, \ldots, \mathcal{S}_{n+\tilde{n}}\}$ with PoS-chain $\tilde{\mathcal{C}}$ in $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$, it holds that for large enough ℓ consecutive PoS-blocks of $\tilde{\mathcal{C}}$ the ratio of honest blocks is at least μ.*

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)**.** *Consider 2-hop blockchain protocol Π. The chain quality property $\mathcal{Q}'_{\mathrm{cq}}$ with parameters $\mu' \in \mathbb{R}$ and $\ell \in \mathbb{N}$ states that for any honest PoW-miner $\mathcal{W} \in \{\mathcal{W}_1, \ldots, \mathcal{W}_n\}$ 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; after that, in Section 3.2, we present our main protocol; finally, we design a process to choose the best valid chain-pair among a set of chain-pairs in Section 3.3.

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

As in the original Bitcoin white paper [34], 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 abstract 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 $\langle \mathcal{S}, B, X \rangle$.

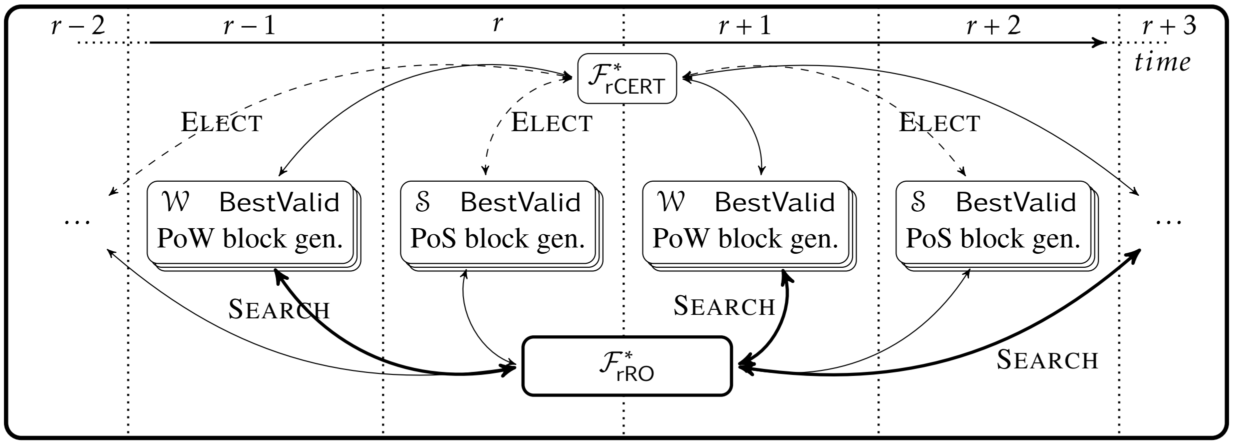
The structure of a PoS-chain is very similar to the PoW-chain, and many notations such as head, length, and subchain can be defined in the same way. We denote a PoS-chain by $\tilde{\mathcal{C}}$. We note that, in PoS-chain, payload is stored, and we use $\mathsf{payload}(\tilde{\mathcal{C}})$ to denote the information we store in $\tilde{\mathcal{C}}$. If $\mathsf{len}(\tilde{\mathcal{C}}) = \ell$, 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 secure against such attack, we need to combine two different resources: physical resource (i.e., computing power) and virtual resource (i.e., stake). Sequentially, we have two types of blockchains (PoW-chain and PoS-chain) corresponding to two types of rounds — PoW-round and PoS-round executing in turn — making 2-hop blockchain. Note that, in reality, one player could play both roles, PoW-miner and PoS-holder; however, without loss of generality, we treat the two roles separately. In order to tie them hard, the scheme maps each PoW-block to no more than 1 stakeholder. Only the stakeholder who has the privilege is able to generate the corresponding PoS-block of each PoW-block. Note that PoW-chains and PoS-chains existing in our system are represented as pairs, and each player locally stores a chain-pair. Therefore, the two member chains of each valid chain-pairs should have the same structure.

We now present our main protocol that describes the behaviour of PoW-miners and PoS-holders. The PoW and PoS executions vary slightly. On one hand, in the PoW execution, PoW-miners search for proof-of-work solutions via $\mathcal{F}^*_{\mathsf{rRO}}$. On the other hand, in the PoS execution, PoS-holders follow the growth of the PoW-chain and use that to extend the PoS-chain via $\mathcal{F}^*_{\mathsf{rCERT}}$. In general, PoW-miners and PoS-holders collect blockchain information from the network functionality $\mathcal{F}_{\mathsf{NET}}$, perform some validation and generating blocks, and then share their states with the network through $\mathcal{F}_{\mathsf{NET}}$. Please see Figure 7 for a pictorial illustration of our main protocol.

The protocol Π is parameterized by a content validation predicate $V(\cdot)$, which determines the proper structure of the information that is stored into the blockchain as in [19, 40]. 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 $\mathsf{len}(\mathcal{C}_{\mathsf{init}}) = \mathsf{len}(\tilde{\mathcal{C}}_{\mathsf{init}})$.



$r-2$
$r-1$
r
$r+1$
$r+2$
$r+3$
time
$\mathcal{F}^*_{\mathsf{rCERT}}$
ELECT
ELECT
ELECT
$\mathcal{W}$ BestValid PoW block gen.
$\mathcal{S}$ BestValid PoS block gen.
$\mathcal{W}$ BestValid PoW block gen.
$\mathcal{S}$ BestValid PoS block gen.
SEARCH
SEARCH
SEARCH
$\mathcal{F}^*_{\mathsf{rRO}}$

Figure 7: Round progression in our protocol.
This figure depicts the progression of protocol Π through round r. In each round, a 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.

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 9.) 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 requesting 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, $\mathcal{W}_i, h$). 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}, 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 $(\textsc{Elect}, \mathcal{S}_j, B)$. If this PoS-holder is the lucky stakeholder (with probability $\tilde{p}$), he would receive a message $(\textsc{Elected}, \mathcal{S}_j, \mathsf{f})$ 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 $(\textsc{Sign}, \mathcal{S}_j, B, X)$ 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 8 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 $\mathsf{BestValid}$, which is run locally by PoW-miners or PoS-holders, to select the best chain-pair. The $\mathsf{BestValid}$ 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 [19], and takes as input chain-pair set $\mathbb{C}'$. Intuitively, the process validates all chain-pair $\langle \mathcal{C}, \tilde{\mathcal{C}} \rangle$ in $\mathbb{C}'$, then finds the valid chain-pairs with the longest PoW-chain.

