A Scalable Proof-of-Stake Blockchain in the Open Setting [*]

(or, How to Mimic Nakamoto's Design via Proof-of-Stake)

Lei Fan
Shanghai Jiao Tong University
fanlei@sjtu.edu.cn

Hong-Sheng Zhou
Virginia Commonwealth University
hszhou@vcu.edu

April 25, 2018

Abstract

Bitcoin and blockchain technologies have proven to be a phenomenal success. The underlying techniques hold huge promise to change the future of financial transactions, and eventually the way people and companies compute, collaborate, and interact. At the same time, the current Bitcoin-like proof-of-work based blockchain systems are facing many challenges. For example, a huge amount of energy/electricity is needed for maintaining the Bitcoin blockchain.

We propose a new approach to constructing energy-efficient blockchain protocols. More concretely, we develop proof-of-stake based, scalable blockchain protocols in the open network setting. Our contributions are as follows:

- We for the first time identify a new security property called *chain soundness* for proof-of-stake based protocols, which captures the intuition of ensuring new players to join the protocol execution securely.
- We for the first time formally investigate greedy strategies for proof-of-stake based protocols; via a greedy strategy, the protocol players may extend the best blockchain faster by attempting to extend multiple positions, instead of only the latest block, in the blockchain. We demonstrate a very useful upper bound of extending blockchain by greedy players, which enables us to give the first natural mimic of Bitcoin blockchain via proof-of-stake mechanism (without using any form of Byzantine fault tolerance).
- Our design is very simple, using only standard hash functions and unique digital signatures, which makes our design very appealing in practice. Our blockchain achieves important security properties including common prefix, chain quality, chain growth, and chain soundness, and is adaptively secure without assuming secure erasure.

[*]All results in this paper have been submitted to Eurocrypt 2018. In this version, the presentation has been improved, according to the feedback from the Eurocrypt reviewers, and from multiple researchers. In addition, in the current version, the related work part has been updated, and some discussions about rational attacks including nothing at stake attacks, selfish mining attacks, are added.

Contents

1 Introduction

Bitcoin and proof-of-work mechanism. Cryptocurrencies like Bitcoin [41] have proven to be a phenomenal success. The system was designed and implemented by an unknown researcher, under the name Satoshi Nakamoto nine years ago. The underlying techniques hold huge promise to change the future of financial transactions, and eventually the way people and companies compute, collaborate, and interact. At the heart of these cryptocurrency systems are distributed *blockchain* protocols, jointly executed by a large-scale peer-to-peer network of nodes called *miners* via the so-called *proof-of-work* mechanism [23, 3]. These blockchain protocols implement a highly trustworthy, append-only, and always-available public ledger, which can then be used to implement a global payment system (as in Bitcoin) or a global computer (as in Ethereum [10]).

Nakamoto's design has unique features: (1) the Bitcoin blockchain protocol can be executed in an *open* network environment in which all miners are allowed to join/leave the protocol execution at any moment they want; essentially, all miners are encouraged/incentivized to invest certain amount of computing power to join the effort of maintaining the blockchain; (2) the protocol has very low communication complexity and *can scale* to a large network of nodes.

From proof-of-work to proof-of-stake. The scalability[1] of Bitcoin blockchain protocol is at a price: the system has "wasted" a huge amount of computing resources over the past several years. It is definitely desirable to utilize alternative resources such as *coins (also called stakes)* to secure a blockchain. If successful, the new system will be "green" in the sense that it does not require a huge amount of *non-recyclable* computing power to back up its security. Attempts have been made: proof-of-stake (PoS) mechanisms have been widely discussed in the cryptocurrency community (e.g., [2, 37, 52, 5]). In a nutshell, in a proof-of-stake based blockchain protocol, players are expected to prove ownership of a certain number of coins/stakes. Only the players that can provide such a proof are allowed to participate in the process of maintaining the blockchain.

Blockchain protocols with provable security. In the past years, the security of Bitcoin-like protocols has been investigated. For example, Garay et al. [27] took the *provable security* approach and investigated Nakamoto's blockchain in a cryptographic framework (please also see [45]); they showed that, assuming the majority of mining power is controlled by the honest players, Nakamoto's blockchain protocol can achieve several important security properties such as *common prefix, chain quality, and chain growth*, as they defined in their cryptographic framework. However, how to ensure the security of new players has not been explicitly investigated in [27, 45], yet.

Alternative provably secure scalable blockchain protocols have been investigated recently. We highlight that, several proof-of-stake based protocols[17, 35, 20] have been proposed; these protocols can achieve common prefix, chain quality, and chain growth properties, and some can be executed in a large-scale closed network. However, it is not clear if these protocols can be scalable in the open setting. This leads to the following interesting question:

> *Is that possible to construct a proof-of-stake based, scalable blockchain protocol with provable security in the open setting?*

[1]The term *scalability* has been used to measure different capabilities of blockchain protocols. In this paper, we focus on the number of independent network nodes, and we say a blockchain protocol is *scalable* if it can be executed among a large-scale network of nodes (e.g., above ten thousands number of nodes as in Bitcoin).

1.1 Chain soundness

To answer the above question, we define a new security property, *chain soundness*, to ensure new players to join the protocol execution securely. Informally, considering a blockchain protocol execution, the chain soundness property states that a new honest player's best local chain is consistent with any existing honest players' local chains. See Subsection 2.2.3 for the formal definition.

We note that, although Nakamoto's protocol can be shown to achieve chain soundness, it has not been explicitly investigated in [27, 45]. In addition, previous proof-of-stake proposals (e.g., [17, 35, 20]) did not discuss the security when new players join the protocol execution, so it is not clear if those protocols can achieve chain soundness. In this paper, we propose new constructions, and we analyze their security in the open setting and we prove that they can achieve chain soundness.

1.2 Our construction

Warm-up: Nakamoto's design and proof-of-work (PoW) based core-chain We first briefly review Nakamoto's design ideas [41]. The blockchain in Bitcoin consists of a chain of ordered blocks $B_1, B_2, B_3, \ldots$, and PoW-players (i.e., miners) in each round (or time slot) attempt to extend the blockchain with a new block by solving proof-of-work puzzles [23, 3]. The *puzzle* for each miner is defined by (1) the "context", i.e., the latest block in the longest blockchain in the miner's view, and (2) the "payload", i.e., the set of valid transactions to be included in the new block; and a valid *puzzle solution* to the problem is defined by a hash inequality. More concretely, assume the longest blockchain for a miner consists of $B_1, B_2, \ldots, B_i$, and B_i is the latest block. The miner now attempts to find a valid puzzle solution *nonce* which can satisfy the following hash inequality:

$$\mathsf{H}(\mathsf{hash}(B_i), payload, \textit{nonce}) < \mathtt{T}$$

where $\mathsf{H}(\cdot)$ and $\mathsf{hash}(\cdot)$ are two hash functions, $payload$ denotes the set of valid transactions to be included in the new block, and $\mathtt{T}$ denotes the target of proof-of-work puzzle difficulty (which specifies how difficult to identify a puzzle solution by making a hash query attempt). In the case that a new valid solution, *nonce*, is identified, such a solution can be used for defining a new valid block B_{i+1} as follows:

$$B_{i+1} := \langle h_i, payload, \textit{nonce} \rangle$$

where $h_i := \mathsf{hash}(B_i)$. Then the new block B_{i+1} will be revealed by the miner, and broadcasted to the network and then accepted by the remaining miners in the system. (The above description is oversimplified.)

We may consider an even further simplified version of the above blockchain protocol, called *Bitcoin core-chain protocol*. In the core-chain protocol, the payload will be ignored, and now puzzle is based on hash inequality: $\mathsf{H}(\mathsf{hash}(B_i), \textit{nonce}) < \mathtt{T}$, and the new block B_{i+1} is defined as $B_{i+1} := \langle h_i, \textit{nonce} \rangle$. (We often call the blocks in a blockchain protocol, *blocks*, while the blocks in a core-chain protocol, *block-cores*.)

We note that, mimicking Nakamoto's footprint via proof-of-stake mechanism is non-trivial, and we have to address many technical challenges. To make our presentation more accessible, we start with the basic version of our proof-of-stake based core-chain protocol, Π^{core}, and then present the improved versions $\Pi^{\text{core}\star}$ and $\Pi^{\text{core}\bullet}$ which deal with greedy strategies and adaptive stake registrations, respectively. These eventually allow us to develop a full-fledged proof-of-stake blockchain protocol Π^{main}. Next, we illustrate our key ideas step by step.

1.2.1 Step 1, Π^{core}: Proof-of-stake (PoS) based core-chain, the basic version

We intend to mimic Nakamoto's design. Our proof-of-stake (PoS) based protocol will be maintained by PoS-players (i.e., stakeholders); We first consider the basic strategy that all players attempt to extend the longest chain with a new block. Similar to that in the PoW-based protocol, a winning PoS-player is chosen with some probability but using a different hash inequality. More concretely, assume the longest core-chain for a PoS-player consists of the following ordered block-cores, $B_1, B_2, \dots, B_i$; let $\texttt{round}$ denote the current time (or round number); consider a *unique* digital signature scheme [38][2] ($\mathsf{uKeyGen}, \mathsf{uKeyVer}, \mathsf{uSign}, \mathsf{uVerify}$), and assume the PoS-player holds the signing-verification key pair $(\textsc{sk}, \textsc{pk})$. If the PoS-player is chosen, then the following hash inequality holds:

$$\mathsf{H}(\mathsf{hash}(B_i), \texttt{round}, \textsc{pk}, \sigma) < \mathtt{T}$$

where $\sigma := \mathsf{uSign}_{\textsc{sk}}(h_i, \texttt{round})$, and $h_i := \mathsf{hash}(B_i)$. The new block-core B_{i+1} is defined as

$$B_{i+1} := \langle h_i, \texttt{round}, \textsc{pk}, \sigma \rangle$$

We remark that our design is very similar to Nakamoto's: the context here consists of the latest block-core in the longest core-chain, and the payload in the core-chain is empty; the puzzle solution consists of the current time, a PoS-player's verification key and his signature of the context.

When the adversary (1) follows the basic strategy, i.e., extending the single longest chain, and (2) has all stakes registered without being aware of the state of protocol execution, then our protocol can be viewed as a proof-of-stake analogy of Nakamoto's, and the security properties i.e., chain growth, chain quality, and common prefix (cf [27, 45]) can be demonstrated.[3]

Achieving chain soundness. Chain soundness property ensures that new players can join the system securely. Note that, a new player is not aware of the current state of the protocol execution (because the player did not participate in the protocol execution). In our design, each (new or existing) honest player takes the longest chain as the best chain. We can show that the adversary now cannot generate a longer chain privately to "confuse" the new players (otherwise the adversary can also confuse the existing honest players, which will violate the common prefix property).

Theorem 1.1 (informal)**.** *Consider core-chain protocol Π^{core} where all players follow the basic strategy of extending the longest chain; in addition, all players have their stakes registered without being aware of the state of the protocol execution. If more than 51% stakes are honest, then the protocol Π^{core} can achieve chain growth, chain quality, common prefix and chain soundness properties.*

See Section 3 for more details.

[2]There are multiple practical candidates of unique signature schemes in literature (e.g., [8]).

[3]We note that generating an unpredictable solution in proof-of-stake protocol design has been previously considered in the Algorand proposal [17]. The difference between Algorand proposal and our basic version Π^{core} is that, in our basic design the probability of finding a solution in a round is very low as in Bitcoin, while this probability in Algorand is high.

1.2.2 Step 2, $\Pi^{\mathrm{core}\star}$: Securing the core-chain against a greedy adversary

For the sake of simplifying our presentation, in the protocol Π^{core} above, we focus on the setting that all players follow the basic strategy to extend the core-chain. That is, each player will make attempts to extend the *single* best chain in his/her local view. We note that, this basic strategy has been widely adopted in the proof-of-work setting; there, extending a chain is expensive in the sense that it requires significant amount of computing power; it will be extremely costly to extend multiple chains simultaneously. However, in the proof-of-stake setting, it is very cheap to extend a chain. The proof-of-stake players may follow a **greedy** strategy to extend the core-chain: they make attempts to extend *a set of chains* and expect to obtain additional advantage for extending the best chain. Now we define greedy strategies and then present a modified protocol $\Pi^{\mathrm{core}\star}$ in the presence of players who are following greedy strategies. (For the sake of simplifying our presentation, we additionally assume that all players have their stakes registered without being aware of the state of the protocol execution as in Π^{core} in the previous step; This restriction will be removed in $\Pi^{\mathrm{core}\bullet}$ in the next step.)

Defining greedy strategies. Consider a blockchain protocol execution. Let P be a protocol player. Without loss of generality, there are multiple chains in P's local view, and these chains form a *tree*. That is, the root of the tree is the genesis block, and each path from the root is a chain. The tree "grows" round after round: the length of each existing chain may increase, and new chains will be created.

Let ℓ be the length of the longest chain at round r. Consider greedy parameter g where $0 \leq \mathrm{g} \leq \ell$. We say the player is g-*greedy* if, for all round r, the player makes attempts to extend a *set of chains* in which all chains have the length at least $(\ell - \mathrm{g})$. Note that, when $\mathrm{g} = 0$, the g-greedy strategy is essentially the basic strategy that we considered in protocol Π^{core}. When $\mathrm{g} = \ell$, we say the protocol player is *fully-greedy.*

Designing and analyzing protocols in the presence of greedy strategies. In order to defend against an adversary who follow a greedy strategy, we may consider to modify our protocol by having all honest players to follow the same greedy strategy. However, if an honest player follows the fully greedy strategy, then he/she will have to maintain a "big" tree and make attempts to extend the tree from all nodes (including leafs and intermediate nodes in the tree). In practice, we often make some tradeoff: an honest player may follow a weakened greedy strategy (even we are aware that the adversary may follow the fully greedy strategy) so that the protocol can be significantly simplified.

We demonstrate a very interesting upper bound: the fully greedy strategy will allow a PoS player to improve his/her chance of extending chains with a factor at most e where $e \approx 2.718$ (i.e., e is Euler's number, or the base of the natural logarithm). This upper bound allows us to develop secure core-chain protocols against greedy adversaries. When the adversary follows the fully greedy strategy and honest players follow the basic strategy, if more than 73% stakes are honest, then the protocol can achieve the security properties. We have the following theorem.

Theorem 1.2 (informal). *Consider core-chain protocol $\Pi^{\mathrm{core}\star}$ where honest players follow the 0-greedy strategy (i.e., basic strategy) while adversarial players follow the fully-greedy strategy; in addition, all players have their stakes registered without being aware of the state of the protocol execution. If more than 73% stakes are honest, then the protocol $\Pi^{\mathrm{core}\star}$ can achieve chain growth, chain quality, common prefix and chain soundness properties.*

If we are willing to complicate the protocol construction slightly, and let honest players to follow the 2-greedy strategy, then our protocol can be secure under a more relaxed assumption that honest players hold, not more than 73% stakes, but more than 57% stakes in the system. This assumption is very close to the "standard" honest majority 51%. We can have the following theorem.

Theorem 1.3 (informal)**.** *Consider core-chain protocol* $\Pi^{\text{core}\star}$ *where honest players follow the* 2-*greedy strategy while adversarial players follow the fully-greedy strategy; in addition, all players have their stakes registered without being aware of the state of the protocol execution. If more than* 57% *stakes are honest, then the protocol* $\Pi^{\text{core}\star}$ *can achieve chain growth, chain quality, common prefix and chain soundness properties.*

See Section 4 for more details.

1.2.3 Step 3, $\Pi^{\text{core}\bullet}$: Securing the core-chain further, against an adaptive adversary

The protocol $\Pi^{\text{core}\star}$ above is expected to be executed in a less realistic setting where all players must have their stakes registered without being aware of the state of the protocol execution. Recall that the hash inequality $\mathsf{H}(\mathit{context}, \mathit{solution}) < \mathtt{T}$ is used in the process of extending the chains. In reality, an adversary may have a stake registered *based on the state* of the protocol execution. More concretely, the adversary can play a "rejection re-sampling" strategy to generate keys, and then have his/her stake registered adaptively: the adversary first runs the key generation algorithm to obtain a key-pair $(\textsc{pk}, \textsc{sk})$, and then checks if the corresponding $(\textsc{pk}, \sigma)$ is a valid solution; if not, the adversary re-samples a new key-pair. This adaptive stake registration strategy enables the adversary (to be selected) to extend the chains with much higher probability. To address this concern, we introduce a new policy to our protocol: to extend the chains with new blocks, a player must have his/her stake registered much earlier.

Theorem 1.4 (informal)**.** *Consider core-chain protocol* $\Pi^{\text{core}\bullet}$ *where honest players follow the* 2-*greedy strategy while adversarial players follow the fully-greedy strategy. If more than* 57% *stakes are honest, then the protocol* $\Pi^{\text{core}\bullet}$ *can achieve chain growth, chain quality, common prefix and chain soundness properties.*

See Section 5 for more details.

1.2.4 Step 4, Π^{main}: From the core-chain to a blockchain

In this step, we will "upgrade" the core-chain protocol to a regular blockchain protocol so that payload (e.g., the transactions) can be included. Intuitively, the core-chain can be viewed as a (biased) randomness beacon; we can use the beacon to select a PoS-player to generate a new block so that the blockchain can be extended. More concretely, once a new block-core B_{i+1} is generated by a PoS-player (in the core-chain protocol), then the PoS-player is selected for generating the new block $\tilde{B}_{i+1}$, in the following format

$$\tilde{B}_{i+1} = \langle \mathsf{hash}(\tilde{B}_i), B_{i+1}, \tilde{X}_{i+1}, \tilde{\textsc{pk}}, \tilde{\sigma} \rangle$$

where $\tilde{\sigma} \leftarrow \mathsf{Sign}_{\tilde{\textsc{sk}}}(\tilde{h}_i, B_{i+1}, \tilde{X}_i)$, $\tilde{X}_{i+1}$ is payload and $\tilde{h}_i := \mathsf{hash}(\tilde{B}_i)$, and $B_{i+1} := \langle h_i, \mathtt{round}, \textsc{pk}, \sigma \rangle$. Here the PoS-player holds two pairs of keys, i.e., $(\textsc{sk}, \textsc{pk})$ of the unique signature scheme $(\mathsf{uKeyGen}, \mathsf{uSign}, \mathsf{uVerify})$, and $(\tilde{\textsc{sk}}, \tilde{\textsc{pk}})$ of a regular digital signature scheme $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$. Now we attach each block to the core-chain via the corresponding block-core; we can reduce the security of the blockchain protocol to the security of the core-chain protocol. Please also see Figure 1 for a pictorial illustration.

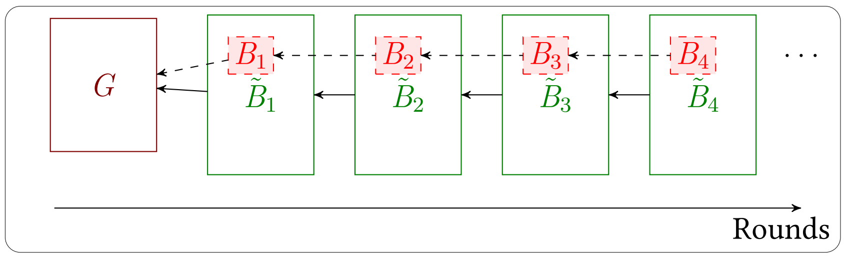

Figure 1: Blockchain structure
Blockchain $\tilde{\mathcal{C}}$ consists of initial setup information (i.e., genesis block) G, and then an ordered sequence of blocks $\tilde{B}_1, \tilde{B}_2, \tilde{B}_3, \ldots$. Here, each block $\tilde{B}_i$ consists of a block-core B_i and additional information. A core-chain $\mathcal{C}$ consists of the initial setup information G and the ordered sequence of block-cores $B_1, B_2, B_3, \ldots$.

Finally, we can show the following theorem.

Theorem 1.5 (informal)**.** *Consider blockchain protocol Π^{main} where honest players follow the 2-greedy strategy while adversarial players follow the fully-greedy strategy. If more than 57% stakes are honest, then the protocol Π^{main} can achieve chain growth, chain quality, common prefix and chain soundness properties.*

See Section 6 for more details.

1.2.5 Discussions and further extensions

Our design can be extended in multiple directions, including enabling adaptive difficulty adjustment and supporting light clients as in the original Bitcoin [41], and incentivizing the system via better strategies (e.g., [46]), managing transactions in blockchain more effectively (e.g., [48]), and more. Our design can also tolerate some well known rational attacks such as nothing at stake attacks and selfish mining attacks. Please see Section 7 for more details.

1.3 Related work

Cryptocurrency and proof-of-work. Anonymous digital currency was introduced by Chaum [16] in the early 1980s. The first decentralized currency system, Bitcoin [41], was launched about 30 years later, by incentivizing a set of players to solve moderately-hard cryptographic puzzles (also called proof-of-work puzzles [23, 3]). After that, many cryptocurrency systems were created based on proof-of-work puzzles (e.g., LiteCoin [1], Ethereum [10, 53]). Please refer to the online textbook and course [42] and the survey [9].

The security of Bitcoin system has been analyzed in the rational setting, e.g., [25, 24, 43, 32, 49, 50], and also in the cryptographic setting, e.g., [27, 45, 51, 33, 34, 28]. Several important cryptographic properties, *common prefix* [27, 45], *chain quality* [27], and *chain growth* [33], have been considered for proof-of-work blockchain protocols.

Proof-of-stake. Using virtual resources (i.e., stake) to construct cryptocurrency has been intensively considered. Since the inception of the idea in an online forum [7], several proof-of-stake proposals have been introduced and/or implemented (e.g., [2, 37, 52, 11, 5]). We remark that these proposals are *ad hoc* without formal security, and it is not clear how to formally prove the security of these proposals.

Very recently, several provably secure proof-of-stake based blockchain proposals (e.g., [17, 35, 20, 47]) have been developed. Unfortunately, all of them suffer from some major drawbacks.

The Sleepy protocol [47], is very efficient but it is designed for the closed setting which means new spawned players are not allowed to join the system during the execution. In the follow-up work, Snow White [20], new players are allowed to join the system; but these new players need to contact a group of honest majority players.

The most related work is Algorand [17, 29]. In Algorand, multiple players will be elected as lead candidates for signing the next block. Another group of players need to be elected to run an improved Byzantine Agreement (BA) protocol to determine which candidate block will be stored. It is not clear whether these heavyweight protocols can be executed efficiently in a large-scale *real-world* open network. We argue that, without relying on any form of BA, our PoS protocol is more suitable to be executed in the real world network environment (in which network delay is non-trivial).

Ouroboros Praos [21] is concurrent and independent work of ours. In their work, the protocol uses a verifiable random function (VRF) to elect a signer for next block. Greedy strategies are not addressed in the security analysis. In particular, in the full protocol of Praos, the blockchain consists of blocks that are generated in different epochs, and blocks in each epoch define the value η_i for generating blocks in next epoch; we note that, by producing different η_i's (e.g., the adversary may withhold some blocks in an epoch to produce different η_i), the adversary can play greedy strategies to extend the blockchain. In addition, we note that, in Praos, honest users are expected to erase certain specified secret data reliably; secure data erasure is a strong assumption which may be too complicated to implement in practice.

Combining proof-of-work and proof-of-stake. The idea of combining proof-of-work and proof-of-stake has been studied in [36, 19, 6, 22, 18]. Very recently, Duong et al [22] provided the first provably secure and scalable blockchain via proof-of-work/proof-of-stake in the open setting.

Additional alternative mechanisms. Alternative consensus techniques via different resources have been considered. For example, the physical storage resource is used in [44, 39]. A hybrid proposal of utilizing both computing and space resources, called proof-of-space-time was introduced in [40]. Recently, blockchain protocols via trusted hardware have also been proposed [31, 54].

1.4 Organization.

The remaining of the paper is organized as follows. In Section 2, we introduce an analysis framework for proof-of-stake protocols. In Section 3, we construct the basic version of our proof-of-stake based core-chain protocol, and then provide the security analysis. In Section 4, we introduce greedy strategies, and develop a modified proof-of-stake based core-chain protocol to defend against greedy adversaries, and then analyze its security. In Section 5, we improve the modified core-chain protocol further so that it can be executed in the real world environment where the players are allowed to register their key-pairs adaptively. In Section 6, we upgrade the core-chain protocol to a full-fledged blockchain protocol. Finally, extensions and related discussions are provided in Section 7. We note that, additional supporting materials can be found in Appendix A.

