iChing: 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

July 5, 2017

Abstract

Bitcoin has proven to be very successful. The Bitcoin blockchain is backed up by a large-scale network of miners via proof-of-work mechanism. Unfortunately, these miners consume huge amount of non-recyclable resources (electricity and computing hardware). To address this issue, alternative blockchain constructions via proof-of-stake mechanism have been proposed. Unfortunately, those proposals either lack of security analysis or cannot scale to a large network of nodes in the open network setting where new players can safely join the blockchain protocol.

In this paper, we propose iChing, the first scalable proof-of-stake blockchain in the open setting. We show that our blockchain protocol can achieve several important security properties including common prefix, chain quality, chain growth, and chain soundness.

Contents

1 Introduction

Bitcoin and proof-of-work mechanism. Bitcoin system [Nak08] has proven to be very successful. The system was designed and implemented by an unknown researcher, under the name Satoshi Nakamoto nine years ago. The key idea is to use proof-of-work (PoW) mechanism [DN93, Bac02] to maintain a global public distributed ledger, also called blockchain, among a peer-to-peer network of nodes (called miners). Nakamoto's design has unique features: (1) the Bitcoin protocol can be executed in an *open* network environment, and all miners are allowed and encouraged/incentivized to invest certain amount of computing power to join the effort of maintaining the blockchain; (2) the protocol has very lower communication complexity and *can scale* to a large network of nodes.

From proof-of-work to proof-of-stake. The scalability of Bitcoin blockchain protocol is at a price: the system has "absorbed" or "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 stake)* 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., [NXT14, Kwo14, Vas14, BGM16]). In a nutshell, in a proof-of-stake based blockchain protocol, players are expected to prove ownership of a certain amount of recyclable resources (i.e., coins or stake). 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 intensively investigated. Notably, Garay et al. [GKL15] took the *provable security* approach and investigated Nakamoto's blockchain in a cryptographic framework (please also see [PSS17]); 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. Very recently, Duong et al. [DZ17] identified a new property, *chain soundness*, to ensure the security of new players, and showed that Nakamoto's protocol can achieve this new property. Indeed, Nakamoto provided a scalable, proof-of-work based blockchain protocol which can be proven secure in the open network environment.

Alternative provably secure scalable blockchain protocols have been proposed recently. For example, Duong et al. [DFZ16] constructed a scalable blockchain by combining proof-of-work and proof-of-stake mechanisms, in the open setting: common prefix, chain quality, and chain growth properties have been proved in their paper; chain soundness property is missing but it can been proven [DFZ17]. At the same time, several scalable proof-of-stake based protocols [CM17, KRDO17, DPS17]) were proposed: chain soundness property is missing in these proposals; in order to prove the chain soundness property, their blockchain protocols have to be augmented with certain non-trivial mechanism (e.g., majority voting) to ensure new participants to securely join the protocol execution. Unfortunately, the augmented protocols cannot scale to a large network of nodes. This leads to the following interesting question:

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

1.1 Our solution

We give an affirmative answer to the above question. In this paper we provide a natural mimic of Nakamoto's design but via proof-of-stake mechanism. Next, we illustrate our basic ideas step by step.

Nakamoto's design and proof-of-work (PoW) based core-chain. We first briefly review Nakamoto's design ideas [Nak08]. 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 [DN93, Bac02]. The *puzzle problem* 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 ordered blocks $B_1, B_2, \ldots, B_i$, 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:

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

where $\mathsf{H}(\cdot)$ and $\mathsf{hash}(\cdot)$ are two hash functions, $\mathit{payload}$ denotes a 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:

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

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. (We remark that, for simplicity, the above description is oversimplified; please see Section 6 for more discussions.)

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 the following hash inequality:

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

and the new block B_{i+1} is defined as

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

Proof-of-stake (PoS) based core-chain. To make our presentation accessible, we now present our proof-of-stake based core-chain protocol. (We will explain why the idea works, and explain how to extend it to a full-fledged blockchain immediately after the core-chain protocol.)