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

In more detail, $\mathsf{BestValid}$ proceeds as follows. On input a set of chains $\mathbb{C}'$ and an index *Type* where $\mathit{Type} \in \{\mathit{PoW\text{-}miner}, \mathit{PoS\text{-}holder}\}$. 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 be 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 $\mathit{Type} = \mathit{PoS\text{-}holder}$ (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, $\mathsf{BestValid}$ 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, $\mathsf{BestValid}$ 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 $1 \le i \le \mathsf{len}(\mathcal{C})$, in the PoW-chain $\mathcal{C}$, $\mathsf{BestValid}$ 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

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 $1 \leq i \leq n$, with local state $state_i$ proceeds as follows. Without lost of generality, we assume that the PoW-miners already registered to the functionality $\mathcal{F}^*_{\mathsf{rRO}}$. 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 many messages of the form $(\text{MESSAGE}, P, \langle \mathcal{C}, \tilde{\mathcal{C}} \rangle)$ from $\mathcal{F}_{\mathsf{NET}}$ for any party $P \in \{\mathcal{W}_1, \ldots, \mathcal{W}_n, \mathcal{S}_{n+1}, \ldots, \mathcal{S}_{n+\tilde{n}}\}$, let $\mathbb{C}$ be the set of all chain-pair collected from $\mathcal{F}_{\mathsf{NET}}$, then compute $\langle \mathcal{C}_{\mathsf{best}}, \tilde{\mathcal{C}}_{\mathsf{best}} \rangle := \mathsf{BestValid}(\mathbb{C} \cup \langle \mathcal{C}_i, \tilde{\mathcal{C}}_i \rangle, \textit{PoW-miner})$, and then set $\mathcal{C}_i := \mathcal{C}_{\mathsf{best}}$ and $\tilde{\mathcal{C}}_i := \tilde{\mathcal{C}}_{\mathsf{best}}$.
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 $n + 1 \leq j \leq n + \tilde{n}$, with local state $state_j$, 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 many messages of the form $(\text{MESSAGE}, P, \langle \mathcal{C}, \tilde{\mathcal{C}} \rangle)$ from $\mathcal{F}_{\mathsf{NET}}$ for any party $P \in \{\mathcal{W}_1, \ldots, \mathcal{W}_n, \mathcal{S}_{n+1}, \ldots, \mathcal{S}_{n+\tilde{n}}\}$, let $\mathbb{C}$ be the set of all chain-pair collected from $\mathcal{F}_{\mathsf{NET}}$, then compute $\langle \mathcal{C}_{\mathsf{best}}, \tilde{\mathcal{C}}_{\mathsf{best}} \rangle := \mathsf{BestValid}(\mathbb{C} \cup \langle \mathcal{C}_j, \tilde{\mathcal{C}}_j \rangle, \textit{PoS-holder})$, and then set $\mathcal{C}_j := \mathcal{C}_{\mathsf{best}}$ and $\tilde{\mathcal{C}}_j := \tilde{\mathcal{C}}_{\mathsf{best}}$.
2. *Attempt to extend PoS-chain:* Consider there is a new PoW-block B in $\mathcal{C}_{\mathsf{best}}$.
   - *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 8: Our main protocol $\Pi = (\Pi^{\mathsf{w}}, \Pi^{\mathsf{s}})$ in the $\{\mathcal{F}^*_{\mathsf{rRO}}, \mathcal{F}^*_{\mathsf{rCERT}}, \mathcal{F}_{\mathsf{NET}}\}$-hybrid model with respect to the local process $\mathsf{BestValid}$ (See Figure 9).

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 requesting $\mathcal{F}^*_{\mathsf{rCERT}}$), and (3) the signature generated by that stakeholder is verifiable (by requesting $\mathcal{F}^*_{\mathsf{rCERT}}$).

After the validation, the best valid chain-pair is the one with the longest PoW-chain. Please refer to Figure 9 for more details.

PROCESS $\mathsf{BestValid}$

The process $\mathsf{BestValid}$ 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 $(\text{COMPUTE}, \mathcal{C}[\ell-1], \tilde{\mathcal{C}}[\ell-1])$ to $\mathcal{F}^*_{\mathsf{rRO}}$ and then receive $(\text{COMPUTED}, h)$. If $h \neq h_\ell$, remove this chain-pair from $\mathbb{C}'$.
   - *Verify the PoW solution in* $\mathcal{C}[\ell]$*:* Send message $(\text{RO-VERIFY}, \mathcal{C}[\ell])$ to the functionality $\mathcal{F}^*_{\mathsf{rRO}}$, and then receive $(\text{RO-VERIFIED}, h')$. 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 $(\text{COMPUTE}, \mathcal{C}[i-1], \tilde{\mathcal{C}}[i-1])$ to $\mathcal{F}^*_{\mathsf{rRO}}$ and then receive $(\text{COMPUTED}, h)$.
     - *Verify the PoW solution in* $\mathcal{C}[i]$*:* Send message $(\text{RO-VERIFY}, \mathcal{C}[i])$ to the functionality $\mathcal{F}^*_{\mathsf{rRO}}$, and then receive $(\text{RO-VERIFIED}, h')$.

     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 $(\text{STAKE-VERIFY}, (B, X, \mathcal{S}_j), \sigma)$ to the functionality $\mathcal{F}^*_{\mathsf{rCERT}}$. Upon receiving message $(\text{STAKE-VERIFIED}, (B, X, \mathcal{S}_j), f)$ 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 $\langle \mathcal{C}_{\mathsf{best}}, \tilde{\mathcal{C}}_{\mathsf{best}} \rangle \in \mathbb{C}'$ with the longest PoW-chain. Then set $\langle \mathcal{C}_{\mathsf{best}}, \tilde{\mathcal{C}}_{\mathsf{best}} \rangle$ as the output.

Figure 9: The chain set validation process $\mathsf{BestValid}$.

Intuitively, $\mathsf{BestValid}$ 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.

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-pair from two chain-pairs of equal length. While there is work that show the advantages of choosing a chain randomly (viz. [18]), we follow the simple strategy considered in [19]; in which the best chain-pair is the one with the PoW-chain that is lexicographically the smallest. If* *two chain-pairs have same length, and the PoW-chains are same, we compare the PoS-chains with the same tie breaking mechanism for PoW-chains.*

4 Security Analysis

Our 2-hop blockchain can be viewed as a natural generalization of Nakamoto's famous 1-hop blockchain [34]. The security analysis of the 2-hop blockchain here is inspired and influenced by previous analysis of Nakamoto's 1-hop protocol [19, 40]. In order to improve the readability, in general, we will keep consistency of notations. We use the original letters to denote the PoW parameters for the first hop, e.g., α; we use letters with tilde to denote the PoS parameters for the second hop, e.g., $\tilde{\alpha}$; finally, we use letters with hat to denote the collective parameters for both hops, e.g., $\hat{\alpha}$.

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

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

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

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

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

It is obvious that we have a parameter $\hat{\alpha} = \alpha\tilde{\alpha}$ which is the probability that honest parties find a new PoW-block and is mapped to an honest PoS-holder in a round. We also have $\hat{\beta} = \beta\tilde{\beta}$ it the expected number that malicious parties find new PoW-blocks and are mapped to malicious PoS-holders in a round. We say $\hat{\alpha}$ and $\hat{\beta}$ are **collective** resources for honest parties and malicious parties respectively.