2 Model

In order to study the security of Bitcoin-like proof-of-work based protocols, Garay et al. [27] proposed a cryptographic framework and showed that (a simplified version of) Bitcoin protocol can achieve several important security properties. Then, Pass et al. [45] strengthened Garay et al.'s analysis by considering a more realistic communication network (i.e., partially synchronous network) in which messages from honest players can be delayed with an upper bound number of rounds. Below we define a framework for analyzing proof-of-stake based blockchain protocols. We note that we take many formulation ideas from the previous framework [27, 45].

2.1 Blockchain protocol executions

The execution of proof-of-stake blockchain protocol. Following Canetti's formulation of the "real world" executions [12, 13], we present an abstract model for proof-of-stake (PoS) blockchain protocol Π in the $\{\mathcal{F}_{\mathsf{Setup}}, \mathcal{F}_{\mathsf{NET}}\}$-hybrid model, where $\mathcal{F}_{\mathsf{NET}}$ denotes the partially synchronous network communication functionality (see Appendix A.4.1), and $\mathcal{F}_{\mathsf{Setup}}$ denotes the setup functionality (which will be explained soon), for the PoS-players. We consider the execution of blockchain protocol Π that is directed by an environment $\mathcal{Z}(1^\kappa)$ (where κ is a security parameter), which activates a set $\mathcal{P}$ of PoS-players. The environment $\mathcal{Z}$ can "manage" protocol players through an adversary $\mathcal{A}$ that can dynamically corrupt honest parties. More concretely, the $\{\mathcal{F}_{\mathsf{Setup}}, \mathcal{F}_{\mathsf{NET}}\}$-hybrid execution proceeds as follows. Each party in the execution is initialized with an initial state including all initial public information 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 round, each PoS-player $\mathrm{P} \in \mathcal{P}$, with a local state $state$ (note that $state$ originally includes the initial state), proceeds as follows.
    - When PoS-player P is activated by the environment $\mathcal{Z}$ by $(\textsc{Input-Stake}, \mathrm{P}, x)$ where x is the input from the environment, and potentially P receives subroutine output message $(\textsc{Message}, \mathrm{P}', m)$ for any $\mathrm{P}' \in \mathcal{P}$, from $\mathcal{F}_{\mathsf{NET}}$, the PoS-player P interacts with the functionality $\mathcal{F}_{\mathsf{Setup}}$ and receives some output y from $\mathcal{F}_{\mathsf{Setup}}$.
    - Next, the PoS-player P executes the protocol Π on input its local state $state$, the value y received from the functionality $\mathcal{F}_{\mathsf{Setup}}$, an input from the environment x, and the message m received from the functionality $\mathcal{F}_{\mathsf{NET}}$; and then P obtains an updated local state $state$ and an outgoing message m', i.e.,$\{state, m'\} \leftarrow \Pi(state, x, y, m)$. After that, P sends $(\textsc{Broadcast}, m')$ to $\mathcal{F}_{\mathsf{NET}}$ and then returns $(\textsc{Return-Stake}, \mathrm{P})$ to the environment $\mathcal{Z}$.

– At any round r of the execution, $\mathcal{Z}$ can send message $(\textsc{Corrupt}, \mathrm{P})$, where $\mathrm{P} \in \mathcal{P}$, to adversary $\mathcal{A}$. Then $\mathcal{A}$ will have access to the party's local state and control P.

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

Remark 2.1. *For simplicity, we focus on the idealized "flat" model where all PoS-players have the same number of stakes. Note that, in the reality, each different honest PoS-player may have a different amount of stake. In addition for simplicity, we focus on the idealized "static difficulty" model where the number of PoS-players that who have stakes, is fixed during the course of the protocol execution. That means, if some new PoS-players join the system, then the same number of PoS-players will leave the system. In Section 7, we will discuss how to extend our main results in the idealized flat, static difficulty model to the more realistic non-flat, adaptive difficulty setting.*

Remark 2.2 (Player joining and leaving)**.** *Protocol players are allowed to join the protocol execution* $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}^{\mathcal{F}_{\mathsf{Setup}},\mathcal{F}_{\mathsf{NET}}}$. *More explicitly, PoS-players, during their first interactions with the setup functionality* $\mathcal{F}_{\mathsf{Setup}}$, *can have themselves registered.*

However, it is very subtle to have PoS-players unregistered *when they decide to leave the protocol execution. In the current version of our modeling, we assume that when (honest) PoS-players leave the protocol execution, they will* erase *their own local internal information. That means, the protocol execution is in the non-erasure model (except for these already unregistered players). For this reason, we currently* disabled *the* $\textsc{Stake-Unregister}$ *command in the* $\mathcal{F}_{\mathsf{Setup}}$ *(e.g.,* $\mathcal{F}^{\bullet}_{\mathsf{rCERT}}$ *in Section 5) but keep the* $\textsc{Stake-Register}$ *command; we note that the* $\textsc{Stake-Unregister}$ *comand can be added back if certain sophisticated mechanisms are introduced.*

2.2 Security properties

2.2.1 Blockchain basics

A *blockchain* $\mathcal{C}$ consists of a sequence of ℓ concatenated blocks $B_0\|B_1\|B_2\|\cdots\|B_\ell$, where $\ell \geq 0$ and B_0 is the initial block (genesis block). We use $\mathrm{len}(\mathcal{C})$ to denote *blockchain length*, i.e., the number of blocks in blockchain $\mathcal{C}$; and here $\mathrm{len}(\mathcal{C}) = \ell$. We use sub blockchain (or subchain) for referring to segment of a chain; here for example, $\mathcal{C}[1, \ell]$ refers to an entire blockchain, whereas $\mathcal{C}[j, m]$, with $j \geq 1$ and $m \leq \ell$ would refer to a sub blockchain $B_j\|\cdots\|B_m$. We use $\mathcal{C}[i]$ to denote the i-th block B_i in blockchain $\mathcal{C}$. If blockchain $\mathcal{C}$ is a prefix of another blockchain $\mathcal{C}'$, we write $\mathcal{C} \preceq \mathcal{C}'$. If a chain $\mathcal{C}$ is truncated the last κ blocks, we write $\mathcal{C}[\neg\kappa]$.

2.2.2 Chain growth, common prefix, and chain quality

Previously, several fundamental security properties for proof-of-work blockchain protocols have been defined: *common prefix property* [27, 45], *chain quality property* [27], and *chain growth property* [33]. Intuitively, the chain growth property states that the chains of honest players should grow linearly to the number of rounds. The common prefix property indicates the consistency of any two honest chains except the last κ blocks. The chain quality property, aims at expressing the number of honest blocks' contributions that are contained in a sufficiently long and continuous part of an honest chain. Specifically, for parameters $\ell \in \mathbb{N}$ and $\mu \in (0, 1)$, the ratio of honest input contributions in a continuous part of an honest chain has a lower bounded μ. We follow the same spirit to define the security properties for proof-of-stake blockchain protocols. The definitions for these properties are formally given as follows.

Definition 2.3 (Chain growth). *Consider a blockchain protocol* Π *with a set* $\mathcal{P}$ *of players. The chain growth property with parameter* $g \in \mathbb{R}$*, states the following: for any honest player* P' *with local chain* $\mathcal{C}'$ *at round* r'*, and honest player* P'' *with local chain* $\mathcal{C}''$ *at round* r''*, where* $\mathrm{P}', \mathrm{P}'' \in \mathcal{P}$ *and* $r'' > r'$*, in the execution* $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$*, it holds that* $\mathrm{len}(\mathcal{C}'') - \mathrm{len}(\mathcal{C}') \geq g(r'' - r')$.

Definition 2.4 (Common prefix). *Consider a blockchain protocol* Π *with a set* $\mathcal{P}$ *of players. The common prefix property states the following: for any honest player* P' *with local chain* $\mathcal{C}'$ *at round* r'*, and honest player* P *with local chain* $\mathcal{C}$ *at round* r*, in the execution* $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$*, where* $\mathrm{P}', \mathrm{P} \in \mathcal{P}$ *and* $r \leq r'$*, it holds that* $\mathcal{C}[\neg\kappa] \preceq \mathcal{C}'$.

Definition 2.5 (Chain quality). *Consider a blockchain protocol* Π *with a set* $\mathcal{P}$ *of players. The chain quality property with parameters* μ, ℓ*, where* $\mu \in \mathbb{R}$ *and* $\ell \in \mathbb{N}$*, states the following: for any honest player* $\mathrm{P} \in \mathcal{P}$*, with local chain* $\mathcal{C}$ *in round* r*, in* $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$*, it holds, for large enough* ℓ *consecutive blocks of* $\mathcal{C}$*, the ratio of honest blocks is at least* μ.

2.2.3 New property: Chain soundness

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

Definition 2.6 (Chain soundness). *Consider a blockchain protocol* Π *with a set* $\mathcal{P}$ *of players. Consider a new player* $\mathrm{P} \in \mathcal{P}$ *with best local chain* $\mathcal{C}$ *in round* r*, in* $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$*. The* chain soundness *property states the following: for the new player* P *and any existing players* P' *with best local chain* $\mathcal{C}'$ *at round* r*, it holds that* $\mathcal{C}'[\neg\kappa] \preceq \mathcal{C}$ *and* $\mathcal{C}[\neg\kappa] \preceq \mathcal{C}'$.

We remark that, the chain soundness is different from the common prefix property above. In the latter, only existing players are considered. Several previous protocols (e.g., [35, 20]) can achieve common prefix but not chain soundness.

3 Proof-of-stake core-chain, the basic design and analysis

High-level protocol description. The informal description of our core-chain protocol Π^{core} has been illustrated in the Introduction: assume the longest core-chain for a PoS-player consists of the following ordered block-cores, $B_1, B_2, \ldots, B_i$; let $\mathtt{round}$ denote the current time (or round number); consider a *unique* digital signature scheme $(\mathsf{uKeyGen}, \mathsf{uKeyVer}, \mathsf{uSign}, \mathsf{uVerify})$, and assume the PoS-player holds the signing-verification key pair $(\textsc{sk}, \textsc{pk})$. If the PoS-player is chosen, then the following hash inequality holds: $\mathsf{H}(\mathsf{hash}(B_i), \mathtt{round}, \textsc{pk}, \sigma) < \mathtt{T}$, where $\sigma := \mathsf{uSign}_{\textsc{sk}}(h_i, \mathtt{round})$, and $h_i := \mathsf{hash}(B_i)$. The new block-core B_{i+1} is defined as $B_{i+1} := \langle h_i, \mathtt{round}, \textsc{pk}, \sigma \rangle$.

When the adversary (1) follows the basic strategy, i.e., extending the single longest chain, and (2) has all stakes registered independent of the state of protocol execution, then our protocol can be viewed as a proof-of-stake analogy of Nakamoto's, and the security properties i.e., chain growth, chain quality, and common prefix (cf [27, 45]) can be demonstrated.

Formal protocol description. Next we will provide a formal description for our protocol. We use a setup functionality $\mathcal{F}_{\mathsf{rCERT}}$ to capture the hash inequality and the block-core signing/verification. This setup functionality can be implemented by using hash function $\mathsf{H}(\cdot)$ and a strengthened unique signature scheme $(\mathsf{uKeyGen}, \mathsf{uKeyVer}, \mathsf{uSign}, \mathsf{uVerify})$.

The power of the adversary. Finally, we note that in this section, the adversary will follow the basic strategy in the sense that all the accounts are registered previously before the protocol execution.

3.1 Setup functionality $\mathcal{F}_{\mathsf{rCERT}}$

Resource certification functionality $\mathcal{F}_{\mathsf{rCERT}}$. The functionality consists of several phases, "Stake Resource Registration", "Stake Election", and "Block Verification". In this version of resource certificate functionality, the "Stake Resource Registration" phase is disabled, and we have all PoS-player P registered initially. (Jumping ahead, in Section 5, a strengthened version of resource certification functionality $\mathcal{F}^{\bullet}_{\mathsf{rCERT}}$ in Figure 10, will be introduced; there, at any time step, a PoS-player P can send a register command to functionality $\mathcal{F}^{\bullet}_{\mathsf{rCERT}}$ for registration.) For simplicity, we assume a registered PoS-player P is granted *one unit of the stake*, and he can then request the functionality for leader election once in each execution round. We will extend this flat model to non-flat model in Section 7.

Firstly, we will introduce "Stake Resource Registration" phase. In this phase, a player P send $(\textsc{Elect}, \mathrm{P}, \langle h^{\text{prev}}, \mathtt{round}\rangle)$ to the functionality; the functionality requests adversary to produce the signature by command $(\textsc{Core-Sign}, \mathrm{P}, \langle h^{\text{prev}}, \mathtt{round}\rangle)$, and then waits until the adversary responds with a signature σ. The functionality then with probability p selects this party as the leader and notifies the player whether he is selected or not.In this phase, We remark that, the elected party P can obtain only a **single** signature for a h^{prev} in one round; in previous certificate or digital signature functionalities (see [14]), multiple signatures are allowed to be generated for the same value. Then the functionality generate a unique id h for the player P as the identity of a new block. With the id h, the functionality can distinguish every valid block. The functionality store a record of the form $\langle h^{\text{prev}}, \mathtt{round}, \mathrm{P}, \sigma, h, 1\rangle$ for the new block. Here, h^{prev} is the id of the previous block. So that the functionality can and trace the order of of blocks. The functionality return $(\textsc{Elected}, \mathrm{P}, h, \sigma, \mathsf{b})$ to P where b is used to indicate if P is elected in this round.

Secondly, the verification process of $\mathcal{F}_{\mathsf{rCERT}}$ proceeds as follows. Upon receiving a verification request, the functionality will check if the signature is valid by sending a request to adversary. Then the functionality will also check there is a valid record of this block been recorded. This would ensure the completeness, unforgeability, and guarantees consistency properties of the block.

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

The functionality interacts with a set $\mathcal{P}$ of parties, as well as an adversary.
The functionality is parameterized by a difficulty parameter p, a security parameter κ.
Initially, a set $\mathcal{P}_0$ of distinct players are registered, where $\mathcal{P}_0 \subseteq \mathcal{P}$; That is, for all $\mathrm{P} \in \mathcal{P}_0$ the records $(\mathrm{P}, 1)$ are stored.

Stake Resource Registration:
(This phase is disabled, and all stake registration must be completed during initialization. In the strengthened version of the functionality in Figure 10, this phase will be enabled for supporting regular stake registration.)

Stake Election: For each round, set $\phi_{\mathrm{P},h^{\mathrm{prev}}} := 0$ for every registered party $\mathrm{P} \in \mathcal{P}_0$.
Upon receiving $(\textsc{Elect}, \mathrm{P}, \langle h^{\mathrm{prev}}, \texttt{round}\rangle)$ from a PoS-player P, proceed as follows.
Set $\mathsf{b} := 0$. (the party P is not elected by default)

1. If $(\mathrm{P}, 1)$ is recorded, and $\phi_{\mathrm{P},h^{\mathrm{prev}}} = 0$, (the party P registered and granted one unit of stake)

   Send $(\textsc{Core-Sign}, \mathrm{P}, \langle h^{\mathrm{prev}}, \texttt{round}\rangle)$ to the adversary.

   Upon receiving $(\textsc{Signature}, \mathrm{P}, \langle h^{\mathrm{prev}}, \texttt{round}\rangle, \sigma)$, do:

   If $\langle h^{\mathrm{prev}}, \texttt{round}, \mathrm{P}, \sigma, \cdot, \cdot\rangle$ has been recorded, then ignore the input. (party P can only obtain one signature for h^{prev} in a round.) Otherwise, send a request to the adversary for a unique value h; if $\langle \cdot, \cdot, \cdot, \cdot, h, \cdot\rangle$ has been recorded, then ignore the input. Otherwise,

   – with probability p, set $\mathsf{b} := 1$ (the party P is elected), and store a record of the form $\langle h^{\mathrm{prev}}, \texttt{round}, \mathrm{P}, \sigma, h, 1\rangle$ in memory.

Set $\phi_{\mathrm{P},h^{\mathrm{prev}}} := 1$
Send $(\textsc{Elected}, \mathrm{P}, h, \sigma, \mathsf{b})$ to P

Block Verification:
Upon receiving $(\textsc{Core-Verify}, \mathrm{P}, \langle h^{\mathrm{prev}}, \texttt{round}\rangle, \sigma, h)$ from a party $\mathrm{P}' \in \mathcal{P}$,
Set $f := 0$

1. Send $(\textsc{Core-Verify}, \mathrm{P}, \langle h^{\mathrm{prev}}, \texttt{round}\rangle, \sigma)$ to the adversary.

   Upon receiving $(\textsc{Core-Verified}, \mathrm{P}, \langle h^{\mathrm{prev}}, \texttt{round}\rangle, \phi)$ from the adversary, do:

   – If $\langle h^{\mathrm{prev}}, \texttt{round}, \mathrm{P}, \sigma, h, 1\rangle$ is recorded, then set $f := 1$.
   – Else, if P is not corrupted, and no entry $\langle h^{\mathrm{prev}}, \texttt{round}, \mathrm{P}, \sigma', h, 1\rangle$ for any σ' is recorded, then set $f := 0$ and record the entry $\langle h^{\mathrm{prev}}, \texttt{round}, \mathrm{P}, \sigma, h, 0\rangle$.
   – Else, if there is an entry $\langle h^{\mathrm{prev}}, \texttt{round}, \mathrm{P}, \sigma, h, f'\rangle$, then set $f := f'$.
   – Else, set $f := \phi$, and record the entry $\langle h^{\mathrm{prev}}, \texttt{round}, \mathrm{P}, \sigma, h, f\rangle$.

Output $(\textsc{Core-Verified}, \mathrm{P}, \langle h^{\mathrm{prev}}, \texttt{round}\rangle, \sigma, h, f)$ to the party P'.

Figure 2: Resource certification functionality $\mathcal{F}_{\mathsf{rCERT}}$.

How to implement functionality $\mathcal{F}_{\text{rCERT}}$? We note that, in [14], $\mathcal{F}_{\text{CERT}}$ can be implemented in the $\{\mathcal{F}_{\text{CA}}, \mathcal{F}_{\text{SIG}}\}$-hybrid model. We can follow the similar approach to implement our functionality $\mathcal{F}_{\text{rCERT}}$ in the $\{\mathcal{F}_{\text{rCA}}, \hat{\mathcal{F}}_{\text{uSIG}}, \mathcal{F}_{\text{RO}}\}$-hybrid model. Please refer to Appendix A.1. Note that $\hat{\mathcal{F}}_{\text{uSIG}}$ is a variant of the multi-session signature functionality [15] in the sense that, for each signer only one signature is allowed to be generated for a value, and this variant can be realized by a multi-session signature protocol based on unique signature scheme. In addition, the multi-session certificate authority functionality $\hat{\mathcal{F}}_{\text{CA}}$ could be implemented in multiple ways. For example, we can instantiate functionality $\hat{\mathcal{F}}_{\text{CA}}$ via a (variant) of a "mature" blockchain such as Bitcoin. Please refer to Appendix A.2 for more details about certificate authority functionality $\hat{\mathcal{F}}_{\text{CA}}$, Appendix A.3 for multi-session signature functionality $\hat{\mathcal{F}}_{\text{uSIG}}$, and Appendix A.4.2 for random oracle functionality $\mathcal{F}_{\text{RO}}$, respectively.

3.2 Our core-chain protocol

We now describe the core-chain protocol Π^{core}. Each PoS-player P, once activated by the environment on $(\text{INPUT-STAKE}, \mathrm{P})$ at round $\mathtt{round}$, and received a core-chain set $\mathbb{C}$ from $\mathcal{F}_{\text{NET}}$, the party P finds the best valid core-chain $\mathcal{C}_{\text{best}}$ by running the subroutine BestCore (in Figure 4), and then updates its local core-chain $\mathcal{C} := \mathcal{C}_{\text{best}}$.

Let ℓ be the length of core-chain $\mathcal{C}$. In our design, only the elected PoS-players are allowed to generate new block-cores (to extend the core-chain). Now, each registered PoS-player P will work on the right "context" which consists of the *latest block-core in the longest core-chain and the current time*; formally $context := \langle h^{\text{prev}}, \mathtt{round}\rangle$ where $\mathcal{C}[\ell]$ is the latest block-core in the longest core-chain $\mathcal{C}$, and h^{prev} is the identity returned by the functionality $\mathcal{F}_{\text{rCERT}}$ for $\mathcal{C}[\ell]$, and $\mathtt{round}$ denotes the current time. The PoS-player P may query $\mathcal{F}_{\text{rCERT}}$ by command $(\text{ELECT}, \mathrm{P}, context, \mathcal{C})$ to see if he is selected to extend $\mathcal{C}$. If the PoS-player P is selected (with certain probability p), he would receive a message $(\text{ELECTED}, \mathrm{P}, h, \sigma, \mathsf{b})$ from $\mathcal{F}_{\text{rCERT}}$ such that $\mathsf{b} = 1$. Once receiving the signature σ from the functionality, the PoS-player P defines a new block-core $B := \langle\langle h^{\text{prev}}, h, \mathtt{round}\rangle, \mathrm{P}, \sigma\rangle$, updates his local core-chain $\mathcal{C}$ and then broadcasts the local core-chain to the network. Please refer to Figure 3 for more details of our core-chain protocol.

Note that here PoS-players have access to the functionality $\mathcal{F}_{\text{rCERT}}$. The players need to register to the functionality $\mathcal{F}_{\text{rCERT}}$ before querying the functionality.

The best core-chain strategy. Our proof-of-stake core-chain protocol Π^{core} uses the subroutine BestCore to single out the best valid core-chain from a set of core-chains. Now we describe the rules of selecting the best core-chain. Roughly speaking, a core-chain is the best one if it is the *current longest valid* core-chain. The BestCore subroutine takes as input, a core-chain set $\mathbb{C}'$ and the current time information $\mathtt{round}'$. Intuitively, the subroutine validates all $\mathcal{C} \in \mathbb{C}'$, then finds the valid longest core-chain.

In more detail, BestCore proceeds as follows. On input the current set of core-chains $\mathbb{C}'$ and the current time information $\mathtt{round}'$, and for each core-chain $\mathcal{C}$, the subroutine then evaluates every block-core of the core-chain $\mathcal{C}$ sequentially. Let ℓ be the length of $\mathcal{C}$. Starting from the head of $\mathcal{C}$, for every block-core $\mathcal{C}[i]$, for all $i \in [\ell]$, in the core-chain $\mathcal{C}$, the BestCore subroutine (1) ensures that $\mathcal{C}[i]$ is linked to the previous block-core $\mathcal{C}[i-1]$ correctly, and (2) tests if the

```
                              PROTOCOL $\Pi^{\text{core}}$

Initially, a set $\mathcal{P}_0$ of players are registered to the functionality $\mathcal{F}_{\text{rCERT}}$, where $\mathcal{P}_0 \subseteq \mathcal{P}$. Initially, for each
$\mathrm{P} \in \mathcal{P}$, set $\mathcal{C} := \emptyset$, and $state := \emptyset$.

Upon receiving message $(\textsc{Input-Stake}, \mathrm{P})$ from the environment $\mathcal{Z}$ at round round, the PoS-player $\mathrm{P} \in$
$\mathcal{P}$, with local state $state$, proceeds as follows.

    1. Select the best local PoS core-chain:
       Let $\mathbb{C}$ be the set of core-chains collected from $\mathcal{F}_{\text{NET}}$.
       Compute $\mathcal{C}_{\text{best}} := \mathsf{BestCore}(\mathbb{C} \cup \{\mathcal{C}\}, \texttt{round})$, and set $\mathcal{C} := \mathcal{C}_{\text{best}}$, and $\ell := \text{len}(\mathcal{C}_{\text{best}})$

    2. Attempt to extend PoS core-chain:
       Parse $\mathcal{C}[\ell]$ as $\langle\langle h_\ell^{\text{prev}}, \texttt{round}_\ell, \mathrm{P}_\ell, \sigma_\ell\rangle, h_\ell\rangle$.

       – Stake election:
             Send $(\textsc{Elect}, \mathrm{P}, \langle h_\ell, \texttt{round}\rangle)$ to functionality $\mathcal{F}_{\text{rCERT}}$,
             and receive $(\textsc{Elected}, \mathrm{P}, h_{\ell+1}, \sigma, \mathsf{b})$ from $\mathcal{F}_{\text{rCERT}}$.
       – If $\mathsf{b} = 1$, generate a new block-core:
             Set the new block-core $B := \langle\langle h_\ell, \texttt{round}, \mathrm{P}, \sigma\rangle, h_{\ell+1}\rangle$,
             and set $\mathcal{C} := \mathcal{C} \| B$, and $state := state \cup \{\mathcal{C}\}$,
             and then send $(\textsc{Broadcast}, \mathcal{C})$ to $\mathcal{F}_{\text{NET}}$.

       Return $(\textsc{Return}\text{-Stake}, \mathrm{P})$ to the environment $\mathcal{Z}$.
```

Figure 3: Our proof-of-stake core-chain protocol Π^{core} in the $\{\mathcal{F}_{\text{rCERT}}, \mathcal{F}_{\text{NET}}\}$-hybrid model. (See Figure 4 for the subroutine BestCore.)

signature generated by that PoS-player can be verified (by interacting with $\mathcal{F}_{\text{rCERT}}$). After the validation, the best valid core-chain is the longest one. Please refer to Figure 4 for more details.