We intend to mimic Nakamoto's design. As described above, Nakamoto's Bitcoin blockchain is maintained by PoW-players (i.e., miners); there, each winning PoW-player can extend the blockchain with a new block. In contrast, our PoS based protocol will be maintained by PoS-players (i.e., stakeholders); now a PoS-player will extend the blockchain with a new block. Similar to that in the PoW-based core-chain protocol, a wining PoS-player is chosen with some probability but with a different hash inequality. More concretely, assume the longest core-chain for a PoS-player consists of the following ordered block-cores, $B_1, B_2, \ldots, B_i$; let $\mathtt{round}$ denote the current time (or round number); consider a *strengthened unique* digital signature scheme[1], $(\mathsf{uKeyGen}, \mathsf{uKeyVer}, \mathsf{uSign}, \mathsf{uVerify})$, and assume the PoS-player holds the signing-verification key pair $(\mathrm{SK}, \mathrm{PK})$. If the PoS-player is chosen, then the following hash inequality holds:

$$\mathsf{H}(\mathsf{hash}(B_i), \mathtt{round}, \mathrm{PK}, \sigma) < \mathtt{T}$$

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

$$B_{i+1} := \langle h_i, \mathtt{round}, \mathrm{PK}, \sigma \rangle$$

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 current time*, and the payload in the core-chain is empty; the puzzle solution consists of a PoS-player's verfication key and his signature of the context.

Why the proof-of-stake based core-chain works? We are inspired by Nakamoto's design [Nak08], and our protocol can be viewed as a proof-of-stake analogy of Nakamoto's. It has shown that Nakamoto's blockchain can achieve several important security properties [GKL15, PSS17, DZ17]. Next, we provide some high-level ideas that why our protocol can also achieve these important security properties.

[1]Unique signature scheme was introduced by Lysyanskaya [Lys02]. In the strengthened version of unique signature scheme, the key generation and key verification can be viewed as a variant of one-way relation (OWR) [DHLW10], and for each verification key, there is only one valid signing key. Furthermore, for each pair of verification key and message, there exists only one signature. The BLS signature [BLS01] can be a good instantiation of the strengthened unique signature scheme. More details can be found in Appendix A.3.

First, in PoW blockchain, all of the players generate a new block in every round with very low probability. The expected number of blocks that are generated in one round is much less than 1. This can be easily achieved by selecting suitable difficulty target $\mathtt{T}$ in our hash inequality. Second, in Nakamoto's proof-of-work based blockchain protocol, under the assumption that honest players control the majority of computing power, the honest players have higher probability to generate a block in a round than the malicious players. In our protocol, under the assumption that he honest players control the majority of stake, the honest players will have higher probability to generate a new block-core. Thirdly, before seeing the broadcast solution, no one can predict who can generate a block in a round. In our design, we put the signature of the previous block in the input of hash inequality. No one can predict the signature of honest player, so no one can predict the result of others.

Finally, the players in Nakamoto's protocol take the longest chain as the best chain; there the honest players control major computing power which can ensure that the honest players can generate the longest chain with high probability except the latest several blocks. As discussed above, the honest player will also generate the longest chain in our scheme. So the players in our scheme can take the longest chains as the best chain. This allows a new player can take the longest chain as the best chain.

From core-chain to blockchain. We now start to extend the core-chain to a blockchain in which payload will be included. 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 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

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

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

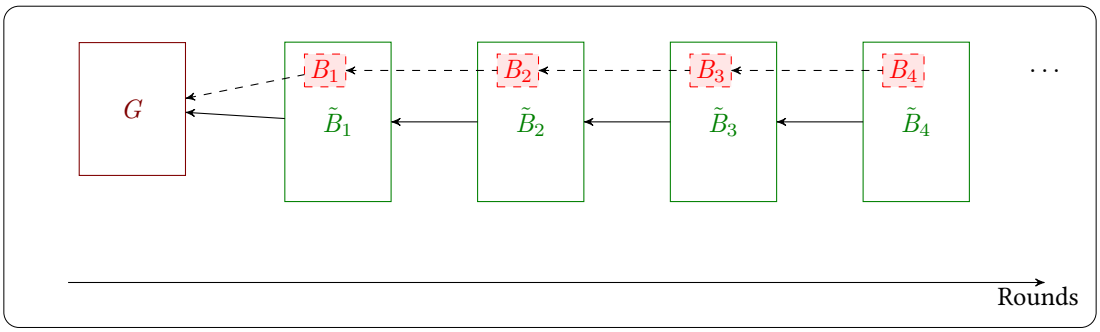


Figure 1: Blockchain structure

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 $B_1, B_2, B_3, \ldots$.