We consider the network delay model as in [40]. We will show in section 4.3, $\hat{\gamma} = \frac{\hat{\alpha}}{1+2\Delta\hat{\alpha}}$ can be viewed as a "discounted" version of $\hat{\alpha}$ due to the fact that the messages sent by honest parties can be delayed by Δ rounds; $\hat{\gamma}$ corresponds to the "effective" honest collective resource. We also assume $(\alpha+\beta)\Delta \ll 1$.

We are now ready to state our main theorems.

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

Theorem 4.2 (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 $\mathcal{S} \in \{\mathcal{S}_{n+1}, \ldots, \mathcal{S}_{n+\tilde{n}}\}$ with PoS-chain $\tilde{\mathcal{C}}$ in $\mathsf{EXEC}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}$, the probability that, for large enough ℓ consecutive PoS-blocks of $\tilde{\mathcal{C}}$ which are generated in s rounds, the ratio of honest blocks is no less than $\mu = 1 - (1+\delta)\frac{(\alpha+\beta)\tilde{\beta}}{\hat{\gamma}}$ is at least $1 - e^{-\Omega(\ell)}$.*

Theorem 4.3 (Common Prefix for PoS-chain)**.** *We assume $\hat{\alpha} = \hat{\lambda}(\alpha+\beta)\tilde{\beta}$ and $\hat{\lambda} > 1$. For any $\delta > 0$, consider protocol $\Pi = (\Pi^{\mathsf{w}}, \Pi^{\mathsf{s}})$ in Section 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 $r \leq r'$ and $i, j \in \{n+1, \ldots, n+\tilde{n}\}$, the probability that $\tilde{\mathcal{C}}_i[1, \ell_i] \preceq \tilde{\mathcal{C}}_j$ where $\ell_i = \mathsf{len}(\tilde{\mathcal{C}}_i) - \Theta(\kappa)$ is at least $1 - e^{-\Omega(\kappa)}$.*

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 Analysis Ideas

In this section, we informally introduce the analysis idea, the whole proof can be found in Appendix **??**. In our scheme, there are two types of players, PoW-miners and PoS-holders (stakeholders). Both PoW-miners and PoS-holders can be honest or malicious. In order to extend the pair of blockchains, a PoW-miner needs to generate a PoW-block first, and then the corresponding stakeholder will sign this block (via $\mathcal{F}^{*}_{\mathsf{rCERT}}$) . We note that, our security analysis mainly focuses on PoS-chain, and the analysis for PoW-chain is followed from PoS-chain's. Consider that players may be honest or malicious, we have 4 types of compositions.

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

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

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

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

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

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

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

In the real execution, the probability of **Case 1** will not be smaller than that in the hybrid execution. The message from malicious players will not decrease the chain growth that contributed by honest players. Therefore, the real chain growth rate is not worse than that in the hybrid execution.

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

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

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

We say a round is an honest successful round if there is at least one honest player generating a new block. An honest successful round is guaranteed by α. We consider a special case of an honest successful round where only one honest miner generates a new block. We say it is a pure successful round. In the flat model, we can view the hash queries are independent. Thus, the success of one query will not effect other queries. In a successful round, there is at least one successful query. We get rid of this successful query and consider the others. The probability of a successful query of the others is still no more than α. Thus, the probability of pure successful round is $\alpha(1 - \alpha)$. It approximates to α if we assume $\alpha \ll 1$.

We assume the malicious players don't send any new block in the following case first. We consider the case that a round is a silent round and also a pure successful round. This means the successful PoW-miner will generate a unique public chain-pair with the longest PoW-chain for all honest players. If in the following

2Δ rounds, no honest PoW-miner sends a new block, then all honest players would take the same best chain-pair. This means all honest players will be convergent. The only exception is that the malicious players generate blocks to prevent this convergence. However, by our assumption, the malicious players cannot generate enough blocks to do this.

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

Definition 4.6 (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 4.7. *As discussed in previous section, a PoS-block may be invalid because the delay of the messages on the network. Here, the honest stake successful round also only consider that the real valid PoS-block is generated.*

Definition 4.8 (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 4.9 (Hidden chain-pair). *We say a chain-pair* $\langle \mathcal{C}, \tilde{\mathcal{C}} \rangle$ *is a hidden chain-pair, if it is not known to any honest players.*

Definition 4.10 (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 4.11 (Public chain-pair). *We say a chain-pair* $\langle \mathcal{C}, \tilde{\mathcal{C}} \rangle$ *is public, if is known to all honest players.*

Definition 4.12 (Best Public PoS-chain). *We say a PoS-chain* $\tilde{\mathcal{C}}$ *is the best public PoS-chain, if a)* $\tilde{\mathcal{C}}$ *has been received by all of the honest players, b)* $\tilde{\mathcal{C}}$ *is the PoS-chain of the best chain-pair among all of the public chain-pairs.*

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

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

Lemma 4.13. *Let* $\alpha \ll 1$*. For any* r*, the probability that round* r *is a pure successful round is at least* $\alpha - \alpha^2$*.*

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

Putting them together, the probability that round r is a pure successful round is at least

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

$\square$

Following a similar argument, the number of valid malicious PoW-blocks that the malicious miners will generate in a PoS-round is $\beta\tilde{\beta}$ on average. We remark that, in an honest stake successful round, the honest PoS-holder can generate a valid PoS-block for the best local chain chain-pair and send to all of the parties. The malicious stakeholders can append the valid malicious PoS-blocks to their (hidden) chain-pair and send the new solution to some (not necessary all) honest parties.

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

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

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

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

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

As the discussion, $\hat{\alpha} = \alpha\tilde{\alpha}$ and $\hat{\beta} = \beta\tilde{\beta}$ are the collective probabilities of **Case 1** and **Case 2**. We define them as the collective resources of the honest and malicious parties, respectively. We remark that, the malicious players may choose different strategy on the 4 cases for different purposes. For example, if they want to prevent the chain growth, they may discard as more PoW-blocks as possible. If they want to decrease the chain quality, they may recycle all of the PoW-blocks that they can "sign" (via $\mathcal{F}^*_{\mathsf{rCERT}}$) to generate more PoS-blocks.

4.3 Analysis with Bounded Delay

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

As discussed, if any message can be delayed for Δ rounds, a new pair of blocks can be delayed in at most 2Δ rounds. The intuition is that the delay of messages will 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 will introduce the "effective" honest collective resources $\hat{\gamma}$ to capture this intuition later.