3.3 Security analysis for core-chain

Our core-chain protocol Π^{core} is in the "flat, static difficulty" model in which each PoS-playerholds a unit of stake and the total number of stakeholders is fixed. Let n be the total number of stakeholders in the protocol. Let p denote the probability that a stakeholder is qualified to extend the core-chain in a round. Let ρ denote the ratio of malicious stake. Let $\alpha_0 = (1-\rho)np$ be the expected number of honest stakeholders that are qualified in a round to extend the longest core-chain. Let $\beta_0 = \rho np$ be the expected number of malicious stakeholders that are qualified in a round to extend any chosen core-chain. Let α and β be the effective counterparts, respectively in the network delay setting. Here we assume $np \ll 1$. This means the expected number of stakeholders that are qualified to extend a core-chain in a round is much less than 1. Additionally, we assume that $\alpha_0 = \lambda\beta_0$ where $\lambda \in (1, \infty)$.

We are now ready to state our theorem for our core-chain protocol Π^{core} in the presence of an adversary who extends blockchain via the basic strategy (i.e., extending a single chain once).

Theorem 3.1 (Theorem 1.1, restated). *Consider core-chain protocol Π^{core} where all players follow the simple strategy of extending the longest chain; in addition, all players have their stake registered independent of the state in the protocol execution. Let α and β be the effective expected number of blocks generated by honest and malicious players in a round respectively. If $\alpha = \lambda\beta$, $\lambda > 1$, then the protocol Π^{core} can achieve chain growth, chain quality, common prefix and chain soundness*

```
SUBROUTINE BestCore

The subroutine BestCore is allowed to access to the functionality $\mathcal{F}_{\mathsf{rCERT}}$, and with input ($\mathbb{C}', \mathtt{round}'$).
For every chain $\mathcal{C} \in \mathbb{C}'$, and proceed as follows.

  1. Set $\ell := \mathrm{len}(\mathcal{C})$.
  2. For $i$ from $\ell$ down to 1, verify block-core $\mathcal{C}[i]$, as follows.
     • Parse $\mathcal{C}[i]$ into $\langle\langle h_i^{\mathrm{prev}}, \mathtt{round}_i, \mathrm{P}_i, \sigma_i\rangle, h_i\rangle$.
       Parse $\mathcal{C}[i-1]$ into $\langle\langle h_{i-1}^{\mathrm{prev}}, \mathtt{round}_{i-1}, \mathrm{P}_{i-1}, \sigma_{i-1}\rangle, h_{i-1}\rangle$.
     • If $\mathtt{round}_i < \mathtt{round}'$ and $\mathtt{round}_{i-1} < \mathtt{round}_i$ , then execute:
       – If $h_i^{\mathrm{prev}} \neq h_{i-1}$, then remove this core-chain $\mathcal{C}$ from $\mathbb{C}'$.
       – Else send (CORE-VERIFY, $\mathrm{P}_i, \langle h_i^{\mathrm{prev}}, \mathtt{round}_i\rangle, \sigma_i, h_i$) to $\mathcal{F}_{\mathsf{rCERT}}$.
         Upon receiving message (CORE-VERIFIED, $\mathrm{P}_i, \langle h_i^{\mathrm{prev}}, \mathtt{round}_i\rangle, \sigma_i, h_i, f_i$) from $\mathcal{F}_{\mathsf{rCERT}}$, if $f_i = 0$ remove this core-chain $\mathcal{C}$ from $\mathbb{C}'$.
       Otherwise, remove the core-chain $\mathcal{C}$ from $\mathbb{C}'$.

Set $\mathcal{C}_{\mathrm{best}}$ be the longest core-chain in $\mathbb{C}'$. Then return $\mathcal{C}_{\mathrm{best}}$ as the output.
```

Figure 4: The core-chain set validation subroutine BestCore.

properties.

3.3.1 Proof ideas

Here we introduce the high-level proof ideas. In our design, malicious players cannot prevent the honest players from being selected to generate new block-cores. This will guarantee the chain growth property. Furthermore, the total number of block-cores from malicious players are bounded by the proportion of stakes they control. Since we assume that the honest players control more stakes than the malicious players, for the same core-chain, the malicious players cannot contribute more block-cores than the honest players. This will give us the chain quality property. Finally, we assume the probability that the stakeholders find a new block-core B in a round is very small. This means, in most of the rounds, no new block-core is broadcast, and all of the honest players will extend on the same core-chain. Note that, even all of malicious players try to extend another core-chain, the growth rate of the malicious core-chain is still lower than that of the public core-chain. This will allow us to prove the common prefix property. Our core-chain protocol will be executed in a setting that the adversary can delay the messages from honest players up to certain say Δ, number of rounds. The honest players may be misled to work on a wrong core-chain during the delayed rounds. As a result, the effort from the honest players is wasted during these delayed rounds. Our analysis will also take care of the network delay.

Here we assume all players have their stakes registered without being aware of the state of the protocol execution. In Section 5, we will discuss how to eliminate this assumption, and show how to improve the core-chain protocol so that it can be executed in a more realistic setting.

Chain growth. In order to calculate the chain growth rate, we consider the worst case for the honest players. The best strategy for the malicious players is to delay all of the messages from the honest players to discount the stakes of honest players. We use α to denote the discounted number of block-cores that honest players can generate. We have $\alpha = \frac{\alpha_0}{1+\Delta\alpha_0}$. (The calculation steps can be found in Section 3.3.3.) 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. In the real execution, the probability that an honest player is chosen will not be lower than that in the hybrid execution. The message from malicious players will not decrease the chain growth that contributed by honest players. Therefore, the chain growth rate is not worse than that in the hybrid execution.

Lemma 3.2 (Chain growth)**.** *Consider core-chain protocol Π^{core}, an honest PoS-player P' with best local core-chain $\mathcal{C}'$ in round r', and an honest PoS-player P'' with best local core-chain $\mathcal{C}''$ in round r'', where $r'' > r'$. Then we have $\Pr\left[\text{len}(\mathcal{C}'') - \text{len}(\mathcal{C}') \geq g \cdot t\right] \geq 1 - e^{-\Omega(t)}$, where $t = r'' - r'$, $g = (1-\delta)\alpha$, and $\delta > 0$.*

Chain quality. In order to reduce the core-chain quality, the best strategy for malicious parties is to generate as many block-cores as they can. When the honest players generate and broadcast a new block-core, they will try to send out another one to compete with the honest one. We focus on the worst case that the malicious players win all of the competition. During any t consecutive rounds, the core-chain growth rate is αt on average. The malicious players will contribute βt block-cores. The core-chain quality will remain at least $1 - \frac{\beta}{\alpha}$.

Lemma 3.3 (Chain quality)**.** *Consider $\alpha = \lambda\beta$, $\lambda > 1$, and $\delta > 0$. Consider core-chain protocol Π^{core}, and an honest PoS-player with core-chain $\mathcal{C}$. Consider that ℓ consecutive block-cores of $\mathcal{C}$, where ℓ_{good} block-cores are generated by honest PoS-players. Then we have $\Pr\left[\frac{\ell_{good}}{\ell} \geq \mu\right] \geq 1 - e^{-\Omega(\ell)}$, where $\mu = 1 - (1+\delta)\frac{1}{\lambda}$.*

Common prefix. We assume $\alpha + \beta \ll 1$. This guarantees that the honest players will work on the same best core-chain in most rounds. We also assume the majority of PoS-players are honest. Together, we have that the public best chain is longer than any other core-chains after a sufficient long period. All of the honest players will converge on the best public chain with high probability except the last several block-cores.

Lemma 3.4 (Common prefix)**.** *Consider $\alpha = \lambda\beta$, $\lambda > 1$, and $\delta > 0$. Consider core-chain protocol Π^{core}. and two honest PoS-players, P in round r and P' in round r', with the local best core-chains $\mathcal{C}$, $\mathcal{C}'$, respectively, where $r' \geq r$. Then we have $\Pr\left[\mathcal{C}[1,\ell] \preceq \mathcal{C}'\right] \geq 1 - e^{-\Omega(\kappa)}$, where $\ell = \text{len}(\mathcal{C}) - \Theta(\kappa)$.*

Chain soundness. For a new player, he will take the longest core-chain he received. As we discussed above, the honest players can generate the longest chain except the latest several block-cores. This means the new player can choose the best core-chain as the existing players in the protocol.

Lemma 3.5 (Chain soundness)**.** *Consider for every round, $\alpha = \lambda\beta$, $\lambda > 1$, and $\delta > 0$. Consider core-chain protocol Π^{core}. Consider two honest PoS-players, P' and P'' in round r, with the local best* *core-chains $\mathcal{C}'$ and $\mathcal{C}''$, respectively, where P' is a new player and P'' is an existing player in round r. Then we have $\mathcal{C}'[\neg\kappa] \preceq \mathcal{C}''$ and $\mathcal{C}''[\neg\kappa] \preceq \mathcal{C}'$.*

3.3.2 Basic terms

Before giving the details of the security analysis, we introduce some terms.

Definition 3.6 (Honest successful round)**.** *We say a round r is an* honest successful *round, if in round r, at least one honest PoS-playerare allowed to extend the core-chain.*

Let p_{good} be the probability that a round is honest successful round. We have $p_{good} = 1-(1-p)^{(1-\rho)n}$. In the case that $np \ll 1$, we have $X \approx p(1-\rho)n$. That is $p_{good} \approx \alpha_0$. In the following sections, we assume the probability that a round is honest successful round is α_0 directly.

Definition 3.7 (Best public chain)**.** *Consider round r. We say a chain $\mathcal{C}$ is a* public chain in round r *if such $\mathcal{C}$ is known by all honest players in round r. We say chain $\mathcal{C}$ is the* best public chain in round r *if it is the longest public chain in round r.*

3.3.3 Analysis with bounded delay

We assume that the malicious parties can delay messages up to Δ number of rounds. (This is guaranteed by $\mathcal{F}_{\mathsf{NET}}$.) When an honest PoS-player is qualified to generate a new PoS block-core, he will broadcast it to the system and expect all parties to receive it. The honest players may not obtain the best PoS core-chain and thus work on a different PoS core-chain. If an honest players produce a new PoS block-core during the delay time and later receive a better PoS block-core, the PoS block-core will be useless and his effort during these time slots is wasted. In this subsection, we provide a formal analysis for our core-chain protocol in the presence of the network delay.

Hybrid expriment To analyze the best strategy of the adversary, and the worst scenario that may happen to the honest players, we consider the following notations.

Let $\mathsf{REAL}(\omega) = \mathsf{EXEC}_{\Pi^{\mathrm{core}},\mathcal{A},\mathcal{Z}}(\omega)$ denote the typical execution of Π^{core} where

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

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

- The randomness is fixed to ω as in $\mathsf{HYB}^r(\omega)$,
- $\mathcal{F}_{\mathsf{NET}}$ delays all messages generated by *honest* PoS-players to exact Δ rounds,
- 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,
- Whenever some message is being delayed, no *honest* PoS-players query the functionality $\mathcal{F}_{\mathsf{rCERT}}$ until the message is delivered.

In $\mathsf{REAL}(\omega)$, the number of honest successful rounds is not less than in the $\mathsf{HYB}^r(\omega)$. The following lemma shows that the real execution is not worse than hybrid execution. In order to distinguish core-chain in $\mathsf{HYB}^r(\omega)$ with in $\mathsf{REAL}(\omega)$ executions, we use $\mathcal{C}_{\mathrm{hybrid}}$ to denote it.

Claim 3.8. *For all $\omega, r, t > 0$, given two executions $\mathsf{REAL}(\omega)$ and $\mathsf{HYB}^r(\omega)$. Let $r' = r+t$. For any honest PoS-player P at round r', let $\mathcal{C}'$ denote the PoS core-chain of P at round r' in the execution*

$\mathsf{REAL}(\omega)$ *and* $\mathcal{C}'_{\text{hybrid}}$ *denote the PoS core-chain of* P *at round* r' *in the* $\mathsf{HYB}^r(\omega)$*. We then have* $\text{len}(\mathcal{C}') \geq \text{len}(\mathcal{C}'_{\text{hybrid}})$.

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 $\text{len}(\mathcal{C}) \geq \text{len}(\mathcal{C}_{\text{hybrid}})$. We suppose it holds for all players before round $s-1$. The only case that $\text{len}(\mathcal{C}^s) < \text{len}(\mathcal{C}^s_{\text{hybrid}})$ is the player P received a new core-chain to extend $\mathcal{C}^s_{\text{hybrid}}$ at round s in $\mathsf{HYB}^r(\omega)$. According to the definition of hybrid experiment, this extended PoS block-core must be generated at round $s-\Delta$ by an honest player P_*, that makes $\text{len}(\mathcal{C}^s_{\text{hybrid}}) = \text{len}(\mathcal{C}^{s-\Delta}_{\text{hybrid}}) + 1$. At the same time, the player P_* must succeed to extend PoS block-core at round $s-\Delta$ in $\mathsf{REAL}(\omega)$. This extension will make $\mathcal{C}^{s-\Delta}_*$ increase by one block. For player P_* is honest, P must have received the extension at (or before) round r'. Putting them together, we have $\text{len}(\mathcal{C}') \geq \text{len}(\mathcal{C}'^{\Delta})$. ☐

Analysis in the worst delay setting As mentioned earlier, the malicious players can delay the messages for at most Δ rounds. As a consequence, some efforts from honest players may be wasted. Below we develop a lemma for the "discount" version of honest players' efforts in the execution of $\mathsf{HYB}^r(\omega)$.

Claim 3.9. *Consider* $\mathsf{HYB}^r(\omega)$ *where the adversary is allowed to delay messages for at most* Δ *rounds. Let* $\alpha_0 > 0$ *be the expected number of honest stakeholders that are chosen in a round. Let* α *be the actual probability that a round* $s > r$ *is an honest successful round. Then we have that* $\alpha = \frac{\alpha_0}{1+\Delta\alpha_0}$.

Proof. In $\mathsf{HYB}^r(\omega)$, if round r', where $r' > r$, is an *honest successful round*, then no PoS-players will query functionality $\mathcal{F}_{\mathsf{rCERT}}$ in the next Δ rounds. Now, assume in $\mathsf{HYB}^r(\omega)$, there are c number of honest successful rounds, from round r to round $(r+t)$, where $t > 0$. We then have the number of actual working rounds for honest stakeholders will remain $t - \Delta c$. For each round, the probability that it is an honest successful round is α_0. We have $\alpha_0(t - \Delta c) = c$. This implies that $c = \frac{\alpha_0 t}{1+\Delta\alpha_0}$. We then have $\alpha = \frac{\alpha_0}{1+\Delta\alpha_0}$. ☐

Let $\texttt{VIEW}^r$ denote the `VIEW` at round r in $\mathsf{REAL}(\omega)$ where $r > 0$. Let $\text{len}(\texttt{VIEW}^r)$ denote the length of the best public PoS core-chain in $\texttt{VIEW}^r$. The following lemma demonstrates that each successful round would contribute one PoS block-core to the best public PoS core-chain after Δ rounds in an execution of $\mathsf{HYB}^r(\omega)$.

Claim 3.10. *Consider* $\mathsf{HYB}^r(\omega)$*. For any honest successful round* s*, where* $s > r$*, it holds that* $\text{len}(\texttt{VIEW}^{s+\Delta}) - \text{len}(\texttt{VIEW}^s) \geq 1$.

Proof. By Definition 3.6, there is at least one honest PoS-player producing a PoS block-core at round s. Let $\mathcal{C}^s_{\text{hybrid}}$ be the PoS core-chain that is extended by the PoS-player at round s. We have $\text{len}(\mathcal{C}^s_{\text{hybrid}}) \geq \text{len}(\texttt{VIEW}^s)$. At the end of round s the honest player will broadcast the extended chain with length $\text{len}(\mathcal{C}^s_{\text{hybrid}}) + 1$. At the end of round $s + \Delta$, all honest players will receive the extended core-chain, we have $\text{len}(\texttt{VIEW}^{s+\Delta}) \geq \text{len}(\mathcal{C}^{s+\Delta}_{\text{hybrid}}) = \text{len}(\mathcal{C}^s_{\text{hybrid}}) + 1$. Putting them together, we have $\text{len}(\texttt{VIEW}^{s+\Delta}) - \text{len}(\texttt{VIEW}^s) \geq 1$. ☐

Corollary 3.11. *Consider* $\mathsf{HYB}^r(\omega)$*. Assume there are* h *number of honest successful rounds from round* r *to round* $r+t$ *where* $t > 0$*. Then it holds that* $\text{len}(\texttt{VIEW}^{r+t+\Delta}) - \text{len}(\texttt{VIEW}^r) \geq h$.

Proof. Let r_k be the kth honest successful round where $r < \texttt{round}_k < r+t$ and $1 \leq k \leq h$. From Claim 3.10, we have $\text{len}(\texttt{VIEW}^{\texttt{round}_k+\Delta}) - \text{len}(\texttt{VIEW}^{\texttt{round}_k}) \geq 1$. Then we have $\text{len}(\texttt{VIEW}^{r+t}) - \text{len}(\texttt{VIEW}^{r}) \geq \sum_{i=1}^{h}(\text{len}(\texttt{VIEW}^{\texttt{round}_k+\Delta}) - \text{len}(\texttt{VIEW}^{\texttt{round}_k})) \geq h$. □

If we consider a long time running, we have $t \gg \Delta$. In this case we can ignore Δ rounds difference, that is $\text{len}(\texttt{VIEW}^{r+t}) - \text{len}(\texttt{VIEW}^{r}) \geq h$.

3.3.4 Achieving chain growth property

We here demonstrate that our core-chain protocol satisfies the growth property (Definition 2.3). The concrete statement to be proved can be found in Lemma 3.2. We next first develop some useful lemmas.

Claim 3.12. *Consider* $\mathsf{HYB}^r(\omega)$*, and* $\delta > 0$*. Let* X *be the number of honest successful rounds from round* r *to round* $r+t$*, where* $t > 0$*. Then we have* $\Pr[X > (1-\delta)\alpha t] > 1 - e^{-\Omega(t)}$*.*

Proof. Based on Claim 3.9, we have that, on average, there are αt number of honest successful rounds in any t consecutive rounds. By Chernoff bound, we have $\Pr[X \leq (1-\delta)\alpha t] \leq e^{-\delta^2 \alpha t/2}$. Thus, we have $\Pr[X > (1-\delta)\alpha t] > 1 - e^{-\delta^2\alpha t/2} = 1 - e^{-\Omega(t)}$. □

Claim 3.13. *Consider* $\mathsf{HYB}^r(\omega)$ *and* $\delta > 0$*. Consider an honest PoS-player*P *with the best PoS core-chain* $\mathcal{C}_{\text{hybrid}}$ *in round* r*, and an honest PoS-player*P' *with the best PoS core-chain* $\mathcal{C}'_{\text{hybrid}}$ *in round* r'*, respectively, where* $r' - r \gg \Delta$*. Then we have*

$$\Pr\left[\text{len}(\mathcal{C}'_{\text{hybrid}}) - \text{len}(\mathcal{C}_{\text{hybrid}}) \geq g \cdot t\right] \geq 1 - e^{-\Omega(t)}$$

where $t = r' - r$ *and* $g = (1-\delta)\alpha$*.*

Proof. First, we note that $\mathcal{C}_{\text{hybrid}}$ will be received by all honest players no later than round $r+\Delta$ because player P is honest. We have $\text{len}(\mathcal{C}_{\text{hybrid}}) \leq \text{len}(\texttt{VIEW}^{r+\Delta})$. Now we consider the chain growth from round $r+\Delta$ to round r'. For $t \gg \Delta$, we have $t \approx t - \Delta$ for simplicity. From Claim 3.12, in any t consecutive rounds the number of honest successful round is more than $(1-\delta)\alpha t$ with the probability at least $1 - e^{-\Omega(t)}$. Together with Claim 3.10 and Corollary 3.11, we have $\text{len}(\texttt{VIEW}^{r'}) - \text{len}(\texttt{VIEW}^{r+\Delta}) \geq (1-\delta)\alpha t$. Chain $\mathcal{C}'_{\text{hybrid}}$ is an valid PoS core-chain accepted by an honest PoS-playerP' at round r'. We have $\text{len}(\mathcal{C}'_{\text{hybrid}}) \geq \text{len}(\texttt{VIEW}^{r'})$. Putting these together, we get $\text{len}(\mathcal{C}'_{\text{hybrid}}) - \text{len}(\mathcal{C}_{\text{hybrid}}) \geq \text{len}(\texttt{VIEW}^{r'}) - \text{len}(\texttt{VIEW}^{r+\Delta}) \geq (1-\delta)\alpha t$ with probability at least $1 - e^{-\Omega(t)}$. The corresponding growth rate is $g = (1-\delta)\alpha$. □

Reminder of Lemma 3.2. *Consider core-chain protocol* Π^{core}*, an honest PoS-player* P' *with best local core-chain* $\mathcal{C}'$ *in round* r'*, and an honest PoS-player* P'' *with best local core-chain* $\mathcal{C}''$ *in round* r''*, where* $r'' > r'$*. Then we have* $\Pr\left[\text{len}(\mathcal{C}'') - \text{len}(\mathcal{C}') \geq g \cdot t\right] \geq 1 - e^{-\Omega(t)}$*, where* $t = r'' - r'$*,* $g = (1-\delta)\alpha$*, and* $\delta > 0$*.*

Proof. In order to distinguish the notation clearly, we use $\mathcal{C}'_{\text{hybrid}}$ and $\mathcal{C}''_{\text{hybrid}}$ to denote the PoS core-chains of the best core-chains of P at round r' and r'' in the execution of $\mathsf{HYB}^r(\omega)$. From Claim 3.13, we have $\Pr[\text{len}(\mathcal{C}''^{)}_{\text{hybrid}} \geq \text{len}(\mathcal{C}'^{)}_{\text{hybrid}} + g \cdot t] \geq 1 - e^{-\Omega(t)}$ where $t = r'' - r'$, in $\mathsf{HYB}^r(\omega)$. We now turn to the core-chain growth property in $\mathsf{EXEC}_{\Pi^{\text{core}},\mathcal{A},\mathcal{Z}}$. From the definition of hybrid execution, we know that all honest players have same initial status at round r'. We have $\text{len}(\mathcal{C}') = \text{len}(\mathcal{C}'_{\text{hybrid}})$. By Claim 3.8, we have $\text{len}(\mathcal{C}'') \geq \text{len}(\mathcal{C}''_{\text{hybrid}})$. It follows that,

$$\Pr[\text{len}(\mathcal{C}'') \geq \text{len}(\mathcal{C}') + g \cdot t] \geq \Pr[\text{len}(\mathcal{C}''_{\text{hybrid}}) \geq \text{len}(\mathcal{C}'_{\text{hybrid}}) + g \cdot t] \geq 1 - e^{-\Omega(t)}$$

where $g = (1 - \delta)\alpha$. This completes the proof. □

3.3.5 Achieving 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 core-chain has a lower bound. We then find the lower bound of the number of PoS block-cores produced by the honest players. We further show that the number of block-cores produced by the adversarial miners is bounded by the number of their stakes. Finally, we demonstrate that the ratio of honest PoS block-cores in an honest player's PoS core-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 core-chain and the number of rounds.