Discussions. In the original Nakamoto's design, adaptive difficulty adjustment mechanism has been introduced. When more computing power is invested into the system, the difficulty target $\mathtt{T}$ will decrease (and now the winning probability for each hash query will be smaller). In a proof-of-stake based system, we cannot expect the protocol is always executed by the same number of players. In Section 6, we show how to mimic Nakamoto's idea to enable our proof-of-stake protocol with adaptive difficulty adjustment mechanism. In addition, our proof-of-stake protocol can

be improved so that it can work properly in the non-flat model. (In the non-flat model, players may have different amount of stake.) Finally, many ideas in Nakamoto's design (e.g., suitable strategies for incentivizing the system and for effectively managing transactions in blockchain) can also be "borrowed" into our protocol. Please see Section 6 for more details.

1.2 Related work

Cryptocurrency and proof-of-work. Anonymous digital currency was introduced by Chaum [Cha82] in the early 1980s. The first decentralized currency system, Bitcoin [Nak08], was launched about 30 years later, by incentivizing a set of players to solve moderately-hard cryptographic puzzles (also called proof-of-work puzzles [DN93, Bac02]). After that, many cryptocurrency systems were created based on proof-of-work puzzles (e.g., LiteCoin [Lit11], Ethereum [But14, Woo14]).

The security of Bitcoin system has been analyzed in the rational setting, e.g., [ES14, Eya15, NKMS15, KKKT16, SSZ16, SBBR16], and also in the cryptographic setting, e.g., [GKL15, PSS17, SZ15, KP15, KP16, GKL17, DZ17]. Several important cryptographic properties, *common prefix* [GKL15, PSS17], *chain quality* [GKL15], *chain growth* [KP15], and *chain soundness* [DZ17] have been considered for proof-of-work blockchain protocols.

Combining proof-of-work and proof-of-stake. The idea of combining proof-of-work and proof-of-stake has been studied in [KN12, Cry14, BLMR14, DFZ16, CDFZ17]. Very recently, Duong et al [DFZ16, DFZ17] provided the first provably secure and scalable blockchain via proof-of-work/proof-of-stake in the open setting.

Proof-of-stake. Using virtual resources (i.e., stake) to construct cryptocurrency has been intensively considered but in an ad hoc way, in the Bitcoin community. In high level, the proof-of-stake mechanism asks protocol players to prove ownership of virtual resources. Only those players who can provide such proofs are allowed to maintain the blockchain. Since the inception of the idea in an online forum [Bit11], several proof-of-stake proposals have been introduced and/or implemented (e.g., [NXT14, Kwo14, Vas14, But15, BGM16]).

Very recently, several provably secure proof-of-stake based blockchain proposals (e.g., [CM17, KRDO17, DPS17]) have been developed. Unfortunately, chain soundness property is missing in these proposals; in order to achieve the chain soundness property, these blockchain protocols have to be augmented with certain non-trivial mechanism (e.g., majority voting) to ensure new participants to securely join the protocol execution. However, the augmented protocols cannot scale to a large network of nodes. Before our work, *how to construct a provably secure, scalable proof-of-stake blockchain in the open setting* is an open question.

Additional alternative mechanisms. Alternative consensus techniques via different resources have been considered. For example, the physical storage resource is used in [PPK+15, MJS+14]. A hybrid proposal of utilizing both computing and space resources, called proof-of-space-time was introduced in [MO16]. Recently, blockchain protocols via trusted hardware have also been proposed [Int16].

Organization. In Section 2, we present the security analysis framework. In Section 3, we provide the details of our proof-of-stake based core-chain construction, and then in Section 4 we give the security analysis. Then in Section 5, we provide the construction of our main blockchain protocol as well as the security analysis. Further extensions and related discussions can be found in Section 6. Finally, additional supporting materials can be found in Appendix A.

2 Model

In order to study the security of Bitcoin-like protocols, Garay et al. [GKL15] proposed a cryptographic framework and showed that the (simplified) Bitcoin can achieve several important security properties. Then, Pass et al. [PSS17] extended Garay et al.'s framework [GKL15] by considering a more realistic communication network (i.e., partial synchronous network) in which messages from honest players can be delayed with an upper bound units of time, i.e., Δ execution rounds. Duong et al [DZ17] extended the previous modeling effort further by considering even

more realistic scenario in which existing players and new players behavior differently. Below we describe the model of protocol execution that formulated in previous framework [GKL15, PSS17, DZ17].

2.1 Blockchain protocol executions

Network communication. The underlying communication for blockchain protocols are formulated via a functionality $\mathcal{F}_{\mathsf{NET}}$ which captures the atomic unauthenticated "send-to-all" broadcast in 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 2.