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

Definition 4.14 (Silent Round). *We say round r is a silent round, if no PoW-miners (both honest and malicious) publish any new PoW-chain in the 2Δ previous rounds of round r to any honest party.*

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

Lemma 4.15. *Let $2(\alpha + \beta)\Delta \ll 1$. For $r > 0$ and any $\delta > 0$, the probability that round r is a silent round is $1 - 2(1 + \delta)(\alpha + \beta)\Delta$ with probability at least $1 - e^{-\Omega(t)}$, where $t > 0$.*

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

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

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

. □

4.3.1 Hybrid expriment

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

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

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

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

1. The randomness is fixed to σ as in $\mathsf{HYB}_r(\sigma)$,

2. $\mathcal{F}_{\mathsf{NET}}$ delays messages generated by *honest* PoW-miners to exact Δ rounds if the new PoW-block is mapped to an honest stakeholder,

3. $\mathcal{F}_{\mathsf{NET}}$ delays all messages generated by *honest* PoS-holders to exact Δ rounds,

4. Remove all new messages sent by the adversary to honest players, and delay currently undelivered messages from corrupted parties to the maximum of Δ rounds,

5. Whenever some message is being delayed, no *honest* PoW-miners query the functionality $\mathcal{F}^{*}_{\mathsf{rRO}}$ until the message is delivered.

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

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

Lemma 4.16. *For all $\sigma, r, t > 0$, given two executions $\mathsf{REAL}(\sigma)$ and $\mathsf{HYB}_r(\sigma)$. Let $s = r + t$. For any honest PoS-holder $\mathcal{S}_j$ at round s, let $\tilde{\mathcal{C}}_{s,j}$ denote the PoS-chain of $\mathcal{S}_j$ at round s in the execution $\mathsf{REAL}(\sigma)$ and $\tilde{\mathcal{C}}'_{s,j}$ denote the PoS-chain of $\mathcal{S}_j$ at round s in the $\mathsf{HYB}_r(\sigma)$. We then have $\mathsf{len}(\tilde{\mathcal{C}}_{s,j}) \geq \mathsf{len}(\tilde{\mathcal{C}}'_{s,j})$.*

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

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

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

□

4.3.2 Analysis in the worst delay setting

As mentioned earlier, the malicious players can delay the messages on the network in at most Δ rounds. This will make some efforts of honest players be wasted. This means the network delay will discount the collective resources of honest players, the following lemma will measure the discount in the execution of $\mathsf{HYB}_r(\sigma)$.

Lemma 4.17. *Consider $\mathsf{HYB}_r(\sigma)$. Let $\alpha, \tilde{\alpha} > 0$. We assume that the malicious players will delay any message for at most Δ rounds. Let $\hat{\gamma}$ be the actual probability that a round $s > r$ is an honest successful stake round, we have $\hat{\gamma} = \frac{\hat{\alpha}}{1+2\Delta\hat{\alpha}}$.*

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

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

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

Lemma 4.18. *Consider execution* $\mathsf{HYB}_r(\sigma)$. *For any honest stake successful round* $s > r$, $\mathsf{len}(\mathsf{view}_s) - \mathsf{len}(\mathsf{view}_{s-2\Delta}) \geq 1$.

Proof. By Definition 4.6, there is at least one honest PoS-holder producing a PoS-block at round s. Let $\tilde{\mathcal{C}}_{s-2\Delta}$ be the PoS-chain that is extended by the PoS-holder at round $s - 2\Delta$. We have $\mathsf{len}(\tilde{\mathcal{C}}_{s-2\Delta}) \geq \mathsf{len}(\mathsf{view}_{s-2\Delta})$. Let $\tilde{\mathcal{C}}_s$ be the PoS-chain that is extended by the PoS-holder at round s. At the end of round s, all honest players will receive the extended chain, we have $\mathsf{len}(\tilde{\mathcal{C}}_s) = \mathsf{len}(\tilde{\mathcal{C}}_{s-2\Delta}) + 1$. Thus, we have $\mathsf{len}(\mathsf{view}_s) \geq \mathsf{len}(\tilde{\mathcal{C}}_s)$. Putting them together, we have $\mathsf{len}(\mathsf{view}_s) - \mathsf{len}(\mathsf{view}_{s-2\Delta}) \geq 1$. □

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

Proof. Let r_k be the kth honest stake successful round where $r < r_k < r + t$ and $1 \leq k \leq h$. From Lemma 4.18, we have $\mathsf{len}(\mathsf{view}_{r_k}) - \mathsf{len}(\mathsf{view}_{r_k - \Delta}) \geq 1$.

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

4.4 Achieving the Chain Growth Property

We here demonstrate that our protocol satisfies the chain growth property for PoS-chain (Definition 2.3). The concrete statement to be proved can be found in Theorem 4.1. We first prove that our blockchain protocol achieves the chain growth property in the execution $\mathsf{HYB}_r(\sigma)$ before moving to the main theorem.

Lemma 4.20. *Consider the execution* $\mathsf{HYB}_r(\sigma)$. *For any $\delta > 0$, during any t consecutive rounds after round r, the number of honest successful rounds is $(1 - \delta)\hat{\gamma} t$ with probability at least $1 - e^{-\Omega(t)}$.*

Proof. We will prove that honest stake successful rounds will happen following the honest collective resources. From Lemma 4.17, in any t consecutive rounds, the number of honest stake successful rounds is $\hat{\gamma} t$ on average. Let X be the number of honest stake successful rounds in the t consecutive rounds. By Chernoff bound, we have $\Pr[X \leq (1 - \delta)\hat{\gamma} t] \leq e^{-\delta^2 \hat{\gamma} t/2}$.

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

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

Lemma 4.21. *Consider the execution* $\mathsf{HYB}_r(\sigma)$. *For any $\delta > 0$, and for any honest PoS-holder $\mathcal{S}$ with the best PoS-chain $\tilde{\mathcal{C}}_r$ and $\tilde{\mathcal{C}}_{r'}$ in round r and r', respectively, where $t = r' - r \gg \Delta$, the probability that* $\mathsf{len}(\tilde{\mathcal{C}}_{r'}) - \mathsf{len}(\tilde{\mathcal{C}}_r) \geq g \cdot t$ *where $g = (1 - \delta)\hat{\gamma}$ is at least $1 - e^{-\Omega(t)}$.*

Proof. For $\mathcal{S}$ is honest, $\tilde{\mathcal{C}}_r$ will be received by all honest players no later than round $r + \Delta$. We have $\mathsf{len}(\tilde{\mathcal{C}}_r) \leq \mathsf{len}(\mathsf{view}_{r+\Delta})$. For $t \gg \Delta$, we consider $t \approx t - \Delta$ for simplicity. From Lemma 4.20, in any t consecutive rounds the number of honest successful round is more than $(1 - \delta)\gamma t$ with the probability at least $1 - e^{-\Omega(t)}$. Together with Lemma 4.18 and Corollary 4.19, we have $\mathsf{len}(\mathsf{view}_{r'}) - \mathsf{len}(\mathsf{view}_{r+\Delta}) \geq \ (1 - \delta)\hat{\gamma} t$.