Claim 3.14. *Consider* $\mathsf{REAL}(\omega)$*, and* $\delta > 0$*. Let* Z *be the number of rounds in which* ℓ *consecutive block-cores are generated. Then we have* $\Pr[Z > (1-\delta)c\ell] > 1 - e^{-\Omega(\ell)}$ *where* $c = \frac{1}{\alpha+\beta}$*.*

Proof. All players can extend $\alpha + \beta$ number of PoS block-cores in a round on average. In order to generate ℓ block-cores, it will consume $\frac{\ell}{\alpha+\beta}$ rounds on average. Let $c = \frac{1}{\alpha+\beta}$, and Z be the number of rounds which generate the ℓ consecutive PoS block-cores. 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 from honest players in any consecutive block-cores. If the adversarial players want to contribute more PoS block-cores on the core-chain, they will try to generate more PoS block-cores and beat the PoS block-cores from honest players in the competition. Thus, the worst case is the adversarial players make use of all the stakes to generate PoS block-cores and win all of the competition. First, we will prove the core-chain quality property in any t consecutive rounds.

Claim 3.15. *Consider* $\mathsf{REAL}(\omega)$*, and an honest PoS-player*P *with PoS core-chain* $\mathcal{C}$*. Consider* ℓ *consecutive PoS block-cores of* $\mathcal{C}$ *that are generated from round* r *to round* $r + t$*. Assume* $\alpha = \lambda\beta$ *where* $\lambda > 1$*. Then we have* $\Pr[\mu \geq 1 - (1+\delta)\frac{1}{\lambda}] > 1 - e^{-\Omega(t)}$ *for any* $\delta > 0$*, where* μ *is the ratio of honest block-cores of the PoS core-chain* $\mathcal{C}$*.*

Proof. Consider the ℓ consecutive PoS block-cores of $\mathcal{C}$ that are generated from round r to round $r + t$. From Lemma 3.2, we have $\Pr[\ell \geq (1 - \delta^*)\alpha \cdot t] \geq 1 - e^{-\Omega(t)}$ for any $\delta^* > 0$. Let Y be the number of valid malicious PoS block-cores which are actually generated in t rounds to extend a core-chain. By Chernoff bound, we have

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

We then have

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

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

$$\Pr\left[\mu \geq 1 - (1+\delta)\frac{1}{\lambda}\right] > 1 - e^{-\Omega(t)}$$

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

Now we are ready to prove the core-chain quality property for consecutive block-cores on a core-chain.

Reminder of Lemma 3.3. *Consider $\alpha = \lambda\beta$, $\lambda > 1$, and $\delta > 0$. Consider core-chain protocol Π^{core}, and an honest PoS-player with core-chain $\mathcal{C}$. Consider that ℓ consecutive block-cores of $\mathcal{C}$, where ℓ_{good} block-cores are generated by honest PoS-players. Then we have $\Pr\left[\frac{\ell_{good}}{\ell} \geq \mu\right] \geq 1 - e^{-\Omega(\ell)}$, where $\mu = 1 - (1+\delta)\frac{1}{\lambda}$.*

Proof. Let t be the rounds that the ℓ block-cores are generated. From Claim 3.14, we have $\Pr[t > (1-\delta)c\ell] > 1 - e^{-\Omega(\ell)}$. From Claim 3.15, the ratio of honest PoS block-cores in t consecutive rounds with ℓ PoS block-cores is $\mu \geq 1 - (1+\delta)\frac{1}{\lambda}$ with probability at least $1 - e^{-\Omega(t)}$. Putting them together, the probability is at least $1 - e^{-\Omega(\ell)}$. This completes the proof. □

3.3.6 Achieving common prefix property

We now turn to proving the common prefix property (Definition 2.4) for the core-chain protocol Π^{core}. The concrete statement can be found in Lemma 3.4. Before providing our formal proof, we here give some informal proof ideas. First, from the assumption, we know that if the malicious parties do not get any help from the honest parties, then they cannot produce more PoS block-cores than the honest parties do. That means if the malicious parties maintain a hidden, forked core-chain, and try to extend it by themselves, then the hidden core-chain will be shorter than the public core-chain. As the assumption $\alpha + \beta \ll 1$, in most rounds there is no new block being generated. This means the honest players will have same view in most rounds. All of the honest will be used to extend the same chain. This will guarantee that the best public chain will dominate the system. All of the honest players will accept the best public chain.

Recall the definition of best public PoS core-chain $\mathcal{C}$: a) $\mathcal{C}$ has been received by all of the honest players which means public. b) $\mathcal{C}$ is the best one among all of the public core-chains. This implies each honest player will not take any core-chain worse than best public core-chain in any round . Before our proof, we need to define the divergent length of two different chains.

Definition 3.16 (Divergent length)**.** *Given two different core-chain $\mathcal{C}'$ and $\mathcal{C}''$. Let B be the last common block on $\mathcal{C}'$ and $\mathcal{C}''$. Let ℓ' be the length from B to the end of $\mathcal{C}'$ and ℓ'' be the length from B to the end of $\mathcal{C}''$. The divergent length of $\mathcal{C}'$ and $\mathcal{C}''$ is $\ell = max\{\ell', \ell''\}$.*

Claim 3.17. *Let $\alpha = \lambda\beta$, $\lambda > 1$ and $(\alpha+\beta)\Delta \ll 1$, exists $\delta > 0$. Consider $\mathsf{REAL}(\omega)$. Let $\mathcal{C}$ be the best public core-chain in round r. Let $\mathcal{C}'$ be another valid core-chain which is different with $\mathcal{C}$. Let ℓ be the divergent length of $\mathcal{C}$ and $\mathcal{C}'$. We have $\Pr[\text{len}(\mathcal{C}) - \text{len}(\mathcal{C}') > (1-\delta)\ell] > 1 - e^{-\Omega(\ell)}$.*

Proof. Suppose the last common block-core of $\mathcal{C}$ and $\mathcal{C}'$ is generated in round $s = r - t$. From Claim 3.14, we have $t > (1-\delta)\frac{\ell}{\alpha+\beta}$ with probability no less than $1 - e^{-\Omega(\ell)}$. Let $X = \text{len}(\mathcal{C}) - \text{len}(\mathcal{C}^s)$ be the length growth of best public core-chain in the t rounds, with Lemma 3.2, we have

$X > (1-\delta)\alpha t$ with probability no less than $1 - e^{-\Omega(t)}$ During the t rounds, all the players will generate $(\alpha+\beta)t$ block-cores which are longer than core-chain $\mathcal{C}^s$ on average. With the network delay, this will confuse the honest players $(\alpha+\beta)\Delta t$ rounds on average. That is the honest players may contribute to other core-chain during the confusing rounds. Let Y be the block-cores that the honest players contribute during the confusing rounds. We have $Y = (\alpha+\beta)\Delta t\alpha$ on average. For $(\alpha+\beta)\Delta \ll 1$, we have $Y \ll X$. Let Z be the number of block-cores that malicious players can extend for a core-chain during the t rounds. We have $Z = \beta t$ on average. With Chernoff bounds, we have $Z < (1+\delta)\beta t$ with probability no less than $1 - e^{-\Omega(t)}$. Putting these together, we have $\Pr[X - (Y+Z) > (1-\delta)\frac{\lambda-1}{\lambda+1}\ell] > 1 - e^{-\Omega(t)} = 1 - e^{-\Omega(\ell)}$. For $\lambda > 1$, we have $\Pr[\mathrm{len}(\mathcal{C}) - \mathrm{len}(\mathcal{C}') > (1-\delta)\ell] = \Pr[X - (Y+Z) > (1-\delta)\ell] > 1 - e^{-\Omega(\ell)}$. This completes the proof. □

Claim 3.18. *Let $\alpha = \lambda\beta$, $\lambda > 1$ and $(\alpha+\beta)\Delta \ll 1$. Consider $\delta > 0$. Consider $\mathsf{REAL}(\omega)$. Let $\mathcal{C}$ be the best public core-chain in round r. Let $\mathcal{C}'$ be another valid core-chain which is different with $\mathcal{C}$. Let ℓ be the divergent length of $\mathcal{C}$ and $\mathcal{C}'$. Consider a round $r' = r + t$ where $t > 0$, let X be the probability that $\mathcal{C}'$ be a prefix of a chain in round r' which is no worst than the best public core-chain. We have $\Pr[X] < e^{-\Omega(\ell)}$.*

Proof. From Claim 3.17, we have $\Pr[\mathrm{len}(\mathcal{C}) - \mathrm{len}(\mathcal{C}') > (1-\delta)\ell] > 1 - e^{-\Omega(\ell)}$. For $\mathcal{C}'$ is worse than the best public core-chain from round r, the honest players will not extend it. In t rounds the malicious players can extend βt block-cores on average. Meanwhile, in the t rounds, the best public core-chain will increase αt block-cores on average. We have $\Pr[(\beta-\alpha)t > 0] < e^{-\Omega(t)}$. In order to fix the distance of ℓ block-cores, the malicious players will use $\frac{\ell}{\beta}$ rounds with probability no less than $1 - e^{-\Omega(\ell)}$ rounds. At the same time the best public core-chain will increase more than ℓ block-cores with probability no less than $1 - e^{-\Omega(\ell)}$. We have that the core-chain $\mathcal{C}'$ will exceed the best public core-chain in length with probability no more than $e^{-\Omega(\ell)}$. □

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

Reminder of Lemma 3.4. *Consider $\alpha = \lambda\beta$, $\lambda > 1$, and $\delta > 0$. Consider core-chain protocol Π^{core}. and two honest PoS-players, P in round r and P' in round r', with the local best core-chains $\mathcal{C}$, $\mathcal{C}'$, respectively, where $r' \geq r$. Then we have $\Pr[\mathcal{C}[1,\ell] \preceq \mathcal{C}'] \geq 1 - e^{-\Omega(\kappa)}$, where $\ell = \mathrm{len}(\mathcal{C}) - \Theta(\kappa)$.*

Proof. Let $\mathcal{C}^r_{\text{public}}$ be the best public core-chain in round r. For $\mathcal{C}$ is accepted by a player it is must be better than $\mathcal{C}^r_{\text{public}}$. Let ℓ be the divergent length of $\mathcal{C}$ and $\mathcal{C}^r_{\text{public}}$, from Claim 3.17 we have $\ell < \kappa$. Otherwise, $\mathcal{C}$ is better than $\mathcal{C}^r_{\text{public}}$ with negligible probability. In round r', $\mathcal{C}'$ is accepted by a honest player. It must be no worse than best public core-chain. We use $\mathcal{C}'^r$ to denote the prefix of core-chain in round r. Let ℓ be the divergent length of $\mathcal{C}^r_{\text{public}}$ and $\mathcal{C}'^r$. From Claim 3.18, we have $\ell < \kappa$. Otherwise, $\mathcal{C}'$ is better than public core-chain in round r' with a low probability. Putting these together, we have that both $\mathcal{C}$ and $\mathcal{C}'^r$ are divergent with $\mathcal{C}^r_{\text{public}}$ less than κ block-cores. That is $\mathcal{C}$ and $\mathcal{C}'^r$ are divergent less than κ block-cores. This completes the proof. □

3.3.7 Achieving chain soundness property

We now turn to proving the chain soundness property for the core-chain protocol $\Pi^{\rm core}$. The concrete statement can be found in Lemma 3.5, and will be restated below. Before providing the formal proof, we here give some informal proof ideas. As in PoW-based blockchain protocols, in our $\Pi^{\rm core}$, all players follow the longest chain. That is, the longest chain is the best chain. The malicious players cannot create a chain which grows faster than the best public chain. Therefore, the malicious players cannot mislead new players by providing them a longer chain.

Reminder of Lemma 3.5. *Consider for every round, $\alpha = \lambda\beta$, $\lambda > 1$, and $\delta > 0$. Consider core-chain protocol $\Pi^{\rm core}$. Consider two honest PoS-players, P' and P'' in round r, with the local best core-chains $\mathcal{C}'$ and $\mathcal{C}''$, respectively, where P' is a new player and P'' is an existing player in round r. Then we have $\mathcal{C}'[\neg\kappa] \preceq \mathcal{C}''$ and $\mathcal{C}''[\neg\kappa] \preceq \mathcal{C}'$.*

Proof. Let $\mathcal{C}$ be the best public chain in round r. This implies that both P' and P'' have already received the public best $\mathcal{C}$. Let ℓ' be the divergent length of $\mathcal{C}'$ and $\mathcal{C}$. From Claim 3.17, we have $\Pr[\mathrm{len}(\mathcal{C}) - \mathrm{len}(\mathcal{C}') \geq (1-\delta)\ell'] > 1 - e^{-\Omega(\ell')}$. If $\mathcal{C}'[\neg\kappa] \not\preceq \mathcal{C}$, we have $\mathrm{len}(\mathcal{C}) > \mathrm{len}(\mathcal{C}')$ with probability no less than $1 - e^{-\Omega(\kappa)}$. This contradicts the fact that P_i already took $\mathcal{C}'$ as the best chain. Therefore, we have $\mathcal{C}'[\neg\kappa] \preceq \mathcal{C}$. Similarly, we can have $\mathcal{C}''[\neg\kappa] \preceq \mathcal{C}$. Note that $\mathrm{len}(\mathcal{C}') \geq \mathrm{len}(\mathcal{C})$, this means that the divergent length of $\mathcal{C}$ is shorter than $\mathcal{C}'$. We now have $\mathcal{C}[\neg\kappa] \preceq \mathcal{C}'$. Similarly, we can have $\mathcal{C}[\neg\kappa] \preceq \mathcal{C}''$. Putting these together, we obtain $\mathcal{C}'[\neg\kappa] \preceq \mathcal{C}''$ and $\mathcal{C}''[\neg\kappa] \preceq \mathcal{C}'$ which completes the proof. □

4 Securing the core-chain against a greedy adversary

In Section 3, for the simplicity of presentation, we consider the setting that all players follow the basic strategy to extend the core-chain. That is, each player will make attempts to extend the *single* best chain in his/her local view. However, in the proof-of-stake setting, it is "very cheap" to extend chains; a proof-of-stake player may take a **greedy** strategy to extend the core-chains: he/she will make attempts to extend *a set of chains* in his/her local view, and expects to obtain additional advantage for extending the best chain.

In this section, we will formally study greedy strategies. We will first define the greedy strategies in Subsection 4.1, and then present a modified core-chain protocol in Subsection 4.2; after that, we will analyze the security of our modified protocol in Subsection 4.3.

4.1 Greedy strategies

We now define greedy strategies for a player. Consider a blockchain protocol execution. There are multiple chains in a player's local view; without loss of generality, these chains can be viewed as a *tree*: the root of the tree is the genesis block, and each path from the root is a chain. The tree will "grow" round after round: the length of each existing chain may increase, and new chains will be created. Let ℓ be the length of the longest chain at round r. A g-greedy player will attempt to extend a *set of chains* in which all chains have the length at least $(\ell - \mathrm{g})$, where $0 \leq \mathrm{g} \leq \ell$. More formally, we have the following definition.

Let ℓ be the length of the longest chain at round r. Consider greedy parameter g where $0 \leq \mathrm{g} \leq \ell$. We say the player is g-*greedy* if, for all round r, the player makes attempts to extend a *set of chains* in which all chains have the length at least $(\ell - \mathrm{g})$. Note that, when $\mathrm{g} = 0$, the g-greedy is essentially the basic strategy that we considered in Π^{core}. When $\mathrm{g} = \ell$, we say the protocol player is *fully-greedy.*

Definition 4.1 (g-greedy strategy)**.** *Consider a blockchain protocol execution. Let* P *be a player of the protocol execution, and* T *be a tree which consists of chains with the same genesis block, in* P*'s local view. Let* ℓ *be the length of the longest chain at round* r*. We say the player is* g-greedy *if, for all* r*, the player makes attempts to extend all chains with length at least* $(\ell - \mathrm{g})$*, where* $0 \le \mathrm{g} \le \ell$*.*

If $\mathrm{g} = \ell$*, we say the player follows the* fully-greedy *strategy; if* $\mathrm{g} = 0$*, we say the player follows the* basic *(greedy) strategy.*

Different greedy strategies are illustrated in Figures 5, 6 and 7. The players with different strategies will try to extend the blocks in the different red rectangles. In Figure 5, the players follow the basic strategy (0-greedy strategy), i.e., the players only make attempts to extend the longest chain. In Figure 6, players follow the 1-greedy strategy; that is, the players attempt to extend the longest chain as well as the chains with only one block behind. In Figure 7, the players follow the full-greedy strategy; the players make attempts to extend all chains.

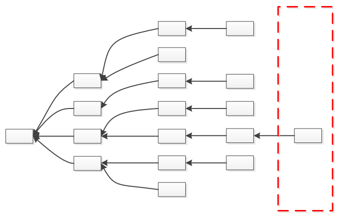
Figure 5: Basic strategy (i.e., 0-greedy strategy)

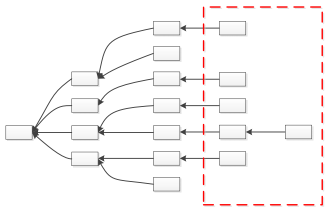
Figure 6: 1-greedy strategy

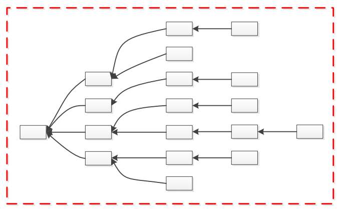
Figure 7: Fully-greedy strategy

As we mentioned before, when a player follows a greedy strategy, he may extend the chains faster. Next, we introduce *amplification ratio.*

Definition 4.2 (Amplification ratio)**.** *Consider a PoS blockchain protocol. Let* N_0 *be the number of blocks that a group of players* P *contribute to extend a blockchain in a fixed number of rounds if they follow the* 0*-greedy strategy on average. Let* N_{g} *be the number of blocks that the same group of players* P *contribute to extend a blockchain in the same time period if they follow the* g*-greedy strategy on average. We define the amplification ratio for following the* g*-greedy strategy as* $\mathtt{A}_{\mathrm{g}} = \frac{N_{\mathrm{g}}}{N_0}$

Jumping ahead, in next sections, we will show that the amplification ratio for following the fully-greedy strategy is $\mathtt{A}_{\mathrm{fully}} = 2.718$, and the amplification ratio for following the 2-greedy strategy is $\mathtt{A}_2 = 2.1$. In addition, by definition, $\mathtt{A}_0 = 1$.

4.2 The modified core-chain protocol $\Pi^{\mathrm{core}\star}$

Next, we modify the core-chain protocol Π^{core} in Section 3 into a new core-chain protocol $\Pi^{\mathrm{core}\star}$ where players follow the g-greedy strategy. Details can be found in Figure 8.

We note that, the subroutine BestCore has also been modified into subroutine BestCore*; now the modified subroutine BestCore*, instead of returning the single longest chain, will output a set of chains including the longest chain, and several chains that are slightly (i.e., g blocks) shorter than the longest chain. Intuitively, the bigger the greedy parameter g is, the better the chance

```
PROTOCOL $\Pi^{\text{core}\star}$

Initially, a set $\mathcal{P}_0$ of players are registered to the functionality $\mathcal{F}_{\text{rCERT}}$, where $\mathcal{P}_0 \subseteq \mathcal{P}$.
Initially, for each $\mathrm{P} \in \mathcal{P}$, set $\mathcal{C} := \emptyset$, and $state := \emptyset$.

Upon receiving message $(\text{INPUT-STAKE}, \mathrm{P})$ from the environment $\mathcal{Z}$ at round $\texttt{round}$, the PoS-player $\mathrm{P} \in \mathcal{P}$, with local state $state$, proceeds as follows.

    1. Select the best local PoS core-chain:
       Let $\mathbb{C}$ be the set of core-chains collected from $\mathcal{F}_{\text{NET}}$.
       Compute $\mathbb{C}_{\text{best}} := \text{BestCore}^\star(\mathbb{C} \cup \{\mathcal{C}\}, \texttt{round})$.
       For each $\mathcal{C} \in \mathbb{C}_{\text{best}}$, and $\ell := \text{len}(\mathcal{C})$: (BestCore$^\star$ will return a best chain set and the players will try to extend all)

    2. Attempt to extend PoS core-chain:
       Parse $\mathcal{C}[\ell]$ as $\langle\langle h_\ell^{\text{prev}}, \texttt{round}_\ell, \mathrm{P}_\ell, \sigma_\ell\rangle, h_\ell\rangle$.

       – Stake election:
             Send $(\text{ELECT}, \mathrm{P}, \langle h_\ell, \texttt{round}\rangle)$ to functionality $\mathcal{F}_{\text{rCERT}}$,
             and receive $(\text{ELECTED}, \mathrm{P}, h_{\ell+1}, \sigma, \mathsf{b})$ from $\mathcal{F}_{\text{rCERT}}$.
       – If $\mathsf{b} = 1$, generate a new block-core:
             Set the new block-core $B := \langle\langle h_\ell, \texttt{round}, \mathrm{P}, \sigma\rangle, h_{\ell+1}\rangle$,
             and set $\mathcal{C} := \mathcal{C} \| B$, and $state := state \cup \{\mathcal{C}\}$,
             and then send $(\text{BROADCAST}, \mathcal{C})$ to $\mathcal{F}_{\text{NET}}$.

       Return $(\text{RETURN-Stake}, \mathrm{P})$ to the environment $\mathcal{Z}$.
```

Figure 8: Our proof-of-stake core-chain protocol $\Pi^{\text{core}\star}$ in the $\{\mathcal{F}_{\text{rCERT}}, \mathcal{F}_{\text{NET}}\}$-hybrid model. (See Figure 9 for the subroutine $\text{BestCore}^\star$.)

that the player extend the set of chains. However, the (computation and storage) complexity of the protocol is proportional to the greedy parameter g. In practice, we can choose $\mathrm{g} = 2$ (and the size of set $\mathbb{C}_{\text{best}}$ is 10 on average).

4.3 Security analysis

In previous section, the security properties of protocol Π^{core} have be proven under the assumption of honest majority of stakes based on α and β. Now, we can prove the security properties of the modified core-chain protocol $\Pi^{\text{core}\star}$ but under the assumption of honest majority of *effective stakes* based on $\alpha^\star$ and $\beta^\star$. Here $\alpha^\star = 2.1\alpha$ and $\beta^\star = 2.718\beta$. We note that here $\alpha^\star \ll 1$ and $\beta^\star \ll 1$ (since $\alpha \ll 1$ and $\beta \ll 1$).

Theorem 4.3 (Theorem 1.3, restated)**.** *Consider core-chain protocol $\Pi^{\text{core}\star}$ where honest players follow the 2-greedy strategy while adversarial players follow the fully-greedy strategy; in addition, all players have their stake registered without being aware of the state of the protocol execution. If $\alpha^\star = \lambda\beta^\star$, $\lambda > 1$, then the protocol $\Pi^{\text{core}\star}$ can achieve chain growth, chain quality, common prefix and chain soundness properties.*

Subroutine BestCore*

The subroutine BestCore* is allowed to access to the functionality $\mathcal{F}_{\mathsf{rCERT}}$, and with input $(\mathbb{C}', \texttt{round}')$.
For every chain $\mathcal{C} \in \mathbb{C}'$, and proceed as follows.

```
1. Set $\ell := \mathrm{len}(\mathcal{C})$.
2. For $i$ from $\ell$ down to 1, verify block-core $\mathcal{C}[i]$, as follows.
   • Parse $\mathcal{C}[i]$ into $\langle\langle h_i^{\mathrm{prev}}, \texttt{round}_i, \mathrm{P}_i, \sigma_i\rangle, h_i\rangle$.
     Parse $\mathcal{C}[i-1]$ into $\langle\langle h_{i-1}^{\mathrm{prev}}, \texttt{round}_{i-1}, \mathrm{P}_{i-1}, \sigma_{i-1}\rangle, h_{i-1}\rangle$.
   • If $\texttt{round}_i < \texttt{round}'$ and $\texttt{round}_{i-1} < \texttt{round}_i$ , then execute:
     – If $h_i^{\mathrm{prev}} \neq h_{i-1}$, then remove this core-chain $\mathcal{C}$ from $\mathbb{C}'$.
     – Else send $(\textsc{Core-Verify}, \mathrm{P}_i, \langle h_i^{\mathrm{prev}}, \texttt{round}_i\rangle, \sigma_i, h_i)$ to $\mathcal{F}_{\mathsf{rCERT}}$.
       Upon receiving message $(\textsc{Core-Verified}, \mathrm{P}_i, \langle h_i^{\mathrm{prev}}, \texttt{round}_i\rangle, \sigma_i, h_i, f_i)$ from
       $\mathcal{F}_{\mathsf{rCERT}}$, if $f_i = 0$ remove this core-chain $\mathcal{C}$ from $\mathbb{C}'$.
     Otherwise, remove the core-chain $\mathcal{C}$ from $\mathbb{C}'$.
```

Let $\mathcal{C}_{\mathrm{best}}$ be the longest core-chain in $\mathbb{C}'$ and $\ell := \mathrm{len}(\mathcal{C}_{\mathrm{best}})$. Set $\mathbb{C}_{\mathrm{best}} = \emptyset$.
For any chain $\mathcal{C} \in \mathbb{C}'$ and $\mathrm{len}(\mathcal{C}) \geq \ell - \mathrm{g}$, $\mathbb{C}_{\mathrm{best}} = \mathbb{C}_{\mathrm{best}} \cup \{\mathcal{C}\}$. (g is a small constant, $\mathrm{g} = 2$ typically.)
Then return $\mathbb{C}_{\mathrm{best}}$ as the output.