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

The execution of proof-of-stake blockchain protocol. Following Canetti's formulation of the "real world" executions [Can00a, Can00b], 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. We consider the execution of the 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 execution proceeds in *rounds*. 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 of the protocol Π, e.g., a genesis block. The environment $\mathcal{Z}$ first actives the adversary $\mathcal{A}$ and provides instructions for the adversary. The execution proceeds in rounds, and in each round, a protocol party could be activated by the environment or the functionalities.

- In each 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 receive subroutine output message $(\textsc{Message}, \mathrm{P}', m)$ for any $\mathrm{P}' \in \mathcal{P}$, from $\mathcal{F}_{\mathsf{NET}}$. It then interacts with the functionality $\mathcal{F}_{\mathsf{Setup}}$ and receives some output y.
  - Next, execute 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 obtain an update local state *state* and an outgoing message m', i.e.,$\{state, m'\} \leftarrow \Pi(state, x, y, m)$. After that, send $(\textsc{Broadcast}, m')$ to $\mathcal{F}_{\mathsf{NET}}$ and then return $(\textsc{Return-Stake}, \mathrm{P})$ to the environment $\mathcal{Z}$.

– At any round r of the execution, $\mathcal{Z}$ can send message $(\text{CORRUPT}, \mathrm{P})$, where $\mathrm{P} \in \mathcal{P}$, to adversary $\mathcal{A}$. Party P then remains honest till round $r + \Delta$. Then $\mathcal{A}$ will have access to the party's local state and control P from round $r + \Delta$.

Let $\mathsf{EXEC}^{\mathcal{F}_{\mathsf{Setup}},\mathcal{F}_{\mathsf{NET}}}_{\Pi,\mathcal{A},\mathcal{Z}}$ be a random variable denoting the joint VIEW of all parties (i.e., all their inputs, random coins and messages received, including those from the random oracle and signatures) in the above $\{\mathcal{F}_{\mathsf{Setup}}, \mathcal{F}_{\mathsf{NET}}\}$-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 $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$.

Initial state. In our PoS blockchain system, we assume the stakes are already distributed before the system start. We assume there are n PoS-players. A PoS-player P_i, where $1 \leq i \leq n$, holds the signing-verification key pair $(\mathrm{SK}_i, \mathrm{PK}_i)$ as account. A PoS-player P_i also holds v_i stakes where $v_i > 0$. The initial public state $\mathbf{I}$ consists of a list of entries (PK_i, v_i). All players are allowed to access $\mathbf{I}$ to obtain the verification key of a PoS-player and check the stakes of a PoS-player. For simplicity, we focus on the flat model where every player has some amount of stakes, and assume $v_i = 1$. (We will consider the non-flat model where v_i can be different values in Section 6.)

Remark 2.1. *For simplicity, we focus on the idealized "flat" model where all PoS-players have the same amount of stake. 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 stake, 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 6, 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.*

2.2 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 $\mathcal{C}[\neg\kappa]$.

2.3 Security properties

Security properties for existing players. Several important security properties for blockchain protocols have been defined: *common prefix property* [GKL15, PSS17], *chain quality property* [GKL15], and *chain growth property* [KP15]. Next, we review the security definitions.

Definition 2.2 (Chain growth for existing players)**.** *Consider a blockchain protocol Π with a set $\mathcal{P}$ of players in the open setting. The chain growth property $\mathcal{Q}_{\mathrm{cg}}$ with parameter $g \in \mathbb{R}$, states the following:*
for any existing honest player P' with local chain $\mathcal{C}'$ at round r', and existing 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 2.3 (Common prefix for existing players)**.** *Consider a blockchain protocol Π with a set $\mathcal{P}$ of players in the open setting. The common prefix property $\mathcal{Q}_{\mathrm{cp}}$ states the following:*
for any existing honest player P' with local chain $\mathcal{C}'$ at round r', and existing 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 2.4 (Chain quality for exiting players)**.** *Consider a blockchain protocol Π with a set $\mathcal{P}$ of players in the open setting. The chain quality property $\mathcal{Q}_{\mathrm{cq}}$, with parameters μ, ℓ, where $\mu \in \mathbb{R}$ and $\ell \in \mathbb{N}$, states the following:*
for any existing 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 that for large enough ℓ consecutive blocks of $\mathcal{C}$ the ratio of honest blocks is at least μ.