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

Reminder of Theorem 4.1. *For any $\delta > 0$, consider protocol $\Pi = (\Pi^{\mathsf{w}}, \Pi^{\mathsf{s}})$ in Section 3.2. For any honest PoS-holder $\mathcal{S} \in \{\mathcal{S}_{n+1}, \ldots, \mathcal{S}_{n+\tilde{n}}\}$ with the local PoS-chain $\tilde{\mathcal{C}}$ in round r and $\tilde{\mathcal{C}}'$ in round r' where $t = r' - r > 0$, in $\mathsf{EXEC}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}$, the probability that $\mathsf{len}(\tilde{\mathcal{C}}') - \mathsf{len}(\tilde{\mathcal{C}}) \geq g \cdot t$ is at least $1 - e^{-\Omega(t)}$ where $g = (1-\delta)\hat{\gamma}$.*

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

From Lemma 4.21, we have $\Pr[\mathsf{len}(\tilde{\mathcal{C}}'_\Delta) \geq \mathsf{len}(\tilde{\mathcal{C}}_\Delta) + g \cdot t] \geq 1 - e^{-\Omega(t)}$ where $t = r' - r$, in $\mathsf{HYB}_r(\sigma)$.

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

By Lemma 4.16, we have $\mathsf{len}(\tilde{\mathcal{C}}') \geq \mathsf{len}(\tilde{\mathcal{C}}'_\Delta)$.

It follows that,

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

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

4.5 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 resources. Finally, we demonstrate that the ratio of honest PoS-blocks in an honest player's PoS-chain is under a suitable lower bound in a sufficient number of rounds with an overwhelming probability.

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

Lemma 4.22. *Consider in $\mathsf{REAL}(\sigma)$. For any $\delta > 0$, let Z be the number of rounds which generate ℓ blocks, we then have $\Pr[Z > c\ell] > 1 - e^{-\Omega(\ell)}$, where $c \gg 1$.*

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

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

Now we consider the contribution of PoS-blocks from honest players in some consecutive rounds. If the adversarial players want to contribute more PoS-blocks on the chain, they will try to generate more PoS-blocks and beat the PoS-blocks from honest players in the competition. Thus, the worst case is the adversarial players make use of all the computing power from both honest and malicious miners to generate PoS-blocks and win all of the competition. We recall the 4 cases in Section 4.2. The adversarial players can make use of **Case 2** and **Case 3** to generate PoS-blocks. The honest players will use **Case 1** to generate PoS-blocks. First, we will prove the chain quality property in t consecutive rounds.

Lemma 4.23. *Consider in* $\mathsf{REAL}(\sigma)$ *and consider* ℓ *PoS-blocks of* $\tilde{\mathcal{C}}$ *that are generated in consecutive rounds from* r *to* $r+t$. *We assume* $\hat{\gamma} = \hat{\lambda}(\alpha+\beta)\tilde{\beta}$ *where* $\hat{\lambda} > 1$ *and any* $\delta > 0$. *Then for any honest PoS-holder* $\mathcal{S}$ *with PoS-chain* $\tilde{\mathcal{C}}$, *we have* $\Pr[\mu \geq 1-(1+\delta)\frac{(\alpha+\beta)\tilde{\beta}}{\hat{\gamma}}] > 1-e^{-\Omega(t)}$, *for any* $\delta > 0$, *where* μ *is the ratio of honest blocks the PoS-chain* $\tilde{\mathcal{C}}$.

Proof. Consider ℓ consecutive PoS-blocks of $\tilde{\mathcal{C}}$ that are generated from round r to round $r+t$. From Theorem 4.1, we have $\Pr[\ell \geq (1-\delta^*)\hat{\gamma}\cdot t] \geq 1-e^{-\Omega(t)}$ for some $\delta^* > 0$.

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

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

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

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

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

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

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

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

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

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

Reminder of Theorem 4.2. *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* $\mathcal{S} \in \{\mathcal{S}_{n+1}, \ldots, \mathcal{S}_{n+\tilde{n}}\}$ *with PoS-chain* $\tilde{\mathcal{C}}$ *in* $\mathsf{EXEC}_{(\Pi^{\mathsf{w}},\Pi^{\mathsf{s}}),\mathcal{A},\mathcal{Z}}$, *the probability that, for large enough* ℓ *consecutive PoS-blocks of* $\tilde{\mathcal{C}}$ *which are generated in* s *rounds, the ratio of honest blocks is no less than* $\mu = 1-(1+\delta)\frac{(\alpha+\beta)\tilde{\beta}}{\hat{\gamma}}$ *is at least* $1-e^{-\Omega(\ell)}$.

Proof. Let t be the rounds that the ℓ blocks are generated. From Lemma 4.22, we have $\Pr[t > c\ell] > 1-e^{-\Omega(\ell)}$.

From Lemma 4.23, the ratio of honest PoS-blocks in t consecutive rounds with ℓ PoS-blocks is $\mu \geq 1-(1+\delta)\frac{(\alpha+\beta)\tilde{\beta}}{\hat{\gamma}}$ with probability at least $1-e^{-\Omega(t)}$.

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

4.6 Achieving the Common Prefix Property

We now turn our attention to proving the common prefix property for PoS-chain (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, we assume there is no new block being generated in most rounds. This implies no new chain will lead the honest players take divergent view in most rounds. The honest players will try to be convergent after some silent rounds. If the adversary players want to keep them be divergent, they must send new blocks in silent rounds. The adversary players don't have enough resources to do that.

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

Lemma 4.24. *Let $\hat{\gamma} = \hat{\lambda}\hat{\beta}$ and $\hat{\lambda} > 1$. For any $\delta > 0$, consider the execution $\mathsf{REAL}(\sigma)$. Let $\tilde{\mathcal{C}}$ be the PoS-chain of the best public chain-pair in round r. Let $\tilde{\mathcal{C}}'$ be the PoS-chain of a hidden valid chain-pair in round r. Let ℓ be the length of the hidden part of $\tilde{\mathcal{C}}'$. We have* $\Pr[\mathsf{len}(\tilde{\mathcal{C}}) > \mathsf{len}(\tilde{\mathcal{C}}')] > 1 - e^{\Omega(\ell)}$.

Proof. Let round $s = r - t$ be the round that last public block in $\tilde{\mathcal{C}}'$ is generated . From Lemma 4.22, ℓ hidden blocks need t rounds to generate with probability at least $1 - e^{\Omega(\ell)}$. That is, $\Pr[t > c\ell] > 1 - e^{\Omega(\ell)}$.

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

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

From Theorem 4.1, during t rounds the best public PoS-chain will increase $Y > (1-\delta)\hat{\gamma}t$ blocks with probability at least $1 - e^{-\Omega(t)}$. For $\hat{\gamma} = \hat{\lambda}\hat{\beta}$, we have

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

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

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

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

□

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

- We assume $\alpha + \beta \ll 1$, that means in most rounds, players will not generate block.