Figure 9: The core-chain set validation subroutine BestCore*.

4.3.1 Important lemmas, and effective stakes $\alpha^\star$ and $\beta^\star$

We now show a very interesting lemma (i.e., Lemma 4.4) that the amplification ratio for following the fully-greedy strategy is bounded by a factor e (the base of natural logarithm). Intuitively, if protocol players follow the fully-greedy strategy and extend all chains, one of the relatively shorter chains will become the longest chain with certain probability; that means, the longest chain will be extended faster. However, we note that, the shorter the chain is, the probability of being extended into the longest chain is lower; collectively, the longest chain will strictly increase but will be bounded by a constant factor.

Lemma 4.4. *Consider core-chain protocol $\Pi^{\mathrm{core}\star}$. Assume that malicious players can generate a new block with probability β in a round. Assume that the malicious players follow the fully-greedy strategy to extend a core-chain $\mathcal{C}'$ at round r' into $\mathcal{C}''$ at round r'', where $r'' > r'$. Then we have*

$$\Pr\left[\mathrm{len}(\mathcal{C}'') - \mathrm{len}(\mathcal{C}') < e\beta \cdot t\right] \geq 1 - e^{-\Omega(t)}$$

where $t = r'' - r'$ and e is the base of natural logarithm.

Proof. Let $f(t, l)$ be the number of chains with length $l + \mathrm{len}(\mathcal{C}')$ at round $t + r'$, where $0 \leq t \leq r'' - r'$. The initial conditions are as follows:

- $f(t, 0) := 1$ for all $t \geq 0$;
  it means that, at $t \geq 0$, there is only 1 chain (i.e. $\mathcal{C}'$) which is 0 block longer than $\mathcal{C}'$;

- $f(0,l) := 0$ for all $l > 0$;
  it means that, at $t = 0$, there is <u>no chain</u> which is l blocks longer than $\mathcal{C}'$;

We first discuss the average case. Recall that malicious players can generate a new block with probability β in a round. We have

$$f(t,l) := f(t-1,l) + f(t-1,l-1)\beta$$

on average. Note that, if we view β as a variable, the coefficients of $f(t,l)$ can be viewed as a polynomial, and they will follow the Pascal's Triangle. That is, we have

$$f(t,l) = \binom{t}{l}\beta^l$$

We now develop a simplified form of $f(t,l)$. We note that, in a round there is at most one block that can be extended from an existing block on average. Let k be the number of rounds for generating one block on average, and we have $k > 1$, and $t = kl$. Now we have:

$$\begin{aligned}
f(t,l) &= \frac{t!}{(l!)(t-l)!}\beta^l \\
&\approx \frac{\sqrt{2\pi t}t^t}{(\sqrt{2\pi l}l^l)\cdot(\sqrt{2\pi(t-l)}(t-l)^{t-l})}\cdot\beta^l \\
&= \frac{\sqrt{2\pi kl}(kl)^{kl}}{(\sqrt{2\pi l}l^l)\cdot(\sqrt{2\pi(kl-l)}(kl-l)^{kl-l})}\cdot\beta^l \\
&= \sqrt{\frac{kl}{2\pi l(kl-l)}}\cdot\frac{(kl)^{kl}}{l^l(kl-l)^{kl-l}}\cdot\beta^l \\
&= \sqrt{\frac{k}{2\pi(k-1)l}}\left[\frac{k^k}{(k-1)^{k-1}}\cdot\beta\right]^l
\end{aligned}$$

The second approximate equality above is based on Stirling's approximation.

Note that, $g(k) = \left(1+\frac{1}{k}\right)^k$ is a monotone increasing function and $\lim_{k\to\infty} g(k) = e$. We now have $\left(\frac{k}{k-1}\right)^{k-1} < e$. That is :

$$f(t,l) < \sqrt{\frac{k}{2\pi(k-1)l}}(ke\beta)^l \tag{1}$$

We note that the chains will always increase, and we will have at least one chain with length $l + \mathrm{len}(\mathcal{C}')$; that means, we have $f(t,l) \geq 1$. From Equation 1, we have $\sqrt{\frac{k}{2\pi(k-1)l}}(ke\beta)^l \geq 1$; we can then obtain $ke\beta > 1$. Otherwise, $f(t,l) \ll 1$ for large enough l. Recall that $t = kl$; we now have $l < e\beta \cdot t$.

Next, we turn to the worst case discussion. With Chernoff bound, for any $\delta > 0$, we have $\Pr\left[l > (1+\delta)e\beta\cdot t\right] \leq -e^{\Omega(t)}$. Here $l = \mathrm{len}(\mathcal{C}'') - \mathrm{len}(\mathcal{C}')$. We have $\Pr\left[\mathrm{len}(\mathcal{C}'') - \mathrm{len}(\mathcal{C}') < e\beta\cdot t\right] \geq 1 - e^{-\Omega(t)}$ This concludes the proof. □

Defining $\beta^\star$. We use $\beta^\star$ to denote the equivalent expected number of blocks that malicious players can extend a chain in a round. From Lemma 4.4, we have $\beta^\star = e\beta$. That is, the amplification ratio of malicious players is bounded by e, the base of natural logarithm .

Next, we will show that, when honest players follow the g-greedy strategy and malicious players follow the fully-greedy strategy, the number of blocks that malicious players contribute in a chain is also bounded.

Lemma 4.5. *Consider core-chain protocol $\Pi^{\mathrm{core}\star}$ in the presence of a fully-greedy adversary. Consider a core-chain $\mathcal{C}'$ in round r'; this core-chain is extended to chain $\mathcal{C}''$ in round r'', where $r'' > r'$. Let $t = r'' - r'$. Let X be the number of blocks that are generated by malicious players during the t rounds. Then for any $\delta > 0$, we have $\Pr[X < (1+\delta)\beta^\star t] > 1 - e^{-\Omega(t)}$.*

Proof sketch. First, consider the case that all the honest players follow the fully-greedy strategy. From Lemma 4.4, we have $\mathrm{len}(\mathcal{C}'') - \mathrm{len}(\mathcal{C}') < (1+\delta)e(\alpha+\beta)t$ with overwhelming probability. Furthermore, the honest players, playing the same strategy as that by malicious players, will contribute more than $\frac{\alpha}{\alpha+\beta}(\mathrm{len}(\mathcal{C}'') - \mathrm{len}(\mathcal{C}'))(1-\delta)$ blocks with overwhelming probability. We use Y to denote the number of blocks that are contributed by malicious players. We have $Y < (1+\delta)e\beta t$ blocks with overwhelming probability.

Second, consider the case that the honest players follow the g-greedy strategy where g is a small constant such as 2. In this case the chain will grow slower than that in the previous case, on average. As a result, in any round, the malicious players will have less opportunity to extend the chain than that in the previous case. Therefore, we have $X < Y$. Putting these together, we have $\Pr[X < (1+\delta)\beta^\star t] > 1 - e^{-\Omega(t)}$. □

Defining $\alpha^\star$. We next show another lemma stating that, honest players, by following the 2-greedy strategy, can obtain extra advantage for extending the public chain.

Lemma 4.6. *Consider core-chain protocol $\Pi^{\mathrm{core}\star}$ where greedy parameter is $\mathrm{g} = 2$. Assume malicious players do not help honest players to extend any chain. Let t be the number of rounds for extending the longest chain with one new block. Then we have $t \approx \frac{1}{2.1\alpha}$ on average.*

Proof. Assume that the length of the longest chain at round r is l, and the length of the longest chain at round r' is $l+1$. That is $t = r' - r$. We further assume, in round r, the number of chains with length $l-1$ is x, and the number of chains with length $l-2$ is y . (We also assume only one chain, i.e., the longest chain, is with length l.)

For simplicity, we assume the chain will grow with a steady rate and we here only investigate the average case. In this case with the above assumption, in round r' the length of the longest chain will increase to $l+1$, and the number of chains with length l will be x and the number of chains with length $l-1$ will be y.

Consider the longest chain ,with length $l+1$ in round r', it is extended by some players during the t rounds from a chain with length l. At round r there is 1 chain with length l and at round r' there are x chains with length l. On average, we have $\frac{1+x}{2}\alpha t = 1$. With similar arguments, considering the chain with length l in round r', they are extended from a chain with length l. On average, we have $\frac{x+y}{2}\alpha t = x - 1$. Finally, considering the chain with length $l-1$ in round r', they are extended from a chain with length l. On average, we have $y\alpha t = y - x$. Putting these together, we have:

$$\begin{cases} \frac{1+x}{2}\alpha t = 1 \\ \frac{x+y}{2}\alpha t = x - 1 \\ y\alpha t = y - x \end{cases} \tag{2}$$

That is $\alpha t \approx 0.48$. We get $t \approx \frac{1}{2.1\alpha}$. □

This lemma shows that, if the honest players follow the 2-greedy strategy they will extend the longest chain with 1 block in $t \approx \frac{1}{2.1\alpha}$ rounds on average. If we use $\alpha^\star$ to denote the equivalent expected number of blocks that the honest players will extend a chain in a round, we have $\alpha^\star = 2.1\alpha$. That is, the amplification ratio (of extending chains by honest players) is $\mathtt{A}_2 = 2.1$.

4.3.2 Security properties in the presence of a fully-greedy adversary

Based on the above discussions, we have $\alpha^\star = \mathtt{A}_2\alpha \ll 1$ and $\beta^\star = e\beta \ll 1$, where $\mathtt{A}_2 = 2.1$. Here we assume, $\alpha \ll 1$ and $\beta \ll 1$. We have $\alpha^\star \ll 1$ and $\beta^\star \ll 1$. Then similarly, the security properties will still hold if we change the previous assumption of honest majority of stakes based on α and β, into the assumption of honest majority of *effective stakes* based on $\alpha^\star$ and $\beta^\star$.

Chain growth. Honest players will extend blockchain faster if they follow, not the basic strategy, but the g-greedy strategy, where $\mathrm{g} > 0$, and the chain growth rate will increase. From Lemma 3.2, we have:

Corollary 4.7 (Chain growth). *Consider core-chain protocol $\Pi^{\text{core}\star}$ in the presence of a fully-greedy adversary. Consider an honest PoS-playerP' with best local PoS core-chain $\mathcal{C}'$ in round r', and an honest PoS-playerP'' with best local core-chain $\mathcal{C}''$ in round r'', where $r'' > r'$. Then we have $\Pr\left[\text{len}(\mathcal{C}'') - \text{len}(\mathcal{C}') \geq g \cdot t\right] \geq 1 - e^{-\Omega(t)}$ where $t = r'' - r'$, $g = (1-\delta)\alpha^\star$, and $\delta > 0$.*

Chain quality. A g-greedy adversary can extend a chain faster than basic adversary, when $\mathrm{g} > 0$. Intuitively, this will reduce the chain quality. However, from Lemma 4.5, the number of blocks from malicious players on any chain is bounded. If we assume the honest players extend chains faster than the malicious players, the chain quality property will still hold as in Lemma 3.3.

Corollary 4.8 (Chain quality). *Consider $\alpha^\star = \lambda\beta^\star$, $\lambda > 1$, and $\delta > 0$. Consider core-chain protocol $\Pi^{\text{core}\star}$ with a greedy adversary. Consider an honest PoS-player with PoS core-chain $\mathcal{C}$. Consider that ℓ consecutive block-cores of $\mathcal{C}$, where ℓ_{good} block-cores are generated by honest PoS-players. Then we have $\Pr\left[\frac{\ell_{good}}{\ell} \geq \mu\right] \geq 1 - e^{-\Omega(\ell)}$ where $\mu = 1 - (1+\delta)\frac{1}{\lambda}$.*

Common prefix. We note that, if a fully-greedy adversary cannot extend chains faster than the g-greedy honest players, he cannot generated a longer forked chain to violate the common prefix property. From Lemma 3.4 we have:

Corollary 4.9 (Common prefix). *Consider $\alpha^\star = \lambda\beta^\star$, $\lambda > 1$, and $\delta > 0$. Consider core-chain protocol $\Pi^{\text{core}\star}$ with a fully-greedy adversary. Consider two honest PoS-players, P in round r and P' in round r', with the local best PoS core-chains $\mathcal{C}$, $\mathcal{C}'$, respectively, where $r' \geq r$. Then we have $\Pr\left[\mathcal{C}[1,\ell] \preceq \mathcal{C}'\right] \geq 1 - e^{-\Omega(\kappa)}$ where $\ell = \text{len}(\mathcal{C}) - \Theta(\kappa)$.*

Chain soundness. As in Lemma 3.5, the protocol can achieve chain soundness property. Otherwise the common prefix property will be violated.

Corollary 4.10 (Chain soundness). *Assume for every round, $\alpha^\star = \lambda\beta^\star$, $\lambda > 1$, and $\delta > 0$. Consider core-chain protocol $\Pi^{\text{core}\star}$ with a fully-greedy adversary. Consider two honest PoS-players, P' and P'' in round r, with the local best core-chains $\mathcal{C}'$ and $\mathcal{C}''$, respectively, where P' is a new player and P'' is an existing player in round r. Then we have $\mathcal{C}'[\neg\kappa] \preceq \mathcal{C}''$ and $\mathcal{C}''[\neg\kappa] \preceq \mathcal{C}'$.*

4.3.3 Simulation of greedy strategy

In this section, through a simulation (in Python), we illustrate the amplification ratio when a player follows the 2-greedy strategy, instead of following the basic strategy (i.e., 0-greedy strategy). The simulation result fits the theoretical calculation well; please see Table 1 for details. In our simulation, we consider in the system, there are 10000 honest players, and the difficulty target $\mathtt{T} = \frac{1}{100000}$. In addition, we consider parameter $\alpha = 0.1$, and the honest players follows the 2-greedy strategy. We run the simulation for 1000 rounds as a test, and we collect the data from 5 tests and calculate the mean value.

Table 1: Simulation with the 2-greedy strategy

	0-greedy	2-greedy	Amplification ratio
Theoretical Value	100	210	2.1
Simulated Mean Value	100.4	210.2	2.1

From above discussion, we have that the amplification ratio is 2.1 when players follow the 2-greedy strategy. Recall that, from previous section, we have that the amplification ratio is $e = 2.718$ when players follow the fully-greedy strategy. In addition, by definition, the amplification ratio is 1 when players follow the basic strategy (0-greedy strategy). Now we can obtain the following result (see Table 2): when honest players follow the 0-greedy (or, 0-greedy, 2-greedy, respectively) strategy, and malicious players follow the 0-greedy (or, fully-greedy, fully-greedy, respectively) strategy, to ensure the security of the core-chain protocol, 51% (or, 73%, 57%, respectively) majority of stakes must be honest.

Table 2: Honest majority for ensuring security

Honest Players	Malicious Players	Honest Majority (ensuring security)
0-greedy	0-greedy	51%
0-greedy	fully-greedy	73%
2-greedy	fully-greedy	57%

5 Securing the core-chain against an adaptive adversary

In previous sections (Sections 3 and 4), we assume that all players must generate their key-pairs (so that they can have their stakes registered), before they join and be aware of the state of the protocol execution. The process of extending the chains is based on the hash inequality $\mathsf{H}(context, solution) < \mathtt{T}$. Note that, there players are not aware of $context$ before generating the keys; the unpredictability of H will ensure that all players have the same probability to find a solution.

In this section, we consider the practical setting where players are allowed to have their stakes registered *during the protocol execution*. The protocols in the previous sections will not work; the adversary now knows the context, and he can play a "rejection re-sampling" strategy to generate their keys adaptively. More concretely, the adversary first runs key generation algorithm to obtain a key-pair $(\textsc{pk}, \textsc{sk})$, and then check if the corresponding $(\textsc{pk}, \sigma)$ is a valid solution; if not, the adversary will repeat the process. By adopting this strategy, malicious players can increase the probability that their stakes are chosen to extend the chain. To defend against this serious rejection re-sampling attack, we will modify the protocol $\Pi^{\text{core}\star}$. We will propose the strategy that players are allowed to extend the chains only if they have had their stakes registered many rounds earlier.

5.1 Setup functionality $\mathcal{F}^{\bullet}_{\mathsf{rCERT}}$

In our core-chain protocol design, we will use a modified setup functionality, resource certification functionality $\mathcal{F}^{\bullet}_{\mathsf{rCERT}}$ (in Figure 10). Our idea is that we only allow players who registered rounds earlier to be qualified to extend the chain. This will effectively discourage players to register accounts adaptively to gain advantage. In order to achieve this, functionality $\mathcal{F}^{\bullet}_{\mathsf{rCERT}}$ must keep track of the exact position/round when an account is registered.

Compared with the original $\mathcal{F}_{\mathsf{rCERT}}$, the players in the set $\mathcal{P}$ are registered in a tree structure $\mathcal{T}$. The tree structure $\mathcal{T}$ is an abstraction of the real blockchains jointly from all real players. The root of $\mathcal{T}$ is corresponding to the genesis block of the blockchain. A node on the tree is mapped from a block on a blockchain. When a player generate a block successfully, $\mathcal{F}^{\bullet}_{\mathsf{rCERT}}$ will also generate a corresponding node for the tree. A `path` of the tree from root to a node corresponds to a real blockchain. The registration information of a valid player is confirmed by a block on a blockchain which is a node on a branch of the tree. The system can keep track of the players registration history with the help of the tree structure $\mathcal{T}$. Furthermore, with the help of $\mathcal{T}$, $\mathcal{F}^{\bullet}_{\mathsf{rCERT}}$ can get the height of the block in which a player is registered; $\mathcal{F}^{\bullet}_{\mathsf{rCERT}}$ can also check if the registration block is an ancestor of the current block. Given a blockchain $\mathcal{C}$ in the protocol the functionality $\mathcal{F}^{\bullet}_{\mathsf{rCERT}}$ can identity the corresponding `path` on $\mathcal{T}$.

As defined in $\mathcal{F}_{\mathsf{rCERT}}$ earlier, the modified functionality $\mathcal{F}^{\bullet}_{\mathsf{rCERT}}$ consists of three phases, "Stake Resource Registration", "Stake Election" and "Stake Verification". At any time, a PoS-player P can send a registration command $(\textsc{Stake-Register}, \mathrm{P}, \mathcal{C})$ to functionality $\mathcal{F}^{\bullet}_{\mathsf{rCERT}}$ for registration where $\mathcal{C}$ is the specified blockchain that P to be registered. Please note that, P will be registered in the new generated leaf node of $\mathcal{C}$ and will also be mapped on the tree $\mathcal{T}$. The functionality then records $(\mathrm{P}, \mathbb{b}_{\mathrm{P}})$ where $\mathbb{b}_{\mathrm{P}} = 1$, if the party P is permitted. Then, for each execution round, a registered PoS-player P is granted *one unit of the stake*, and he can then request the functionality for leader election in this round. Just like in the basic $\mathcal{F}_{\mathsf{rCERT}}$, this is processed in "Stake Election" phase. The difference here is that a player will be verified more carefully to prevent adaptive registration. Intuitively, only a player who registers on the prefix of a chain can be elected to extend a block on this chain. This verification is abstracted as subroutine $\mathsf{StakeVerification}$ (in Figure 11).

Finally, the block verification request $(\textsc{Core-Verify}, \mathrm{P}, \langle h^{\mathrm{prev}}, \texttt{round}\rangle, \sigma, h, \mathcal{C}, i)$ will proceed as follows. In the request, the functionality will verify if the i-th block on the $\mathcal{C}$ is valid. Upon receiving a verification request, the functionality would check if there is a corresponding path on the $\mathcal{T}$ with $\mathcal{C}$. The functionality then verifies if the signature is valid for the block. Then the functionality will check if the player is registered long enough before the block is generated. We use the parameter η to denote that the account is registered η blocks before.

5.2 The modified core-chain protocol $\Pi^{\mathrm{core}\bullet}$

In order to prevent the malicious players from generating and registering a key for the next block adaptively, the protocol is modified as:

- A player is qualified to be elected to sign a block in the functionality $\mathcal{F}^{\bullet}_{\mathsf{rCERT}}$ (see Figure 10), only if his key pair has been registered at least η blocks before.
- Given two divergent blockchains, if the divergent parts are more than η blocks the chain which generates the η blocks earlier is the better one.

With this modification, malicious players cannot register a biased key pair for the following η blocks to increase the probability he will be elected. However, there is still an issue that the malicious players may register a biased key pair for a round, η blocks later. We will prove this issue can be resolved if we further improve the best chain strategy. The intuition is that if the malicious players prepare a biased key pair for a public chain, then the honest player will win some blocks among the η blocks with high probability. The malicious players cannot predict the signature of honest players, so he cannot predict the input of the blocks η blocks after. This means that the malicious players cannot get advantage for η blocks after if the chain is public. If the malicious players decide to extend a hidden blockchain, he can prepare a biased player for a block η blocks after. However, he will lose the chain growth competition for the first η blocks. Hence, we modify the $\mathsf{BestCore}^{\star}$ into $\mathsf{BestCore}^{\bullet}$ as in Figure 13:

5.3 Security analysis

In previous section, the security properties of protocol $\Pi^{\mathrm{core}\star}$ have be proven under the assumption of honest majority of *effective stakes* based on $\alpha^{\star}$ and $\beta^{\star}$. Now, under the same assumption, we will show that our modified core-chain protocol $\Pi^{\mathrm{core}\bullet}$ can also achieve the same properties. Please note that our new adversary is stronger since there is not restriction on how players are registered with respect to the protocol execution. More concretely, we have the following theorem:

Theorem 5.1 (Theorem 1.5, restated)**.** *Consider core-chain protocol $\Pi^{\mathrm{core}\bullet}$ where honest players follow the 2-greedy strategy while adversarial players follow the fully-greedy strategy. If $\alpha^{\star} = \lambda\beta^{\star}$, $\lambda > 1$, then the protocol $\Pi^{\mathrm{core}\bullet}$ can achieve chain growth, chain quality, common prefix and chain soundness properties.*

5.3.1 Important lemmas

In our modified protocol $\Pi^{\mathrm{core}\bullet}$, malicious players cannot register key pairs so that they can extend the chains immediately. What the malicious players can do, however, is to register biased key

Functionality $\mathcal{F}^{\bullet}_{\mathsf{rCERT}}$

The functionality interacts with a set $\mathcal{P}$ of parties, as well as an adversary.
The functionality is parameterized by a difficulty parameter p, a security parameter κ, a stake registration bound η, as well as a tree $\mathcal{T}$. Initially, a set $\mathcal{P}_0$ of players are enabled, where $\mathcal{P}_0 \subseteq \mathcal{P}$; and for all $\mathrm{P} \in \mathcal{P}_0$ the records $(\mathrm{P}, 1)$ are stored at the root of the tree $\mathcal{T}$.

Stake Resource Registration.

1. Upon receiving a message $(\textsc{Stake-Register}, \mathrm{P}, \mathcal{C})$ from party $\mathrm{P} \in \mathcal{P}$, retrieve $\mathtt{path}$ corresponding with $\mathcal{C}$. If there is an entry $(\mathrm{P}, 1)$ with location $\mathtt{path}'$ in the tree $\mathcal{T}$, and $\mathtt{path}' \prec \mathtt{path}$, then ignore the message. Otherwise, pass the message to the adversary. Upon receiving a message $(\textsc{Stake-Registered}, \mathrm{P})$ from the adversary, set $\mathbb{b}_{\mathrm{P}} := 1$, record $(\mathrm{P}, \mathbb{b}_{\mathrm{P}})$ with location $\mathtt{path}$ in the tree $\mathcal{T}$, and pass the message to the party P. (the party P registered.)