- If only honest players broadcast a new PoW-block, in 2Δ silent rounds, all honest players will take the same best chain-pair (or PoS-chain), unless adversarial players send new valid blocks.

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

Lemma 4.25. *Let $\hat{\alpha} = \hat{\lambda}(\alpha+\beta)\tilde{\beta}$, $\hat{\lambda} > 1$, $(\alpha+\beta)\Delta \ll 1$. Assume $0 < \delta, \delta', \delta'', \delta''' < 1$, consider the execution* $\mathsf{REAL}(\sigma)$. *Except with probability $e^{-\Omega(t)}$, there does not exist round $r \leq r'$ and PoS-holders $\mathcal{S}_i$, $\mathcal{S}_j$ such that $\mathcal{S}_i$ is honest at r, $\mathcal{S}_j$ is honest at r' and $\tilde{\mathcal{C}}_i^r$ and $\tilde{\mathcal{C}}_j^{r'}$ diverge at round $s = r - t$.*

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

Let X' be the number of pure successful rounds in t round. By Lemma 4.13, X' is $(\alpha - \alpha^2)t$ on average. By Chernoff bound, we have

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

Let X'' be the number rounds which are both pure successful round and silent round. A round is pure successful round is independent with the event it is silent round. By Lemma 4.15, we have

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

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

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

A round k is also independent with the event round $k+2\Delta$ is a silent round. Also by Lemma 4.15, we have

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

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

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

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

Let Z' be the number of blocks generated by adversarial players in t consecutive rounds. We have $Z' = (\alpha+\beta)\tilde{\beta}t$ on average. By Chernoff bound, we have $\Pr[Z' > (1+\delta)(\alpha+\beta)\tilde{\beta}t] < e^{-\Omega(t)}$. From Lemma 4.24, before round s, the adversarial players can hide at most κ blocks with an overwhelming probability in κ. Let $Z = Z' + \kappa$, the adversarial players can use Z new blocks to prevent the Y convergence opportunities. If t is large enough, we can have δ' that $\Pr[Z > (1+\delta')(\alpha+\beta)\tilde{\beta}t] < e^{-\Omega(t)}$.

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

$$\begin{aligned} Y - Z &> (1-\delta''')\alpha\tilde{\alpha}t - (1+\delta')(\alpha+\beta)\tilde{\beta}t \\ &= ((1-\delta''')\hat{\lambda} - (1+\delta'))(\alpha+\beta)\tilde{\beta}t \\ &> 0 \end{aligned} \tag{2}$$

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

We are now ready to prove the main theorem which asserts that our protocol achieves the common-prefix property with an overwhelming probability in the security parameter κ. The theorem is formally given as follows.

Reminder of Theorem 4.3. *We assume $\hat{\alpha} = \hat{\lambda}(\alpha+\beta)\tilde{\beta}$ and $\hat{\lambda} > 1$. For any $\delta > 0$, consider protocol $\Pi = (\Pi^{\mathsf{w}}, \Pi^{\mathsf{s}})$ in Section 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 $r \le r'$ and $i, j \in \{n+1, \dots, n+\tilde{n}\}$, the probability that $\tilde{\mathcal{C}}_i[1,\ell_i] \preceq \tilde{\mathcal{C}}_j$ where $\ell_i = \mathsf{len}(\tilde{\mathcal{C}}_i) - \Theta(\kappa)$ is at least $1-e^{-\Omega(\kappa)}$.*

Proof. From Lemma 4.25, the probability that $\tilde{\mathcal{C}}_i$ and $\tilde{\mathcal{C}}_j$ diverge at round $s = r - t$ is at most $e^{-\Omega(t)}$.

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

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] TwinsCoin source code. `https://bitbucket.org/TwCoin/twinscoin`.

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

[3] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, CCS '93, pages 62–73. ACM, 1993.

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

[5] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld. Proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract]. *SIGMETRICS Perform. Eval. Rev.*, 42(3):34–37, Dec. 2014.

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

[7] 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.

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

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

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

[11] 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.

[12] R. Canetti and T. Rabin. Universal composition with joint state. In D. Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 265–281. Springer, Heidelberg, Aug. 2003.

[13] 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.

[14] A. Chepurnoy, T. Duong, L. Fan, and H.-S. Zhou. Twinscoin: A cryptocurrency via proof-of-work and proof-of-stake. In *Cryptology ePrint Archive, Report 2017/232*, 2017. https://eprint.iacr.org/2017/232.

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

[16] 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.

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

[18] 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.

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

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

[21] D. Hofheinz and J. Müller-Quade. Universally composable commitments using random oracles. In M. Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 58–76. Springer, Heidelberg, Feb. 2004.

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

[23] J. Katz, A. Miller, and E. Shi. Pseudonymous broadcast and secure computation from cryptographic puzzles. Cryptology ePrint Archive, Report 2014/857, 2014. http://eprint.iacr.org/2014/857.

[24] 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.

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

[26] 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.

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

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

[29] J. Kwon. Tendermint: Consensus without mining. 2014. https://tendermint.com/static/docs/tendermint.pdf.

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

[31] 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.

[32] 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.

[33] 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.

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

[35] 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.

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

[37] T. Okamoto. An efficient divisible electronic cash scheme. In D. Coppersmith, editor, *CRYPTO'95*, volume 963 of *LNCS*, pages 438–451. Springer, Heidelberg, Aug. 1995.

[38] T. Okamoto and K. Ohta. Universal electronic cash. In J. Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 324–337. Springer, Heidelberg, Aug. 1992.

[39] 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.

[40] R. Pass, L. Seeman, and A. Shelat. Analysis of the blockchain protocol in asynchronous networks. In *EUROCRYPT*, 2017. https://eprint.iacr.org/2016/454.

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

[42] 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.

[43] 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.

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

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

A Supporting Material for Our Model

A.1 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 the blockchain protocol has been stably run for a while; 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 10. In [10], 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.

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 $(\text{CA-REGISTER}, \mathcal{S}_j, B, \mathsf{vk}_j)$ 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 10.

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

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

Registration. Upon receiving a message $(\text{CA-REGISTER}, \mathcal{S}_j, B, \mathsf{vk}_j)$ from party $\mathcal{S}_j \in \{\mathcal{S}_{n+1}, \ldots, \mathcal{S}_{n+\tilde{n}}\}$ where $\mathsf{vk}_j \in \{0,1\}^{\mathrm{poly}(\kappa)}$, it then passes the message to the adversary. Upon receiving a message $(\text{CA-REGISTERED}, \mathcal{S}_j)$ from the adversary,

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

Retrieve: Upon receiving $(\text{RETRIEVE}, \mathcal{S}_j, B)$ from a player $P \in \{\mathcal{W}_1, \ldots, \mathcal{W}_n, \mathcal{S}_{n+1}, \ldots, \mathcal{S}_{n+\tilde{n}}\}$, send $(\text{RETRIEVE}, \mathcal{S}_j, P)$ to the adversary, and wait for a message $(\text{RETRIEVED}, \mathcal{S}_j, P)$ from the adversary. Then, if there is a recorded entry $(\mathcal{S}_j, B, \mathsf{vk}_j)$, output $(\text{RETRIEVED}, \mathsf{vk}_j)$ to P. Else, output $(\text{RETRIEVED}, \perp)$ to P.