<u>The Stake-Registered will take effect only after a block is generated on the $\mathcal{C}$.</u>

Stake Election:
For each round, set $\phi_{\mathrm{P},h^{\mathrm{prev}}} := 0$ for every registered party $\mathrm{P} \in \mathcal{P}$.
Upon receiving $(\textsc{Elect}, \mathrm{P}, \langle h^{\mathrm{prev}}, \mathtt{round}\rangle, \mathcal{C})$ from a PoS-player P, proceed as follows.
Retrieve $\mathtt{path}$ corresponding with $\mathcal{C}$ and $l = \mathrm{len}(\mathcal{C})$.
Set $\mathsf{b} := 0$. (the party P is not elected by default)

1. If $\mathsf{StakeVerification}(\mathtt{path}, \mathrm{P}, l) = 1$ and $\phi_{\mathrm{P},h^{\mathrm{prev}}} = 0$, send $(\textsc{Core-Sign}, \mathrm{P}, \langle h^{\mathrm{prev}}, \mathtt{round}\rangle)$ to the adversary.

   Upon receiving $(\textsc{Signature}, \mathrm{P}, \langle h^{\mathrm{prev}}, \mathtt{round}\rangle, \sigma)$, do:

   If $\langle h^{\mathrm{prev}}, \mathtt{round}, \mathrm{P}, \sigma, \cdot, \cdot\rangle$ has been recorded, then ignore the input. Otherwise, send a request to the adversary for a unique value h; if $\langle \cdot, \cdot, \cdot, \cdot, h, \cdot\rangle$ has been recorded, then ignore the input. Otherwise,

   – with probability p, set $\mathsf{b} := 1$ (the party P is elected), and store a record of the form $\langle h^{\mathrm{prev}}, \mathtt{round}, \mathrm{P}, \sigma, h, 1\rangle$ in memory.

Set $\phi_{\mathrm{P},h^{\mathrm{prev}}} := 1$
Output $(\textsc{Elected}, \mathrm{P}, h, \sigma, \mathsf{b})$ to P.

Block Verification:
Upon receiving $(\textsc{Core-Verify}, \mathrm{P}, \langle h^{\mathrm{prev}}, \mathtt{round}\rangle, \sigma, h, \mathcal{C}, i)$ from party $\mathrm{P}' \in \mathcal{P}$,
Retrieve $\mathtt{path}$ corresponding with $\mathcal{C}$.
Set $f := 0$

1. If $\mathsf{StakeVerification}(\mathtt{path}, \mathrm{P}, i) = 1$, send $(\textsc{Core-Verify}, \mathrm{P}, \langle h^{\mathrm{prev}}, \mathtt{round}\rangle, \sigma)$ to the adversary.

   Upon receiving $(\textsc{Core-Verified}, \mathrm{P}, \langle h^{\mathrm{prev}}, \mathtt{round}\rangle, \phi)$ from the adversary, do:

   – If $\mathtt{path}[i] = \langle h^{\mathrm{prev}}, \mathtt{round}, \mathrm{P}, \sigma, h, 1\rangle$, then set $f := 1$.
   – Else, if P is not corrupted, and no entry $\langle h^{\mathrm{prev}}, \mathtt{round}, \mathrm{P}, \sigma', h, 1\rangle$ for any σ' is recorded, then set $f := 0$ and record the entry $\langle h^{\mathrm{prev}}, \mathtt{round}, \mathrm{P}, \sigma, h, 0\rangle$.
   – Else, if there is an entry $\langle h^{\mathrm{prev}}, \mathtt{round}, \mathrm{P}, \sigma, h, f'\rangle$, then set $f := f'$.
   – Else, set $f := \phi$, and record the entry $\langle h^{\mathrm{prev}}, \mathtt{round}, \mathrm{P}, \sigma, h, f\rangle$.

2. Retrieve the registration $\mathtt{path}'$ of P.

   If $\mathtt{path}' \not\prec \mathtt{path}$ or $i - \mathrm{len}(\mathtt{path}') \leq \eta$, then set $f := 0$.

Output $(\textsc{Core-Verified}, \mathrm{P}, \langle h^{\mathrm{prev}}, \mathtt{round}\rangle, \sigma, h, f)$ to the party P'.

Figure 10: Resource certification functionality $\mathcal{F}^{\bullet}_{\mathsf{rCERT}}$.

```
Subroutine StakeVerification

This subroutine verifies if a player is valid with input $\langle \texttt{path}, \mathrm{P}, i\rangle$.
Set $f := 0$

    1. If there exists a record of the form $(\mathrm{P}, 1)$ on the path which is located on the $j$-th block where
       $j < i$,
       set $f := 1$.

Output $f$.
```

Figure 11: Stake verification subroutine StakeVerification.

pairs now, and then try to extend the chains many rounds later. We will prove that malicious players cannot obtain additional advantage by playing this strategy. First, we will show that the probability that malicious players are able to predict the latest block of the best public chain is negligible.

Lemma 5.2. *Let chain $\mathcal{C}$ be the best valid public chain with length ℓ in round r. Suppose the length of best valid public chain $\mathcal{C}'$ will be $\ell + \eta$ in round $r + t$. The probability that the malicious players predict the last block on chain $\mathcal{C}'$ in round r is $e^{-\Omega(\eta)}$ at most in round r.*

Proof. From chain quality property, we know that the honest players will contribute blocks in the last η blocks with probability no less than $1 - e^{-\Omega(\eta)}$. Blocks generated by honest players are unpredictable for malicious players. We have that the malicious players cannot predict any block from honest players before it is published. Furthermore, he cannot predict the last block of $\mathcal{C}'$ in round r if there is a honest block on chain $\mathcal{C}'$ at last. We get the conclusion that the malicious players predict the last block of chain $\mathcal{C}'$ in round r is at most $e^{-\Omega(\eta)}$. □

If malicious players cannot predict the last block the best chain, then he cannot have a biased key pair registered so that the corresponding stakeholder can be chosen in a future round with much higher probability. From Lemma 5.2, we conclude that a malicious player, by playing adaptive key registration strategy, cannot improve the probability that he is chosen for extending the public chain. Next, we will show that the malicious players cannot gain advantage, by playing this adaptive strategy, for extending a private (hidden) chain.

Lemma 5.3. *Assume the malicious players fork a hidden chain $\mathcal{C}$ from a block during t rounds. Let $\ell_{hidden} < \eta$ be the length of the forked hidden chain $\mathcal{C}$. Assume the length of public best chain increase ℓ_{public} blocks during this t rounds. We have $\Pr[\ell_{public} > \ell_{hidden}] > 1 - e^{-\Omega(\ell_{hidden})}$.*

Proof. From the modified protocol, we also know that the adaptive key generation will not affect the first η blocks. It is said that the adaptive strategy will not help the malicious players to shorten the rounds for the first η blocks. The probability that $\ell_{hidden} > \ell_{public}$ equals to the probability that the malicious players break common prefix property. From Corollary 4.9, we have $\Pr[\ell_{public} > \ell_{hidden}] > 1 - e^{-\Omega(\ell_{hidden})}$. □

From the modified protocol, we know that if there are two divergent chains, the honest players will compare the first η different blocks. Suppose η is large enough. From Lemma 5.3, we know that the hidden chain will not be accepted with overwhelming probability, even the adversary

PROTOCOL $\Pi^{\text{core}\bullet}$

Initially, a set $\mathcal{P}_0$ of players are registered to the functionality $\mathcal{F}^{\bullet}_{\mathsf{rCERT}}$.

Upon receiving message (REGISTER-STAKE, P) from the environment $\mathcal{Z}$ at round $\mathtt{round}$,
Let $\mathbb{C}$ be the set of core-chains collected from $\mathcal{F}_{\mathsf{NET}}$.
For each $\mathcal{C} \in \mathbb{C}$, send (STAKE-REGISTER, P, $\mathcal{C}$) to functionality $\mathcal{F}^{\bullet}_{\mathsf{rCERT}}$.

Upon receiving message (INPUT-STAKE, P) from the environment $\mathcal{Z}$ at round $\mathtt{round}$, the PoS-player $\mathrm{P} \in \mathcal{P}$, proceeds as follows.

1. *Select the best local PoS core-chain:*

   Let $\mathbb{C}$ be the set of core-chains collected from $\mathcal{F}_{\mathsf{NET}}$.

   Compute $\mathbb{C}_{\text{best}} := \mathsf{BestCore}^{\bullet}(\mathbb{C}, \mathtt{round})$.

2. *Attempt to extend PoS core-chain:*

   For each $\mathcal{C} \in \mathbb{C}_{\text{best}}$, and $\ell := \text{len}(\mathcal{C})$: (BestCore$^{\bullet}$ will return a best chain set and the players will try to extend all)

   Parse $\mathcal{C}[\ell]$ as $\langle\langle h^{\text{prev}}_{\ell}, \mathtt{round}_{\ell}, \mathrm{P}_{\ell}, \sigma_{\ell}\rangle, h_{\ell}\rangle$.

   Set $h^{\text{prev}} := h_{\ell}$.

   - *Stake election:* Send (ELECT, P, $\langle h_{\ell}, \mathtt{round}\rangle$, $\mathcal{C}$) to functionality $\mathcal{F}^{\bullet}_{\mathsf{rCERT}}$, and receive (ELECTED, P, $h_{\ell+1}$, σ, b) from $\mathcal{F}^{\bullet}_{\mathsf{rCERT}}$.
   - *If* $\mathsf{b} = 1$, *generate a new block-core:* Set the new block-core $B := \langle\langle h_{\ell}, \mathtt{round}, \mathrm{P}, \sigma\rangle, h_{\ell+1}\rangle$, and set $\mathcal{C} := \mathcal{C} \| B$, and $state := state \cup \{\mathcal{C}\}$, and then send (BROADCAST, $\mathcal{C}$) to $\mathcal{F}_{\mathsf{NET}}$.

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

Figure 12: Our proof-of-stake core-chain protocol $\Pi^{\text{core}\bullet}$ in the $\{\mathcal{F}^{\bullet}_{\mathsf{rCERT}}, \mathcal{F}_{\mathsf{NET}}\}$-hybrid model. (See Figure 13 for the subroutine BestCore$^{\bullet}$.)

follows the adaptive key registration strategy. That means, the adaptive key registration strategy is not helpful for extending the hidden chain.

5.3.2 Security properties in the presence of an adaptive adversary

Chain growth. Honest players in protocol $\Pi^{\text{core}\bullet}$ will extend the chains in the same way as that in protocol $\Pi^{\text{core}\star}$. From Corollary 4.7 we have:

Corollary 5.4 (Chain growth). *Consider core-chain protocol $\Pi^{\text{core}\bullet}$ in the presence of a fully-greedy and adaptive adversary. Consider an honest PoS-player P' with best local PoS core-chain $\mathcal{C}'$ in round r', and an honest PoS-player P'' with best local core-chain $\mathcal{C}''$ in round r'', where $r'' > r'$. Then we have $\Pr\left[\text{len}(\mathcal{C}'') - \text{len}(\mathcal{C}') \geq g \cdot t\right] \geq 1 - e^{-\Omega(t)}$, where $t = r'' - r'$, $g = (1-\delta)\alpha^{\star}$, and $\delta > 0$.*

Chain quality. From Lemma 5.3 and 5.2, the adversary cannot obtain additional advantage by playing the adaptive strategy. That is, they cannot produce more blocks by adaptively selecting key-pairs and having their stakes registrated. From Corollary 4.8 we have:

Corollary 5.5 (Chain quality). *Consider $\alpha^{\star} = \lambda\beta^{\star}$, $\lambda > 1$, and $\delta > 0$. Consider core-chain protocol $\Pi^{\text{core}\bullet}$ with a greedy and adaptive adversary. Consider an honest PoS-player with PoS core-chain $\mathcal{C}$. Consider that ℓ consecutive block-cores of $\mathcal{C}$, where ℓ_{good} block-cores are generated by honest PoS-*

```
SUBROUTINE BestCore$^\bullet$

The subroutine BestCore$^\bullet$ is allowed to access to the functionality $\mathcal{F}^\bullet_{\mathsf{rCERT}}$, and with input ($\mathbb{C}'$, round$'$).
For every chain $\mathcal{C} \in \mathbb{C}'$, and proceed as follows.

  1. Set $\ell := \mathrm{len}(\mathcal{C})$.
  2. For $i$ from $\ell$ down to 1, verify block-core $\mathcal{C}[i]$, as follows.
       • Parse $\mathcal{C}[i]$ as $\langle\langle h_i^{\mathrm{prev}}, \mathtt{round}_i, \mathrm{P}_i, \sigma_i\rangle, h_i\rangle$.
         Parse $\mathcal{C}[i-1]$ as $\langle\langle h_{i-1}^{\mathrm{prev}}, \mathtt{round}_{i-1}, \mathrm{P}_{i-1}, \sigma_{i-1}\rangle, h_{i-1}\rangle$.
       • If $\mathtt{round}_i < \mathtt{round}'$ and $\mathtt{round}_{i-1} < \mathtt{round}_i$ , then execute:
           – If $h_i^{\mathrm{prev}} \neq h_{i-1}$, then remove this core-chain $\mathcal{C}$ from $\mathbb{C}'$.
           – Else send (CORE-VERIFY, $\mathrm{P}_i$, $\langle h_i^{\mathrm{prev}}, \mathtt{round}_i\rangle, \sigma_i, h_i, \mathcal{C}, i$) to $\mathcal{F}^\bullet_{\mathsf{rCERT}}$.
             Upon receiving message (CORE-VERIFIED, $\mathrm{P}_i$, $\langle h_i^{\mathrm{prev}}, \mathtt{round}_i\rangle, \sigma_i, h_i, f_i$) from $\mathcal{F}^\bullet_{\mathsf{rCERT}}$, if $f_i = 0$ remove this core-chain $\mathcal{C}$ from $\mathbb{C}'$.
         Otherwise, remove the core-chain $\mathcal{C}$ from $\mathbb{C}'$.

Let $\mathcal{C}_{\mathrm{best}} := \emptyset$
For every chain $\mathcal{C} \in \mathbb{C}'$ do
If $\mathcal{C}_{\mathrm{best}} = \emptyset$ then $\mathcal{C}_{\mathrm{best}} := \mathcal{C}$
Let $\ell$ be the divergent length of $\mathcal{C}_{\mathrm{best}}$ with $\mathcal{C}$, $\ell'$ be the divergent length of $\mathcal{C}$ with $\mathcal{C}_{\mathrm{best}}$.

  • If $\ell < \eta$ and $\ell' < \eta$
      – If $\ell < \ell'$ then $\mathcal{C}_{\mathrm{best}} := \mathcal{C}$
  • If $\ell \geq \eta$ or $\ell' \geq \eta$
    Let $t'$ be the round number for the $\eta$-th divergent block on $\mathcal{C}_{\mathrm{best}}$. Set $t' := \infty$ if $\ell < \eta$
    Let $t$ be the round number for the $\eta$-th divergent block on $\mathcal{C}$. Set $t := \infty$ if $\ell' < \eta$
      – If $t < t'$ then $\mathcal{C}_{\mathrm{best}} := \mathcal{C}$.

Let $\ell := \mathrm{len}(\mathcal{C}_{\mathrm{best}})$. Set $\mathbb{C}_{\mathrm{best}} := \emptyset$.
For any chain $\mathcal{C} \in \mathbb{C}'$ and $\mathrm{len}(\mathcal{C}) \geq \ell - \mathrm{g}$, let $\ell'$ be the divergent length of $\mathcal{C}$ with $\mathcal{C}_{\mathrm{best}}$ ,
if $\ell' < \eta$ then $\mathbb{C}_{\mathrm{best}} := \mathbb{C}_{\mathrm{best}} \cup \{\mathcal{C}\}$. (g is a small constant, g = 2 typically.)
Then return $\mathbb{C}_{\mathrm{best}}$ as the output.
```

Figure 13: The core-chain set validation subroutine BestCore$^\bullet$.

players. Then we have $\Pr\left[\frac{\ell_{good}}{\ell} \geq \mu\right] \geq 1 - e^{-\Omega(\ell)}$, *where* $\mu = 1 - (1+\delta)\frac{1}{\lambda}$.

Common prefix. Again, from Lemma 5.3 and Lemma 5.2, the adversary cannot obtain extra benefit by playing the adaptive strategy. They cannot produce more blocks by adaptively selecting key pairs. From Corollary 4.9 we have:

Corollary 5.6 (Common prefix)**.** *Consider $\alpha^\star = \lambda\beta^\star$, $\lambda > 1$, and $\delta > 0$. Consider core-chain protocol $\Pi^{\text{core}\bullet}$ with a fully-greedy and adaptive adversary. Consider two honest PoS-players, P in round r and P' in round r', with the local best PoS core-chains $\mathcal{C}, \mathcal{C}'$, respectively, where $r' \geq r$. Then we have $\Pr\left[\mathcal{C}[1,\ell] \preceq \mathcal{C}'\right] \geq 1 - e^{-\Omega(\kappa)}$, where $\ell = \text{len}(\mathcal{C}) - \Theta(\kappa)$.*

Chain soundness. As in Corollary 4.10, we can achieve the chain soundness property because we use *the longest chain is the best chain* policy. Otherwise, if a new player accepts a chain which is not the best public chain, this chain will also be accepted by existing players. The common prefix property will be violated.

Corollary 5.7 (Chain soundness)**.** *Consider for every round, $\alpha^\star = \lambda\beta^\star$, $\lambda > 1$, and $\delta > 0$. Consider core-chain protocol $\Pi^{\text{core}\bullet}$ with a fully-greedy and adaptive adversary. Consider two honest PoS-players, P' and P'' in round r, with the local best core-chains $\mathcal{C}'$ and $\mathcal{C}''$, respectively, where P' is a new player and P'' is an existing player in round r. Then we have $\mathcal{C}'[\neg\kappa] \preceq \mathcal{C}''$ and $\mathcal{C}''[\neg\kappa] \preceq \mathcal{C}'$.*

6 From core-chain to blockchain

In this section, we start to extend the core-chain protocol in Section 5 to a blockchain protocol in which payload (transactions) will be included. We want to emphasize that the payload cannot be included into the core block directly. If the payload is in the core block, the malicious players may try to brute-force different payloads to get the solution that satisfies the hash inequality. Furthermore, the scheme must guarantee that a malicious player can not change the payload he signed before. In a concurrent work, Ouroboros Praos [21] requires honest users are able to erase the signing key after it is used to sign a block. This is a strong assumption and it may be too complicated to implement. On contrary, our construction here requires nothing more other than a regular signature.

Intuitively, the core-chain can be viewed as a (biased) random beacon; we can use the beacon to select a PoS-player to generate a new block with payload. The block with payload will also be linked as a hash chain which is called main blockchain. More concretely, once a new block-core B_{i+1} is generated by a PoS-player (in the blockchain protocol), then the PoS-player is selected to generate the new block $\tilde{B}_{i+1}$, in the following format $\tilde{B}_{i+1} = \langle \mathsf{hash}(\tilde{B}_i), B_{i+1}, \tilde{X}_{i+1}, \tilde{\text{PK}}, \tilde{\sigma} \rangle$ where $\tilde{\sigma} \leftarrow \mathsf{Sign}_{\tilde{\text{SK}}}(\tilde{h}_i, B_{i+1}, \tilde{X}_i)$ and $\tilde{h}_i := \mathsf{hash}(\tilde{B}_i)$, and $B_{i+1} := \langle h_i, \texttt{round}, \text{PK}, \sigma \rangle$. $\tilde{X}_{i+1}$ is payload. Here we note that in our blockchain protocol design, the PoS-player holds two combined pairs of keys, (SK, PK) of the strengthened unique signature scheme $(\mathsf{uKeyGen}, \mathsf{uSign}, \mathsf{uVerify})$, and $(\tilde{\text{SK}}, \tilde{\text{PK}})$ of a regular digital signature scheme $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$. Now the blocks in the main blockchain are "glued" with the block-cores in the blockchain, and we can reduce the security of the blockchain protocol to the security of the blockchain protocol.

In the formal description of our blockchain protocol below, we will use a slightly augmented setup functionality $\tilde{\mathcal{F}}^\bullet_{\mathsf{rCERT}}$ to capture the hash inequality and the block and block-core signing/verification. Similarly, this setup functionality can be implemented by using hash function $\mathsf{H}(\cdot)$ and a digital signature scheme.

6.1 Setup functionality $\tilde{\mathcal{F}}^{\bullet}_{\mathsf{rCERT}}$

In our blockchain protocol design, we will use the setup functionality, $\tilde{\mathcal{F}}^{\bullet}_{\mathsf{rCERT}}$ (in Figure 14), which is an augmented version of the resource certification functionality in Section 5.1. The first two phases, "Stake Resource Registration" and "Stake Election", remain the same. A new phase, "Signature Generation", is defined to be utilized to sign a main block. And "Block Verification" is extended to verify main block.

FUNCTIONALITY $\tilde{\mathcal{F}}^{\bullet}_{\mathsf{rCERT}}$

The functionality performs same functions with $\mathcal{F}^{\bullet}_{\mathsf{rCERT}}$ when receiving commands for **Stake Resource Registration**, **Stake Election**, and **Block Verification**.

In addition:

Signature Generation:

Upon receiving $(\textsc{Block-Sign}, \mathrm{P}, msg)$ from a party P, send $(\textsc{Block-Sign}, \mathrm{P}, msg)$ to the adversary.
Upon receiving $(\textsc{Signature}, \mathrm{P}, msg, \tilde{\sigma})$ from the adversary, verify that no entry $(\mathrm{P}, msg, \tilde{\sigma}, 0)$ is recorded. If it is, then output an error message $(\textsc{Error})$ to P and halt. Else, output $(\textsc{Block-Signed}, \mathrm{P}, msg, \tilde{\sigma})$ to P, and record the entry $(\mathrm{P}, msg, \tilde{\sigma}, 1)$.

Block Verification:

Upon receiving $(\textsc{Block-Verify}, \mathrm{P}, msg, \tilde{\sigma})$ from a party $\mathrm{P}' \in \mathcal{P}$,

1. If there exists a record of the form $(\mathrm{P}, \cdot)$, (the party P is elected) then send $(\textsc{Block-Verify}, \mathrm{P}, msg, \tilde{\sigma})$ to the adversary. Upon receiving $(\textsc{Block-Verified}, \mathrm{P}, msg, \tilde{\phi})$ from the adversary, do:
   - If $(\mathrm{P}, msg, \tilde{\sigma}, 1)$ is recorded, then set $f := 1$.
   - Else, if P is not corrupted, and no entry $(\mathrm{P}, msg, \tilde{\sigma}', 1)$ for any $\tilde{\sigma}'$ is recorded, then set $f := 0$ and record the entry $(\mathrm{P}, msg, \tilde{\sigma}, f)$.
   - Else, if there is an entry $(\mathrm{P}, msg, \tilde{\sigma}, f')$, then set $f := f'$.
   - Else, set $f := \tilde{\phi}$, and record the entry $(\mathrm{P}, msg, \tilde{\sigma}, f)$.

   Output $(\textsc{Block-Verified}, \mathrm{P}, msg, f)$ to the party P'.

2. Otherwise, if there is no record of the form $(\mathrm{P}, \cdot)$, (the party P is not elected)

   set $f := 0$ and output $(\textsc{Block-Verified}, \mathrm{P}, msg, f)$ to the party P'.

Figure 14: Augmented resource certification functionality $\tilde{\mathcal{F}}^{\bullet}_{\mathsf{rCERT}}$.

6.2 Main blockchain protocol

We now describe our PoS based blockchain protocol Π^{main}. The blockchain protocol can be viewed as an augmented version of the core-chain protocol in Section 5.2, and now it uses the augmented resource certificate functionality $\tilde{\mathcal{F}}^{\bullet}_{\mathsf{rCERT}}$ as setup functionality. The functionality $\tilde{\mathcal{F}}^{\bullet}_{\mathsf{rCERT}}$ has the same phases as of $\mathcal{F}^{\bullet}_{\mathsf{rCERT}}$. The difference is $\tilde{\mathcal{F}}^{\bullet}_{\mathsf{rCERT}}$ will provide extra service for main block signature and verification.

As in the core-chain protocol, for each PoS-player P, once activated by the environment on $(\textsc{Input-Stake}, \mathrm{P})$ at a round, and received a blockchain set $\tilde{\mathbb{C}}$ from $\mathcal{F}_{\mathsf{NET}}$, the party P finds the best valid blockchain $\tilde{\mathcal{C}}_{\text{best}}$ by running the subroutine $\mathsf{BestMain}$ (in Figure 16), and then updates its local blockchain $\tilde{\mathcal{C}} := \tilde{\mathcal{C}}_{\text{best}}$. Note that, the i-th block in blockchain $\tilde{\mathcal{C}}$, is in the following format $\tilde{B}_i := \langle\langle \tilde{h}_{i-1}, B_i, \tilde{X}_i\rangle, \mathrm{P}_i, \tilde{\sigma}_i\rangle$. That means, from $\tilde{B}_i$, we can obtain the i-th block-core

B_i. We thus can derive the core-chain $\mathcal{C}$ from the blockchain $\tilde{\mathcal{C}}$. If the PoS-player P is selected (with certain probability p), he can query the functionality to generate a signature σ for $context := \langle h_\ell, \mathtt{round}_{\ell+1}\rangle$. Then he defines a new block-core $B_{\ell+1} := \langle\langle h_\ell, \mathtt{round}_\ell\rangle, \mathrm{P}, \sigma\rangle$, updates his local core-chain $\mathcal{C}$. Once the new block-core $B_{\ell+1}$ is generated, the PoS-player P can query the functionality to generate a signature $\tilde{\sigma}$ for $msg := \langle \tilde{h}_\ell, B_{\ell+1}, \tilde{X}_{\ell+1}\rangle$. Then he can define a new block $\tilde{B}_{\ell+1} := \langle\langle \tilde{h}_\ell, B_{\ell+1}, \tilde{X}_{\ell+1}\rangle, \mathrm{P}, \tilde{\sigma}\rangle$, and update his local blockchain $\tilde{\mathcal{C}}$. He then broadcasts the local blockchain to the network. Please refer to Figure 15 for more details of our blockchain protocol. In the verification phase, the functionality $\tilde{\mathcal{F}}^{\bullet}_{\mathsf{rCERT}}$ will verify if the main-chain is valid after the core-chain is verified to ensure the payload is correct.

Protocol Π^{main}

Π^{main} performs same as $\Pi^{\mathrm{core}\bullet}$ by calling subroutine $\mathsf{BestMain}$ and $\tilde{\mathcal{F}}^{\bullet}_{\mathsf{rCERT}}$ instead of $\mathsf{BestCore}^{\bullet}$ and $\mathcal{F}^{\bullet}_{\mathsf{rCERT}}$.
In addition, in *Stake election* :
If the player is elected to generate a block after generating a core-block he will:

- *Attempt to extend PoS blockchain:*
  - *Generate a new block:*
    If the new block-core B is obtained, then compute $\tilde{h}_\ell := \mathsf{hash}(\tilde{\mathcal{C}}[\ell])$, send $(\textsc{Sign}, \mathrm{P}, \langle \tilde{h}_\ell, B, \tilde{X}\rangle)$ to functionality $\tilde{\mathcal{F}}^{\bullet}_{\mathsf{rCERT}}$, and receive $(\textsc{Signed}, \mathrm{P}, \langle \tilde{h}_\ell, B, \tilde{X}\rangle, \tilde{\sigma})$ from $\tilde{\mathcal{F}}^{\bullet}_{\mathsf{rCERT}}$.
    set the new block $\tilde{B} := \langle\langle \tilde{h}_\ell, B, \tilde{X}\rangle, \mathrm{P}, \tilde{\sigma}\rangle$, and set $\tilde{\mathcal{C}} := \tilde{\mathcal{C}} \| \tilde{B}$, and $state := state \cup \{\tilde{\mathcal{C}}\}$, and then send $(\textsc{Broadcast}, \tilde{\mathcal{C}})$ to $\mathcal{F}_{\mathsf{NET}}$.

Figure 15: Our proof-of-stake blockchain protocol Π^{main} in the $\{\tilde{\mathcal{F}}^{\bullet}_{\mathsf{rCERT}}, \mathcal{F}_{\mathsf{NET}}\}$-hybrid model. (See Figure 16 for the subroutine $\mathsf{BestMain}$.)

In Section 5, our proof-of-stake core-chain protocol $\Pi^{\mathrm{core}\bullet}$ uses the subroutine $\mathsf{BestCore}^{\bullet}$ to single out the best valid core-chain from a set of core-chains. Here we describe a similar strategy, subroutine $\mathsf{BestMain}$, to single out the best blockchain from a set of blockchains. The subroutine $\mathsf{BestMain}$ here is a slightly augmented version of the subroutine $\mathsf{BestCore}^{\bullet}$ in our core-chain protocol. Intuitively, a blockchain is the best one if it is the *current longest valid* blockchain. The $\mathsf{BestMain}$ subroutine takes as input, a blockchain set $\tilde{\mathbb{C}}'$ and the current time information $\mathtt{round}'$. Intuitively, the subroutine validates all $\tilde{\mathcal{C}} \in \tilde{\mathbb{C}}'$, then finds the valid longest blockchain.

In more detail, $\mathsf{BestMain}$ proceeds as follows. On input the current set of blockchains $\tilde{\mathbb{C}}'$ and the current time information $\mathtt{round}$, and for each blockchain $\tilde{\mathcal{C}}$, the subroutine first unfolds the blockchain $\tilde{\mathcal{C}}$ into the corresponding core-chain $\mathcal{C}$; the subroutine then evaluates every block-core of the core-chain $\mathcal{C}$, and then every block of the blockchain $\tilde{\mathcal{C}}$, sequentially. Let ℓ be the length of $\tilde{\mathcal{C}}$. (ℓ is also the length of the corresponding core-chain $\mathcal{C}$.) Starting from the head of $\mathcal{C}$, for every block-core $\mathcal{C}[i]$, for all $i \in [\ell]$, in the core-chain $\mathcal{C}$, the $\mathsf{BestMain}$ subroutine (1) ensures that $\mathcal{C}[i]$ is linked to the previous block-core $\mathcal{C}[i-1]$ correctly, and (2) tests if the signature generated by that PoS-player can be verified (by interacting with $\tilde{\mathcal{F}}^{\bullet}_{\mathsf{rCERT}}$). Then for every block-core $\tilde{\mathcal{C}}[i]$, for all $i \in [\ell]$, in the blockchain $\tilde{\mathcal{C}}$, the $\mathsf{BestMain}$ subroutine (1) ensures that $\tilde{\mathcal{C}}[i]$ is linked to the previous block $\tilde{\mathcal{C}}[i-1]$ correctly, and (2) tests if the signature generated by that PoS-player can be verified (by interacting with $\tilde{\mathcal{F}}^{\bullet}_{\mathsf{rCERT}}$). After the validation, the best valid blockchain set is selected. Please refer to Figure 16 for more details.

Subroutine BestMain

The subroutine BestMain is allowed to access to the functionality $\tilde{\mathcal{F}}^{\bullet}_{\mathsf{rCERT}}$, and with input $(\tilde{\mathbb{C}}', \mathtt{round})$.
For every chain $\tilde{\mathcal{C}} \in \tilde{\mathbb{C}}'$, and proceed as follows.

1. Set $\ell := \mathrm{len}(\tilde{\mathcal{C}})$. Derive $\mathcal{C}$ from $\tilde{\mathcal{C}}$.
2. Perform same as in BestCore$^{\bullet}$.

   In addition verify each block on $\tilde{\mathcal{C}}$ by query $(\textsc{Block-Verify}, \mathrm{P}_i, \langle \tilde{h}_{i-1}, B_i, \tilde{X}_i \rangle, \tilde{\sigma}_i)$ to $\tilde{\mathcal{F}}^{\bullet}_{\mathsf{rCERT}}$.

Perform same best chain strategy with $\tilde{\mathcal{C}}$ as in BestCore$^{\bullet}$.

Figure 16: The chain set validation subroutine BestMain.

6.3 Analysis of blockchain protocol

As mentioned before, our blockchain protocol Π^{main} can be viewed as an augmented version of the core-chain protocol $\Pi^{\mathrm{core}\bullet}$ in Section 5; each security property of our blockchain protocol can be reduced to the corresponding property of the core-chain protocol. We note that, as in the core-chain protocol $\Pi^{\mathrm{core}\bullet}$, the security properties hold under the assumption of honest majority of effective stakes based on $\alpha^\star$ and $\beta^\star$.

Theorem 6.1. *Consider blockchain protocol Π^{main} where honest players follow the 2-greedy strategy while adversarial players follow the fully-greedy strategy. If $\alpha^\star = \lambda\beta^\star$, $\lambda > 1$, then the protocol Π^{main} can achieve chain growth, chain quality, common prefix and chain soundness properties.*

Chain growth. Chain growth property comes from that of the core-chain protocol. If a PoS-player is chosen to generate a block-core in the core-chain, the PoS-player is also be chosen to generate the corresponding block in the blockchain. That means, when the core-chain is extended with a new block-core, the corresponding blockchain is also extended with a new block. More formally, we have the following statement.

Corollary 6.2 (Chain growth). *Consider the blockchain protocol Π^{main}. Consider $\alpha^\star = \lambda\beta^\star$, $\lambda > 1$, and $\delta > 0$. Consider an honest PoS-player with the best PoS blockchain $\tilde{\mathcal{C}}$ in round r, and local PoS blockchain $\tilde{\mathcal{C}}'$ in round r', where $r' > r$. Then we have $\Pr\left[\mathrm{len}(\tilde{\mathcal{C}}') - \mathrm{len}(\tilde{\mathcal{C}}) \geq g \cdot t\right] \geq 1 - e^{-\Omega(t)}$, where $t = r' - r$, $g = (1-\delta)\alpha^\star$.*

Proof. From the protocol, we know that every PoS blockchain $\tilde{\mathcal{C}}$ is associated with a PoS core-chain $\mathcal{C}$. Each valid block-core B has a corresponding block $\tilde{B}$. We have, $\mathrm{len}(\tilde{\mathcal{C}}') = \mathrm{len}(\mathcal{C}')$ and $\mathrm{len}(\tilde{\mathcal{C}}) = \mathrm{len}(\mathcal{C})$. That means, $\mathrm{len}(\tilde{\mathcal{C}}') - \mathrm{len}(\tilde{\mathcal{C}}) = \mathrm{len}(\mathcal{C}') - \mathrm{len}(\mathcal{C})$. From Corollary 5.4, we have $\Pr\left[\mathrm{len}(\tilde{\mathcal{C}}') - \mathrm{len}(\tilde{\mathcal{C}}) \geq g \cdot t\right] = \Pr\left[\mathrm{len}(\mathcal{C}') - \mathrm{len}(\mathcal{C}) \geq g \cdot t\right] \geq 1 - e^{-\Omega(t)}$. This completes the proof. □

Chain quality. Similarly, if an honest player contributes a block-core to the core-chain, he also contributes a block to the blockchain. More formally, we have the following statement.

Corollary 6.3 (Chain quality)**.** *Consider the blockchain protocol Π^{main}. Consider $\alpha^\star = \lambda\beta^\star$, $\lambda > 1$, and $\delta > 0$. Consider an honest PoS-player with the best PoS blockchain $\tilde{\mathcal{C}}$. Consider any ℓ consecutive blocks on $\tilde{\mathcal{C}}$, including ℓ_{good} blocks are generated by honest PoS-players. Then we have $\Pr[\frac{\ell_{good}}{\ell} \geq \mu] \geq 1 - e^{-\Omega(\ell)}$ where $\mu = 1 - (1+\delta)\frac{1}{\lambda}$.*

Proof. From the algorithms, we know that every PoS blockchain $\tilde{\mathcal{C}}$ is associated with a PoS core-chain $\mathcal{C}$. Let $\ell^{\text{core}}_{good}$ be the number of block-cores from honest stakeholders on core-chain $\mathcal{C}$. Let $\ell^{\text{main}}_{good}$ be the number of blocks from honest stakeholders on blockchain $\tilde{\mathcal{C}}$. Recall that both block-core $\mathcal{C}[i]$ and the corresponding block $\tilde{\mathcal{C}}[i]$ are signed by the same stakeholder. We have $\ell^{\text{core}}_{good} = \ell^{\text{main}}_{good} = \ell_{good}$. We also have that $\text{len}(\mathcal{C}) = \text{len}(\tilde{\mathcal{C}}) = \ell$. From Corollary 5.5 we have $\Pr[\frac{\ell_{good}}{\ell} \geq \mu] \geq 1 - e^{-\Omega(\ell)}$, where $\mu = 1 - (1+\delta)\frac{1}{\lambda}$. □

Common prefix. Our analysis is based on the common prefix analysis of core-chain. The core-chain can achieve common prefix as we discussed. The opportunity for malicious players to destroy common prefix probability is to generate different blockchain for the same core-chain. For the malicious players can sign different blocks for one block-core, this will allow him to fork the blockchain. So the malicious players can fork the blockchain when they are chosen to generate block. However, with the property of hash function, the malicious players can not generate two blocks with same hash value. When an honest player is chosen to extend a block, he will only support one blockchain. Then all of the honest players will converge on one blockchain.

Corollary 6.4 (Common prefix)**.** *Consider the blockchain protocol Π^{main}. Consider $\alpha^\star = \lambda\beta^\star$, $\lambda > 1$, and $\delta > 0$. Consider two honest PoS-players, P in round r and P' in round r', with the local best PoS blockchains $\tilde{\mathcal{C}}$, $\tilde{\mathcal{C}}'$, respectively, where $r' \geq r$. Then we have $\Pr[\tilde{\mathcal{C}}[1,\ell] \preceq \tilde{\mathcal{C}}'] \geq 1 - e^{-\Omega(\kappa)}$, where $\ell = \text{len}(\mathcal{C}) - \Theta(\kappa)$.*

Proof. As we discussed, $\tilde{\mathcal{C}}$ and $\tilde{\mathcal{C}}'$ are associated with core-chains $\mathcal{C}$ and $\mathcal{C}'$ respectively. From Corollary 5.6 we know that $\Pr[\mathcal{C}[1,\ell] \preceq \mathcal{C}'] \geq 1 - e^{-\Omega(\kappa)}$.

Based on the assumption that $\alpha^\star = \lambda\beta^\star$ and $\lambda > 1$, we can have that the malicious players are not able to generate more than $\Theta(\kappa)$ blocks before an honest player is chosen to generate block with high probability. All of the honest players will converge on the same chain. Put them together, we have $\Pr[\tilde{\mathcal{C}}[1,\ell] \preceq \tilde{\mathcal{C}}'] \geq 1 - e^{-\Omega(\kappa)}$ where $\ell = \text{len}(\mathcal{C}) - \Theta(\kappa)$. □

Chain soundness. A new player will accept a blockchain (in which the corresponding core-chain is included). The proof idea for achieving chain soundness property of our blockchain protocol directly follows that for the core-chain protocol. We have the following statement.

Corollary 6.5 (Chain soundness)**.** *Consider the blockchain protocol Π^{main}. Consider for every round, $\alpha = \lambda\beta$, $\lambda > 1$, and $\delta > 0$. There are two honest PoS-players, P' and P'' in round r, with the local best PoS blockchains $\tilde{\mathcal{C}}'$ and $\tilde{\mathcal{C}}''$, respectively. Let P' be a new player and P'' be an existing player in round r. Then we have $\tilde{\mathcal{C}}'[\neg\kappa] \preceq \tilde{\mathcal{C}}''$ and $\tilde{\mathcal{C}}''[\neg\kappa] \preceq \tilde{\mathcal{C}}'$.*

Proof. Blockchains $\tilde{\mathcal{C}}'$ and $\tilde{\mathcal{C}}''$ are associated with core-chains $\mathcal{C}'$ and $\mathcal{C}''$ respectively. From Lemma 3.5 we know that $\mathcal{C}'[\neg\kappa] \preceq \mathcal{C}''$ and $\mathcal{C}''[\neg\kappa] \preceq \mathcal{C}'$. We immediately have $\tilde{\mathcal{C}}'[\neg\kappa] \preceq \tilde{\mathcal{C}}''$ and $\tilde{\mathcal{C}}''[\neg\kappa] \preceq \tilde{\mathcal{C}}'$. □

7 Extensions and Discussions

Our design is a natural mimic of Nakamoto's but via proof-of-stake. We can easily "borrow" many ideas in Nakamoto's white paper (and in follow-up papers), to our design. Our design can tolerate well-known rational attacks. In this section, we discuss a few of them.

Blockchain with adaptive difficulty adjustment. In Bitcoin, in order to maintain a steady chain growth rate, the system adjusts the PoW hash target difficulty adaptively. The smaller target, the lower probability to get a valid PoW block by a hash function query, and vice versa. Our scheme can be extended to support adaptive difficulty easily. As in Nakamoto's system, the target difficulty is adjusted every m blocks for some integer m. The time span of difficulty adjustment is called an *epoch*; and let t be the expected time of an epoch. Let t_i be the actual time span of the i-th epoch, and $\mathtt{T}_i$ be the target difficulty in the i-th epoch. We have the target difficulty in the $(i+1)$-th epoch as follows:

$$\mathtt{T}_{i+1} = \frac{t_i}{\mathsf{t}}\mathtt{T}_i$$

From the equation above we can observe that, if $t_i > \mathsf{t}$ then $\mathtt{T}_{i+1} > \mathtt{T}_i$ and vice-versa. In the case that $t_i > \mathsf{t}$, the stakeholders spend longer time to obtain m blocks; it means the system requires more time than expected for the i-th epoch; thus, the target difficulty should be increased so that the stakeholders can find new blocks faster in the next epoch. This *negative feedback* mechanism makes the system stable. To extend a PoS blockchain, we modify the hash inequality as $\mathsf{H}(\mathsf{hash}(B_i), \mathtt{round}, \textsc{pk}, \sigma) < \mathtt{T}_i$. A player will test if he is qualified to sign a PoS-block based on the current target difficulty $\mathtt{T}_i$.

Blockchain in the non-flat model. Our ideas in previous sections are described in the "flat" model, where all PoS-players are assumed to hold the same number of stakes (and they are selected as the winning player with the same probability in each round). In reality, PoS-players have different amounts of stake. We next discuss how to extend our design ideas properly into this more realistic "non-flat" model. Consider a PoS-player, with verification-signing key pair $(\textsc{pk}, \textsc{sk})$, holding v number of stakes. Let $\mathtt{T}_j$ denote the target difficulty in the current epoch, i.e., the j-th epoch. We change the hash inequality as follows:

$$\mathsf{H}(\mathsf{hash}(B_i), \mathtt{round}, \textsc{pk}, \sigma) < v\mathtt{T}_i$$

Now we argue that the winning probability of a PoS-player for generating a new block-core is proportional to the amount of stake he controls. We assume the total amount of stakes in the whole system is n; consider hash function $\mathsf{H} : \{0,1\}^* \mapsto \{0,1\}^\kappa$. We assume $np \ll 1$, where $p = \frac{\mathtt{T}_i}{2^\kappa}$. Now the PoS-player can play different strategies. If the PoS-player puts his v coins in one account, the probability that he is selected to sign a PoS block is vp. If the PoS-player puts his v coins in v accounts and every account has one stake, the probability that an account is selected to sign a PoS block is p. The outputs of hash function are independent for different verification keys. The total probability that the PoS-player is selected is $1-(1-p)^v \approx vp$. That is, the probability a stakeholder is selected in the non-flat model is (approximately) equal to the accumulated probability that he distributes the stakes to different accounts as in the flat-model. For a PoS-player, the probability that he is selected only depends on the total amount of stakes he controls.

Nothing at stake. Nothing at stake is a rational attack against the PoS blockchain, where the optimal strategy for any rational player is to validate every chain, i.e. greedily, so that the player gets his reward no matter which fork wins.

Our design can effectively defeat nothing at stake attack. As shown in Section 4, in the modified core-chain protocol $\Pi^{\text{core}\star}$, players follows the g-greedy strategy. As we pointed out earlier, nothing at stake attackers will follow the full-greedy strategy to try to obtain the best outcome. From Lemma 4.4, we have proved that in our protocol, the amplification ratio for nothing at stake attackers (following the full-greedy strategy) is bounded, which means the number of blocks that attackers can contribute to a chain is also bounded. The length of chains will grow and follow Pascal's Triangle coefficients with greedy players. Based on the above results, we have showed that in the presence of full-greedy adversary, our protocol can still achieve chain soundness.

Selfish mining. Selfish mining is a significant threat to the cryptocurrencies. This is where one miner or mining pool do not publish and distribute a valid solution to the rest of the network once they solve the puzzle. Following this strategy, selfish miners will continuously maintain a lead during the whole mining process. Several analysis showed that the rewards that selfish miners can claim does not necessarily reflect the computing power they control. This greatly endangers the fairness of the PoW blockchain. In general, PoS systems are even more vulnerable to the selfish miners. This is because selfish miners in a PoS system can predict the moment they should generate two or more blocks on a private hidden chain in a given round.

Our design can effectively weaken the affect of selfish mining. As shown in Section 4, in the modified core-chain protocol where adversaries follow the g-greedy strategy (meaning players will try to extend the longest g layers of blockchain instead of the longest one),the blocks generated by the honest players will be defeated by the ones from selfish miners only if they are shorter more than g blocks. The greater g is, the lower the probability that selfish miners succeed.

Synchronization. In our construction, in the hash inequality $\mathsf{H}(\mathsf{hash}(B_i), \mathtt{round}, \textsc{pk}, \sigma) < \mathtt{T}_i$, each player relies on his local clock to set the value round. In our security analysis in previous sections, for simplicity, we assume these local clocks can be perfectly synchronized via a global clock. In our future work, we will improve our security analysis so that our protocol can be based on a "relaxed" global clock (in the sense that, players' local clocks may deviate from the "idealized" clock slightly). We note that, this "relaxed" global clock can be instantiated via the Network Time Protocol (NTP). Typically, NTP can synchronize players within tens of milliseconds over the public Internet.

Other considerations. We can also mimic Nakamoto's design and incentivize the players to participate in the protocol by collecting the "rewards". We note that new ideas (e.g., [46]) can be adopted. To extend our design idea to a full-fledged blockchain protocol, we also need to use authenticated data structure to more effectively manage the transactions. Instead of straightforwardly including the entire "payload" $\tilde{X}_i$ in the block $\tilde{B}_i$ (as in Section 6, and in [27, 45]), we can store a Merkle root in $\tilde{B}_i$. New ideas (e.g., [48]) can also be used in our design.

Acknowledgement: We are very grateful to Andrew Miller and Jonathan Katz for many useful discussions about proof-of-stake protocol design; in particular, we thank Andrew for pointing out a technical issue in an early version [26] of this proposal, and Jonathan for verifying the current results. We thank Alex Chepurnoy, Yi Ding, Tuyet Duong, and Yanxue Jia for helpful discussions about scalable blockchain protocols in the open network settings, and for proofreading the early versions of the paper. Finally, we thank Euorcrypt 2018 PC for their valuable feedback, and the presentation has been improved based on their suggestions.

References

[1] Litecoin. 2011. https://litecoin.org.

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

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

[4] 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.

[5] I. Bentov, A. Gabizon, and A. Mizrahi. Currencies without proof of work. In *Bitcoin Workshop*, 2016.

[6] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld. Proof of activity: Extending bitcoin's proof of work via proof of stake. In *SIGMETRICS Perform. Eval. Rev.*, pages 34–37. ACM, 2014.

[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] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. In C. Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Heidelberg, Dec. 2001.

[9] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten. SoK: Research perspectives and challenges for bitcoin and cryptocurrencies. In *2015 IEEE Symposium on Security and Privacy*, pages 104–121. IEEE Computer Society Press, May 2015.

[10] V. Buterin. A Next-Generation Smart Contract and Decentralized Application Platform. 2014. https://github.com/ethereum/wiki/wiki/White-Paper.

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

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

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

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

[15] 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.

[16] 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.

[17] J. Chen and S. Micali. Algorand. In *arXiv:1607.01341*, May 2017. http://arxiv.org/abs/1607.01341.

[18] 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.

[19] CryptoManiac. Proof of stake. novacoin wiki. 2014. https://github.com/novacoin-project/novacoin/wiki/Proof-of-stake/.