Figure 10: Resource certificate authority functionality $\mathcal{F}^*_{\mathsf{rCA}}$.

There are multiple ways to instantiate $\mathcal{F}^*_{\mathsf{rCA}}$. Intuitively, in our main application scenario, $\mathcal{F}^*_{\mathsf{rCA}}$ is implemented by a protocol in $\{\hat{\mathcal{F}}_{\mathsf{CA}}, \mathcal{F}_{\mathsf{RO}}\}$-hybrid model, and then multi-session certificate authority functionality

$\hat{\mathcal{F}}_{\mathsf{CA}}$ can be implemented by an already "mature" blockchain (i.e., Bitcoin). 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_{rCA} as the protocol in the $\{\hat{\mathcal{F}}_{\mathsf{CA}}, \mathcal{F}_{\mathsf{RO}}\}$-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_{rCA} execute the protocol and then pass the outputs to the environment. The protocol π_{rCA} is described in Figure 11.

PROTOCOL π_{rCA}

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

1. Upon receiving $(\text{CA-REGISTER}, \mathcal{S}_j, B, \mathsf{vk}_j)$ from the environment $\mathcal{Z}$, PoS-holder $\mathcal{S}_j \in \{\mathcal{S}_{n+1}, \ldots, \mathcal{S}_{n+\tilde{n}}\}$ sends (B, vk_j) to the functionality $\mathcal{F}_{\mathsf{RO}}$ and receives h.
   - If $h > \tilde{\mathtt{D}}$ where $\tilde{\mathtt{D}} = \tilde{p} \cdot 2^\kappa$, then set $\mathsf{f} := 0$, and pass $(\text{CA-REGISTERED}, \mathcal{S}_j, \mathsf{f})$ to the environment.
   - Else, if $h \le \tilde{\mathtt{D}}$, send $(\text{REGISTER}, \mathsf{sid}, \mathsf{ssid}, \mathcal{S}_j, B, \mathsf{vk}_j)$ for some $\mathsf{sid}, \mathsf{ssid}$ to the functionality $\hat{\mathcal{F}}_{\mathsf{CA}}$. Upon receiving $(\text{REGISTERED}, \mathsf{sid}, \mathsf{ssid}, \mathcal{S}_j, B, \mathsf{vk}_j)$ from $\hat{\mathcal{F}}_{\mathsf{CA}}$, 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, party $P \in \{\mathcal{W}_1, \ldots, \mathcal{W}_n, \mathcal{S}_{n+1}, \ldots, \mathcal{S}_{n+\tilde{n}}\}$ send $(\text{RETRIEVE}, \mathsf{sid}, \mathsf{ssid}, \mathcal{S}_j, B)$ to the functionality $\hat{\mathcal{F}}_{\mathsf{CA}}$ and then receive the output $(\text{RETRIEVED}, \mathsf{sid}, \mathsf{ssid}, \mathsf{vk}_j)$. Then send $(\text{RETRIEVED}, \mathsf{vk}_j)$ to the environment.

Figure 11: Resource certificate authority protocol π_{rCA}.

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

Lemma A.1. *Consider ϕ^*_{rCA} described above and described above and π_{rCA} in Figure 11. It holds that the two ensembles $\mathsf{EXEC}^{\mathcal{F}^*_{\mathsf{rCA}}}_{\phi^*_{\mathsf{rCA}}, \mathcal{S}, \mathcal{Z}}$ and $\mathsf{EXEC}^{\hat{\mathcal{F}}_{\mathsf{CA}}, \mathcal{F}_{\mathsf{RO}}}_{\pi_{\mathsf{rCA}}, \mathcal{A}, \mathcal{Z}}$ 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 $h > \tilde{\mathtt{D}}$ where $\tilde{\mathtt{D}} = \tilde{p} \cdot 2^\kappa$, 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 $h \le \tilde{\mathtt{D}}$, 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 $\mathsf{EXEC}^{\hat{\mathcal{F}}_{\mathsf{CA}}, \mathcal{F}_{\mathsf{RO}}}_{\pi_{\mathsf{rCA}}, \mathcal{A}, \mathcal{Z}}$ 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 $\hat{\mathcal{F}}_{\mathsf{CA}}$, $\mathcal{S}$ 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 $\hat{\mathcal{F}}_{\mathsf{CA}}$ 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.*

A.2 Ordinary Ideal Functionalities

A.2.1 Network Functionality

We fomally present the network functionality $\mathcal{F}_{\mathsf{NET}}$ in Figure 12.

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

The functionality is parametrized by Δ, and interacts with an adversary $\mathcal{S}$ and a set of PoW-miners $\{\mathcal{W}_1,\ldots,\mathcal{W}_n\}$, as well as a set of PoS-holders $\{\mathcal{S}_{n+1},\ldots,\mathcal{S}_{n+\tilde{n}}\}$.

- Upon receiving $(\textsc{Broadcast}, m)$ from a party P at round r where $P \in \{\mathcal{W}_1,\ldots,\mathcal{W}_n,\mathcal{S}_{n+1},\ldots,\mathcal{S}_{n+\tilde{n}}\}$, send $(\textsc{Broadcast}, m)$ to $\mathcal{S}$ and record (P,m,b,r) where $b=0$.
- Upon receiving $(\textsc{Delay}, m, P', t)$ from $\mathcal{S}$ where $P' \in \{\mathcal{W}_1,\ldots,\mathcal{W}_n,\mathcal{S}_{n+1},\ldots,\mathcal{S}_{n+\tilde{n}}\}$ (here, $\mathcal{S}$ can "spoof" the source of the message), then
  - If there is a record (P,m,b,r) such that $b=0$ and $t \le \Delta$, then send $(\textsc{Message}, P', m)$ to all other PoW-miners and PoS-holders at round $r+t$ and reset $b=1$.
  - Else, if $t > \Delta$, send $(\textsc{Message}, P', m)$ to all other PoW-miners and PoS-holders at round $r+\Delta$ and reset $b=1$.
  - Else, ignore the message.

Figure 12: Network functionality $\mathcal{F}_{\mathsf{NET}}$.

A.2.2 Multi-Session Certificate Authority Functionality $\hat{\mathcal{F}}_{\mathsf{CA}}$

We present the certificate authority functionality following the modeling of [10, 12].

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

The functionality $\hat{\mathcal{F}}_{\mathsf{CA}}$ interacts with a set of parties $\{P_1,\ldots,P_n\}$, and an adversary $\mathcal{S}$.