[20] P. Daian, R. Pass, and E. Shi. Snow white: Robustly reconfigurable consensus and applications to provably secure proofs of stake. In *Cryptology ePrint Archive, Report 2016/919*, April 2017. http://eprint.iacr.org/2016/919.

[21] B. David, P. Gazi, A. Kiayias, and A. Russelly. Ouroboros Praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In *EUROCRYPT*, 2018. http://eprint.iacr.org/2017/573.

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

[23] 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.

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

[25] 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.

[26] L. Fan and H.-S. Zhou. iChing: A Scalable Proof-of-Stake Blockchain in the Open Setting (or, How to Mimic Nakamoto's Design via Proof-of-Stake). July 2017. https://eprint.iacr.org/2017/656/20170705:220223.

[27] 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.

[28] J. A. Garay, A. Kiayias, and N. Leonardos. The bitcoin backbone protocol with chains of variable difficulty. In *CRYPTO*, 2017. https://eprint.iacr.org/2016/1048.

[29] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *SOSP*, 2017. https://eprint.iacr.org/2017/454.

[30] 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.

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

[32] 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.

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

[34] 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.

[35] A. Kiayias, A. Russell, B. David, and R. Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *CRYPTO*, 2017. http://eprint.iacr.org/2016/889.

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

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

[38] A. Lysyanskaya. Unique signatures and verifiable random functions from the DH-DDH separation. In M. Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 597–612. Springer, Heidelberg, Aug. 2002.

[39] 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.

[40] 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.

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

[42] A. Narayanan, J. Bonneau, , E. W. Felten, A. Miller, and S. Goldfeder. Bitcoin and cryptocurrency technology. 2015. https://www.coursera.org/learn/cryptocurrency.

[43] K. Nayak, S. Kumar, A. Miller, and E. Shi. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In *IEEE Euro SP* 2016. http://eprint.iacr.org/2015/796.

[44] S. Park, K. Pietrzak, A. Kwon, J. Alwen, G. Fuchsbauer, and P. Gaži. Spacemint: A cryptocurrency based on proofs of space. In *FC*, 2018. http://eprint.iacr.org/2015/528.

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

[46] R. Pass and E. Shi. FruitChains: A fair blockchain. In *PODC*, 2017. http://eprint.iacr.org/2016/916.

[47] R. Pass and E. Shi. The sleepy model of consensus. In *ASIACRYPT*, 2017. http://eprint.iacr.org/2016/918.

[48] L. Reyzin, D. Meshkov, A. Chepurnoy, and S. Ivanov. Improving authenticated dynamic dictionaries, with applications to cryptocurrencies. In *FC*, 2017. http://eprint.iacr.org/2016/994.

[49] A. Sapirshtein, Y. Sompolinsky, and A. Zohar. Optimal selfish mining strategies in bitcoin. In J. Grossklags and B. Preneel, editors, *FC 2016*, volume 9603 of *LNCS*, pages 515–532. Springer, Heidelberg, Feb. 2016.

[50] O. Schrijvers, J. Bonneau, D. Boneh, and T. Roughgarden. Incentive compatibility of bitcoin mining pool reward functions. In J. Grossklags and B. Preneel, editors, *FC 2016*, volume 9603 of *LNCS*, pages 477–498. Springer, Heidelberg, Feb. 2016.

[51] 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.

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

[53] G. Wood. Ethereum: A secure decentralized transaction ledger. 2014. http://gavwood.com/paper.pdf.

[54] F. Zhang, I. Eyal, R. Escriva, A. Juels, and R. van Renesse. REM: Resource-Efficient Mining for Blockchains. In *USENIX Security*, 2017. https://eprint.iacr.org/2017/179.

A Supporting Materials

A.1 Implementing $\mathcal{F}_{\mathsf{rCERT}}$ in the $\{\hat{\mathcal{F}}_{\mathsf{CA}}, \hat{\mathcal{F}}_{\mathsf{uSIG}}, \mathcal{F}_{\mathsf{RO}}\}$-hybrid model

We denote ϕ_{rCERT} as the ideal protocol for an ideal functionality $\mathcal{F}_{\mathsf{rCERT}}$, and π_{rCERT} as protocol in the $\{\hat{\mathcal{F}}_{\mathsf{CA}}, \hat{\mathcal{F}}_{\mathsf{uSIG}}, \mathcal{F}_{\mathsf{RO}}\}$-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-player through his stake determines whether he is the elected leader in the current round or not; then he is able to generate a valid signature, which can later be verified by any other players. The protocol π_{rCERT} is formally described in Figure 17.

PROTOCOL π_{rCERT}

The protocol is parameterized by a PoS parameter p and a security parameter κ.
Each $\mathrm{P} \in \mathcal{P}_0$, *proceeds as follows.*
Initially, pass (KEYGEN, sid, ssid) for some sid, ssid where ssid $= (\mathrm{P}, \mathsf{ssid}')$ for some ssid' to the functionality $\hat{\mathcal{F}}_{\mathsf{uSIG}}$. Upon receiving (VERIFICATION-KEY, sid, ssid, PK) from $\hat{\mathcal{F}}_{\mathsf{uSIG}}$ where PK $\in \{0,1\}^{\mathrm{poly}(\kappa)}$, record PK. Next, send (REGISTER, sid, ssid, P, PK) to the functionality $\hat{\mathcal{F}}_{\mathsf{CA}}$ and receive (REGISTERED, sid, ssid) from the functionality $\hat{\mathcal{F}}_{\mathsf{CA}}$.

Stake Election: For each round, each registered party P sets $\phi_{\mathrm{P},h^{\mathrm{prev}}} z := 0$, then proceeds as follows. Upon receiving (ELECT, P, $\langle h^{\mathrm{prev}}, \mathtt{round}\rangle$) from the environment $\mathcal{Z}$,

1. If $\phi_{\mathrm{P},h^{\mathrm{prev}}} = 0$, send (SIGN, sid, ssid, P, $\langle h^{\mathrm{prev}}, \mathtt{round}\rangle$) to the functionality $\hat{\mathcal{F}}_{\mathsf{uSIG}}$.

   Upon receiving (SIGNATURE, sid, ssid, (P, $\langle h^{\mathrm{prev}}, \mathtt{round}\rangle$), σ) from $\hat{\mathcal{F}}_{\mathsf{uSIG}}$, send ($h^{\mathrm{prev}}, \mathtt{round}$, PK, σ) to the functionality $\mathcal{F}_{\mathsf{RO}}$ and receives $\tilde{h}^{\mathrm{puzz}}$.

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

   Set $\phi_{\mathrm{P},h^{\mathrm{prev}}} := 1$

   Send (ELECTED, P, σ, b) to the environment.

2. Otherwise, ignore the message.

Block Verification: Upon receiving (CORE-VERIFY, P, $\langle h^{\mathrm{prev}}, \mathtt{round}\rangle$, σ) from the environment $\mathcal{Z}$, send (RETRIEVE, sid, ssid) for some sid, ssid where ssid $= (\mathrm{P}, \mathsf{ssid}')$ for some ssid' to the functionality $\hat{\mathcal{F}}_{\mathsf{CA}}$. Upon receiving (RETRIEVED, sid, ssid, PK) from the functionality.

- If PK $\neq \perp$, send (VERIFY, sid, ssid, (P, $\langle h^{\mathrm{prev}}, \mathtt{round}\rangle$, σ, PK) to the functionality $\hat{\mathcal{F}}_{\mathsf{uSIG}}$. Upon receiving (VERIFIED, sid, ssid, (P, $\langle h^{\mathrm{prev}}, \mathtt{round}\rangle$), f) from the functionality $\hat{\mathcal{F}}_{\mathsf{uSIG}}$. If $f = 1$, send ($h^{\mathrm{prev}}, \mathtt{round}$, PK, σ) to the functionality $\mathcal{F}_{\mathsf{RO}}$ and receives $\tilde{h}^{\mathrm{puzz}}$.

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

  send (CORE-VERIFIED, (P, $\langle h^{\mathrm{prev}}, \mathtt{round}\rangle$), σ, f) to the environment.

- Else, if PK $= \perp$, set $f = 0$, send (CORE-VERIFIED, (P, $\langle h^{\mathrm{prev}}, \mathtt{round}\rangle$), σ, f) to the environment.

Figure 17: Resource certification protocol π_{rCERT}.

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 $\mathsf{EXEC}^{\hat{\mathcal{F}}_{\mathsf{CA}},\hat{\mathcal{F}}_{\mathsf{uSIG}},\mathcal{F}_{\mathsf{RO}}}_{\pi_{\mathsf{rCERT}},\mathcal{A},\mathcal{Z}}$ 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 $\mathcal{Z}$.

Lemma A.1. *Consider ϕ_{rCERT} described above and π_{rCERT} in Figure 17. It holds that the two ensembles $\mathsf{EXEC}^{\mathcal{F}_{\mathsf{rCERT}}}_{\phi_{\mathsf{rCERT}},\mathcal{S},\mathcal{Z}}$ and $\mathsf{EXEC}^{\hat{\mathcal{F}}_{\mathsf{CA}},\hat{\mathcal{F}}_{\mathsf{uSIG}},\mathcal{F}_{\mathsf{RO}}}_{\pi_{\mathsf{rCERT}},\mathcal{A},\mathcal{Z}}$ are perfectly indistinguishable.*

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 p for ϕ_{rCERT} as follows. $\mathcal{S}$ stores a table T

Initialization and Stake Election:

- Simulating the execution with an uncorrupted party P as follows. When $\mathcal{S}$ receives in the ideal process a message (Core-Verify, P, $\langle h^{\text{prev}}, \texttt{round}\rangle, \sigma$) from $\mathcal{F}_{\mathsf{rCERT}}$, where P is uncorrupted, it proceeds as follows:
  - If this is the first time that P generates a signature, then simulate for $\mathcal{A}$ the process of key generation . That is, send to $\mathcal{A}$ (in the name of $\hat{\mathcal{F}}_{\mathsf{uSIG}}$) (Keygen, sid, ssid) to the adversary, and receive (Verification-Key, sid, ssid, pk) from the adversary $\mathcal{A}$, then send (Register, sid, ssid, pk) to the adversary $\mathcal{A}$; upon receiving (Registered, sid, ssid) from the adversary, then record the pair (ssid, pk).
  - Simulate for $\mathcal{A}$ the process of signing $\langle h^{\text{prev}}, \texttt{round}\rangle$, send (Sign, sid, ssid, $\langle h^{\text{prev}}, \texttt{round}\rangle$) (in the name of $\hat{\mathcal{F}}_{\mathsf{uSIG}}$) to $\mathcal{A}$. Upon receiving (Signature, sid, ssid, $\langle h^{\text{prev}}, \texttt{round}\rangle, \sigma$) from the adversary, forward (Signature, sid, ssid, $\langle h^{\text{prev}}, \texttt{round}\rangle, \sigma$) to $\mathcal{F}_{\mathsf{rCERT}}$.
- Simulating the execution with a corrupted party P with $\hat{\mathcal{F}}_{\mathsf{CA}}$ and $\hat{\mathcal{F}}_{\mathsf{uSIG}}$ as follows.
  - Upon receiving (Keygen, sid, ssid) from party P, send (in the name of $\hat{\mathcal{F}}_{\mathsf{uSIG}}$) (Keygen, sid, ssid) to the adversary, and receive (Verification-Key, sid, ssid, pk) from the adversary $\mathcal{A}$, then send (Verification-Key, sid, ssid, pk) (in the name of $\hat{\mathcal{F}}_{\mathsf{uSIG}}$) to P and record (ssid, P, pk).
  - Upon receiving (Register, sid, ssid, P, pk) from party P, send (Register, sid, ssid, pk) to the adversary $\mathcal{A}$; upon receiving (Registered, sid, ssid) from the adversary, then record the pair (ssid, pk) and (Registered, sid, ssid) to P. Then, instruct the corrupted party P send the message (Elect, P, $\langle h^{\text{prev}}, \texttt{round}\rangle$) to $\mathcal{F}_{\mathsf{rCERT}}$.
  - Upon receiving (Sign, sid, ssid, P, $\langle h^{\text{prev}}, \texttt{round}\rangle$) from P, check if (ssid, $\langle h^{\text{prev}}, \texttt{round}\rangle, \cdot, \cdot$) has been recorded. If yes, ignore the request. Otherwise, send (Sign, sid, ssid, $\langle h^{\text{prev}}, \texttt{round}\rangle$) to the adversary. Upon receiving (Signature, sid, ssid, $\langle h^{\text{prev}}, \texttt{round}\rangle, \sigma$) from the adversary, output (Signature, sid, ssid, $\langle h^{\text{prev}}, \texttt{round}\rangle, \sigma$) to party P, and record the entry (ssid, $\langle h^{\text{prev}}, \texttt{round}\rangle, \sigma$, pk, 1).
- Simulate the interaction of any party P with $\mathcal{F}_{\mathsf{RO}}$ as follows. For query $(h^{\text{prev}}, \texttt{round}, \textsc{pk}, \sigma)$ from party P, $(h^{\text{prev}}, \texttt{round}, \textsc{pk}, \sigma)$, send (Core-Verify, P, $\langle h^{\text{prev}}, \texttt{round}\rangle, \sigma$) to $\mathcal{F}_{\mathsf{rCERT}}$ and receive (Core-Verified, (P, $\langle h^{\text{prev}}, \texttt{round}\rangle$), σ, f). If $f = 0$, choose random $\tilde{h}^{\text{puzz}} \in \{0,1\}^\kappa$ such that $\tilde{h}^{\text{puzz}} > \mathrm{T}$ where $\mathrm{T} = p \cdot 2^\kappa$. If $f = 1$, choose $\tilde{h}^{\text{puzz}} \in \{0,1\}^\kappa$ such that $\tilde{h}^{\text{puzz}} \leq \mathrm{T}$. Then store $((h^{\text{prev}}, \texttt{round}, \textsc{pk}, \sigma), \tilde{h}^{\text{puzz}})$ in the table T and send $\tilde{h}^{\text{puzz}}$ to P.

Block Verification:

- Simulating the execution with an uncorrupted party P as follows. When notified by $\mathcal{F}_{\mathsf{rCERT}}$ that some uncorrupted party P made a verification request, proceed as follows. Upon receiving message $(\textsc{Core-Verify}, \mathrm{P}, \langle h^{\mathrm{prev}}, \mathtt{round}\rangle, \sigma)$ from $\mathcal{F}_{\mathsf{rCERT}}$, then forward this message to $\mathcal{A}$ (in the name of $\hat{\mathcal{F}}_{\mathsf{uSIG}}$). Forward $\mathcal{A}$'s response back to $\mathcal{F}_{\mathsf{rCERT}}$.
- Simulating the execution with a corrupted party P with $\hat{\mathcal{F}}_{\mathsf{CA}}$ and $\hat{\mathcal{F}}_{\mathsf{uSIG}}$ as follows.
  - Upon receiving $(\textsc{Retrieve}, \mathsf{sid}, \mathsf{ssid})$ from a corrupted party P for some party P', send $(\textsc{Retrieve}, \mathsf{sid}, \mathsf{ssid}, \mathrm{P})$ to the adversary $\mathcal{A}$, upon receiving $(\textsc{Retrieved}, \mathsf{sid}, \mathsf{ssid}, \mathrm{P})$ from the adversary. Then, if there is a recorded pair $(\mathsf{ssid}, \textsc{pk})$ output $(\textsc{Retrieved}, \mathsf{sid}, \mathsf{ssid}, \textsc{pk})$ to P. Else output $(\textsc{Retrieved}, \mathsf{sid}, \mathsf{ssid}, \perp)$ to P. Then, instruct the corrupted party P to send the message $(\textsc{Core-Verify}, \mathrm{P}, \langle h^{\mathrm{prev}}, \mathtt{round}\rangle, \sigma)$ to $\mathcal{F}_{\mathsf{rCERT}}$.
  - Upon receiving $(\textsc{Verify}, \mathsf{sid}, \mathsf{ssid}, (\mathrm{P}, \langle h^{\mathrm{prev}}, \mathtt{round}\rangle, \sigma, \textsc{pk})$ from a corrupted party P, generate a response following the instructions of $\hat{\mathcal{F}}_{\mathsf{uSIG}}$.
- Simulate the interaction of any party P with $\mathcal{F}_{\mathsf{RO}}$ as follows. For query $(h^{\mathrm{prev}}, \mathtt{round}, \textsc{pk}, \sigma)$ from party P check if $((h^{\mathrm{prev}}, \mathtt{round}, \textsc{pk}, \sigma), \tilde{h}^{\mathrm{puzz}}))$ in the table T and send $\tilde{h}^{\mathrm{puzz}}$ to party P. Otherwise, send $(h^{\mathrm{prev}}, \mathtt{round}, \textsc{pk}, \sigma)$, send $(\textsc{Core-Verify}, \mathrm{P}, \langle h^{\mathrm{prev}}, \mathtt{round}\rangle, \sigma)$ to $\mathcal{F}_{\mathsf{rCERT}}$ and receive $(\textsc{Core-Verified}, (\mathrm{P}, \langle h^{\mathrm{prev}}, \mathtt{round}\rangle), \sigma, f)$. If $f = 0$, choose random $\tilde{h}^{\mathrm{puzz}} \in \{0,1\}^{\kappa}$ such that $\tilde{h}^{\mathrm{puzz}} > \mathtt{T}$ where $\mathtt{T} = p \cdot 2^{\kappa}$. If $f = 1$, choose $\tilde{h}^{\mathrm{puzz}} \in \{0,1\}^{\kappa}$ such that $\tilde{h}^{\mathrm{puzz}} \leq \mathtt{T}$. Then store $((h^{\mathrm{prev}}, \mathtt{round}, \textsc{pk}, \sigma), \tilde{h}^{\mathrm{puzz}}))$ in the table T and send $\tilde{h}^{\mathrm{puzz}}$ to P.

We now show that the two ensembles $\mathsf{EXEC}^{\mathcal{F}_{\mathsf{rCERT}}}_{\phi_{\mathsf{rCERT}}, \mathcal{S}, \mathcal{Z}}$ and $\mathsf{EXEC}^{\hat{\mathcal{F}}_{\mathsf{CA}}, \hat{\mathcal{F}}_{\mathsf{uSIG}}, \mathcal{F}_{\mathsf{RO}}}_{\pi_{\mathsf{rCERT}}, \mathcal{A}, \mathcal{Z}}$ are perfectly close. Notice that for each election query, the adversary $\mathcal{S}$ is noticed by the functionality $\mathcal{F}_{\mathsf{rCERT}}$ whether this query is successful or not, then it samples the output randomly from a set $\{0,1\}^{\kappa}$ that satisfied inequality $\mathsf{H}(h^{\mathrm{prev}}, \mathtt{round}, \textsc{pk}, \sigma) \leq \mathtt{T}$ if the query is successful. Putting them together, the views of players in the two executions are perfectly indistinguishable.

□

A.2 Multi-Session Certificate Authority Functionality $\hat{\mathcal{F}}_{\mathsf{CA}}$

We present the certificate authority functionality following the modeling of [14, 15].

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

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

1. Upon receiving message $(\textsc{Register}, \mathsf{sid}, \mathsf{ssid}, v)$ from party $\mathrm{P} \in \mathcal{P}$, verify that $\mathsf{ssid} = (\mathrm{P}, \mathsf{ssid}')$ 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 $\mathrm{P}' \in \mathcal{P}$, send $(\textsc{Retrieve}, \mathsf{sid}, \mathsf{ssid}, \mathrm{P}')$ to the adversary, upon receiving $(\textsc{Retrieved}, \mathsf{sid}, \mathsf{ssid}, \mathrm{P}')$ 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 18: Multi-session certificate authority functionality $\hat{\mathcal{F}}_{\mathsf{CA}}$.

A.3 Multi-Session Signature Functionality $\hat{\mathcal{F}}_{\mathsf{uSIG}}$

We present the multi-session version of the digital signature functionality in [14]. While the digital signature functionality can be realized by a signature protocol which is based on ordinary signature scheme, this functionality here can be realized by a signature protocol which is based on unique signature scheme. The underlying part highlights the difference be between ours and that in [14]. We note that the definition of unique signature scheme can be found in next subsection.

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

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 input $(\textsc{Keygen}, \mathsf{sid}, \mathsf{ssid})$ 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 $(\textsc{Verification-Key}, \mathsf{sid}, \mathsf{ssid}, \textsc{pk})$ from the adversary, output $(\textsc{Verification-Key}, \mathsf{sid}, \mathsf{ssid}, \textsc{pk})$ to the party P, and record $(\mathsf{ssid}, \mathrm{P}, \textsc{pk})$.

Signature Generation: Upon receiving input $(\textsc{Sign}, \mathsf{sid}, \mathsf{ssid}, m)$ from a signer $P \in \{S_1, \ldots, S_k\}$, verify that $\mathsf{ssid} = (P, \mathsf{ssid}')$ for some ssid' $\underline{\text{and no } (\mathsf{ssid}, m, \cdot, \cdot) \text{ has been recorded}}$. 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 $(\mathsf{ssid}, m, \sigma, \textsc{pk}, 0)$ 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 $(\mathsf{ssid}, m, \sigma, \textsc{pk}, 1)$.

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

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

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

Figure 19: Multi-session signature functionality $\hat{\mathcal{F}}_{\mathsf{uSIG}}$.

A.3.1 Unique signature scheme.

Unique signature scheme was introduced in [38], which consists of four algorithms, a randomized key generation algorithm uKeyGen, a deterministic key verification algorithm uKeyVer, a deterministic signing algorithm uSign, and a deterministic verification algorithm uVerify. We expect for each verification key there exists only one signing key. We also expect for each pair of message and verification key, there exists only one signature. We have the following definition.

Definition A.2. *We say* $(\mathsf{uKeyGen}, \mathsf{uKeyVer}, \mathsf{uSign}, \mathsf{uVerify})$ *is a strengthened unique signature scheme, if it satisfies:*

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

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

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

$$\Pr\left[\begin{array}{l} (\textsc{pk}, \textsc{sk}_1, \textsc{sk}_2) \leftarrow \mathcal{A}(1^\kappa) \ : \\ \mathsf{uKeyVer}(\textsc{pk}, \textsc{sk}_1) = 1 \wedge \mathsf{uKeyVer}(\textsc{pk}, \textsc{sk}_2) = 1 \wedge \textsc{sk}_1 \neq \textsc{sk}_2 \end{array}\right] \leq \mathsf{negl}(\kappa)$$

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

$$\Pr\left[(\textsc{pk}, \textsc{sk}) \leftarrow \mathsf{uKeyGen}(1^\kappa); \sigma := \mathsf{uSign}(\textsc{sk}, x) \ : \ \mathsf{uVerify}(\textsc{pk}, x, \sigma) = 1\right] \geq 1 - \mathsf{negl}(\kappa)$$

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

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

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

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

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

Remark A.3 (Instantiations for the unique signature scheme)**.** *Efficient instantiations can be found in literature. For example, the well-known BLS signature [8] can be a good candidate.*

A.4 Additional Functionalities

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

A.4.1 Network communication $\mathcal{F}_{\mathsf{NET}}$

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 a semi-synchronous 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 20.

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

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

- Upon receiving $(\textsc{Broadcast}, m)$ from a party P at round r where $\mathrm{P} \in \mathcal{P}$, send $(\textsc{Broadcast}, m)$ to $\mathcal{S}$ and record (P, m, b, r) where $b = 0$.
- Upon receiving $(\textsc{Delay}, m, \mathrm{P}', t)$ from $\mathcal{S}$ where $\mathrm{P}' \in \mathcal{P}$ (here, the adversary can "spoof" the source of the message), then
  - If there is a record (P, m, b, r) such that $b = 0$ and $t \leq \Delta$, then send $(\textsc{Message}, \mathrm{P}', m)$ to all other PoS-players at round $r + t$ and reset $b := 1$.
  - Else, if $t > \Delta$, send $(\textsc{Message}, \mathrm{P}', m)$ to all other PoS-players at round $r + \Delta$ and reset $b := 1$.
  - Else, ignore the message.

Figure 20: Network functionality $\mathcal{F}_{\mathsf{NET}}$.

A.4.2 Random Oracle Functionality $\mathcal{F}_{\mathsf{RO}}$

The random oracle model (e.g., [4]) captures an idealization of a hash function. We here present the random oracle functionality $\mathcal{F}_{\mathsf{RO}}$ that has been defined in [30].

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

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

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

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

Figure 21: Random oracle functionality $\mathcal{F}_{\mathsf{RO}}$.