1. Upon receiving message $(\textsc{Register}, \mathsf{sid}, \mathsf{ssid}, v)$ from party $P \in \{P_1,\ldots,P_n\}$, verify that $\mathsf{ssid} = (P, \mathsf{ssid}')$ for some ssid'. If not, ignore the request. Otherwise, send $(\textsc{Register}, \mathsf{sid}, \mathsf{ssid}, v)$ to the adversary; upon receiving $(\textsc{Registered}, \mathsf{sid}, \mathsf{ssid},)$ from the adversary, then record the pair (ssid, v).
2. Upon receiving message $(\textsc{Retrieve}, \mathsf{sid}, \mathsf{ssid})$ from party $P' \in \{P_1,\ldots,P_n\}$, send $(\textsc{Retrieve}, \mathsf{sid}, \mathsf{ssid}, P')$ to the adversary, upon receiving $(\textsc{Retrieved}, \mathsf{sid}, \mathsf{ssid}, P')$ from the adversary. Then, if there is a recorded pair (ssid, v) output $(\textsc{Retrieved}, \mathsf{sid}, \mathsf{ssid}, v)$ to P'. Else output $(\textsc{Retrieved}, \mathsf{sid}, \mathsf{ssid}, \perp)$ to P'.

Figure 13: Multi-session certificate authority functionality $\hat{\mathcal{F}}_{\mathsf{CA}}$.

A.2.3 Multi-Session Signature Functionality $\hat{\mathcal{F}}_{\mathsf{SIG}}$

We present the multi-session version of the digital signature functionality in [10].

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

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

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

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

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

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

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

Figure 14: Multi-session signature functionality $\hat{\mathcal{F}}_{\mathsf{SIG}}$.

A.2.4 Random Oracle Functionality $\mathcal{F}_{\mathsf{RO}}$

The random oracle model (e.g., [3]) captures an idealization of a hash function. We here present the random oracle functionality $\mathcal{F}_{\mathsf{RO}}$ that has been defined in [21].

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

The functionality $\mathcal{F}_{\mathsf{RO}}$ is parameterized by a security paramater κ, and interacts with a set of parties $\{P_1, \ldots, P_n\}$, and an adversary $\mathcal{S}$.

1. $\mathcal{F}_{\mathsf{RO}}$ keeps a list L (which is initially empty) of pairs of bitstrings.
2. Upon receiving a value (m) (with $m \in \{0,1\}^*$) from some party $P \in \{P_1, \ldots, P_n\}$ or from $\mathcal{S}$, proceed as follows.
   - If there is a pair $(m, \tilde{h})$ for some $\tilde{h} \in \{0,1\}^\kappa$ in the list L,set $h := \tilde{h}$.
   - if there is no such pair, choose uniformly $h \in \{0,1\}^\kappa$ and store the pair (m, h) in L.

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

Figure 15: Random oracle functionality $\mathcal{F}_{\mathsf{RO}}$.

A.3 Deferred Proofs

A.3.1 Proof of Lemma 2.1

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

1. *Simulating registration phase:*

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

2. *Simulating regular query phase:* Upon receiving (B, X) from $\mathcal{A}$, then 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 $(\text{COMPUTE}, B, X)$ to $\mathcal{F}^*_{\mathsf{rRO}}$ and receive $(\text{COMPUTED}, h)$. Then record $((B,X),h)$ and send h to $\mathcal{A}$.

3. *Simulating work query phase:* Upon receiving (h, w) from $\mathcal{A}$ on the behaviour 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 $(\text{SEARCH}, \mathcal{W}_i, h)$ to $\mathcal{F}^*_{\mathsf{rRO}}$ and receive $(\text{SEARCHED}, \mathcal{W}_i, w')$. If $w' = \perp$, choose random $h' \in \{0,1\}^\kappa$ such that $h' > \mathtt{D}$ where $\mathtt{D} = p \cdot 2^\kappa$. Otherwise, if $w' \neq \perp$, choose random $h' \in \{0,1\}^\kappa$ such that $h' \leq \mathtt{D}$. Then record $((h,w),h')$, and send h' to $\mathcal{A}$.

4. *Simulating verification query phase:* 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 $(\text{RO-VERIFY}, B)$ to $\mathcal{F}^*_{\mathsf{rRO}}$ and receive $(\text{RO-VERIFIED}, h')$; if $h' = \perp$, choose random $h'' \in \{0,1\}^\kappa$ such that $h'' > \mathtt{D}$, otherwise, if $h' \neq \perp$, choose random $h'' \in \{0,1\}^\kappa$ such that $h'' \leq \mathtt{D}$; record (B, h'') and send h'' to $\mathcal{A}$.

We demonstrate that $\mathsf{EXEC}^{\mathcal{F}_{\mathsf{RO}}}_{\pi_{\mathsf{rRO}},\mathcal{A},\mathcal{Z}}$ 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_{rRO} 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 $\mathsf{EXEC}^{\mathcal{F}_{\mathsf{RO}}}_{\pi_{\mathsf{rRO}},\mathcal{A},\mathcal{Z}}$ and $\mathsf{EXEC}^{\mathcal{F}^*_{\mathsf{rRO}}}_{\phi^*_{\mathsf{rRO}},\mathcal{S},\mathcal{Z}}$ are perfectly close.

$\square$

A.3.2 Proof of Lemma 2.2

Proof. We show that the two executions are perfectly indistinguishable by the following simulation. Consider the adversary $\mathcal{A}$ for π_{rCERT}, we now construct an adversary $\mathcal{S}$ on input 1^κ and a PoS 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 ($\textsc{CA-Register}, \mathcal{S}_j, B, \mathsf{vk}$) 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 ($\textsc{CA-Registered}, \mathcal{S}_j, \mathsf{f}$) 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 $\mathcal{S}$ simulates the ideal functionalities $\hat{\mathcal{F}}_{\mathsf{SIG}}$ and $\mathcal{F}^*_{\mathsf{rCA}}$ for the hybrid protocol π_{rCERT}. It is clearly that the two ensembles $\mathsf{EXEC}^{\mathcal{F}^*_{\mathsf{rCERT}}}_{\phi^*_{\mathsf{rCERT}}, \mathcal{S}, \mathcal{Z}}$ and $\mathsf{EXEC}^{\mathcal{F}^*_{\mathsf{rCA}}, \hat{\mathcal{F}}_{\mathsf{SIG}}}_{\pi_{\mathsf{rCERT}}, \mathcal{A}, \mathcal{Z}}$ are perfectly indistinguishable since (1) the string vk is randomly chosen from a set $\{0,1\}^{\mathrm{poly}(\kappa)}$, and (2) other than that, the adversary $\mathcal{S}$ only needs to query the functionality $\mathcal{F}^*_{\mathsf{rCERT}}$ and forwards messages (with different headers) to $\mathcal{A}$. This completes the proof.

□