Security properties for new players. *Chain soundness property* was recently introduced in [DZ17]. Intuitively, we should provide the security guarantee that new players can obtain the recent blockchain which is compatible with the local chain of an existing player in some recent rounds. This property is not needed in the *closed* setting where new players are not allowed. However, this property is critical for blockchain protocols in the open setting. Without this security requirement, unsatisfactory protocols could be allowed. The chain soundness definition is formally described as follows.

Definition 2.5 (Chain soundness for new players)**.** *Consider a blockchain protocol* Π *with a set* $\mathcal{P}$ *of players in the open setting. Consider a new player* $\mathrm{P} \in \mathcal{P}$ *with local chain* $\mathcal{C}$ *in round* r*, in* $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$*. The* chain soundness *property* $\mathcal{Q}_{\mathrm{cs}}$*, with security parameter* κ *for new players, states the following: for any new player* P *in round* r*, there exists a pair existing players* P' *with local chain* $\mathcal{C}'$ *at round* r *such that* $\mathcal{C}'[\neg\kappa] \preceq \mathcal{C}$ *and* $\mathcal{C}[\neg\kappa] \preceq \mathcal{C}''$*.*

3 Proof-of-stake core-chain

In this section, we provide the details of our core-chain protocol. This core-chain protocol will the further extended to a full-fledged blockchain in Sections 5 and 6. We mimic Nakamoto's design: Nakamoto's protocol is maintained by PoW-players (i.e., miners), and each winning PoW-player can extend the longest blockchain with a new block; here, our core-chain protocol is maintained by PoS-players (i.e., stakeholders), and a winning PoS-player extends the longest blockchain with a new block. In Nakamoto's design, a PoW-player is chosen as the winner with some probability based on a moderately hard puzzle (defined by a hash inequality); here, in our PoS based core-chain protocol, a wining PoS-player is chosen with some probability but with a different hash inequality.

More concretely, assume the longest core-chain for a PoS-player consists of the following ordered block-cores, $B_1, B_2, \ldots, B_i$; let $\mathtt{round}$ denote the current time slots (or round number); consider a *strengthened unique* signature scheme (see Appendix A.3 and [Lys02, BLS01]) $(\mathsf{uKeyGen}, \mathsf{uKeyVer}, \mathsf{uSign}, \mathsf{uVerify})$, and assume the PoS-player holds the signing-verification key pair $(\mathrm{SK}, \mathrm{PK})$. If the PoS-player is chosen, then the following hash inequality holds:

$$\mathsf{H}(\mathsf{hash}(B_i), \mathtt{round}, \mathrm{PK}, \sigma) < \mathtt{T}$$

where $\sigma := \mathsf{uSign}(\mathrm{SK}, \langle h_i, \mathtt{round}\rangle)$, and $h_i := \mathsf{hash}(B_i)$. The new block-core B_{i+1} is defined as

$$B_{i+1} := \langle h_i, \mathtt{round}, \mathrm{PK}, \sigma\rangle$$

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 current time*, and the payload in the core-chain is empty; the puzzle solution consists of a PoS-player's verification key and his signature of the context.

Next we will provide a formal description for our protocol. We use a setup functionality $\mathcal{F}_{\mathsf{rCERT}}^{\mathbf{I}}$ 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 $(\mathsf{uKeyGen}, \mathsf{uKeyVer}, \mathsf{uSign}, \mathsf{uVerify})$.

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

In our core-chain protocol design, we will use the setup functionality, resource certification functionality $\mathcal{F}_{\mathsf{rCERT}}^{\mathbf{I}}$ (in Figure 3) that introduced by the authors in [DFZ16]. For completeness, we describe the functionality below.

Resource certification functionality $\mathcal{F}_{\mathsf{rCERT}}^{\mathbf{I}}$. The functionality consists of four phases, "Stake Resource Registration", "Stake Election", "Signature Generation" and "Stake Verification". At any time step, a PoS-player P can send a register command $(\textsc{Stake-Register}, \mathrm{P})$ to functionality $\mathcal{F}_{\mathsf{rCERT}}^{\mathbf{I}}$ for registration. The functionality then records $(\mathrm{P}, \mathbb{b}_{\mathrm{P}})$ where $\mathbb{b}_{\mathrm{P}} = 1$, if the party P is permitted. If the player discontinues the services, this bit is set to 0 indicating that the stake would not be granted to this player any longer. Then, for each execution round, a registered PoS-player P is granted *one unit of the stake*, and he can then request the functionality for leader election in this round. Specifically, he can send message $(\textsc{Elect}, \mathrm{P}, \mathit{context})$ to the functionality; the functionality then with probability p selects this party as the leader and notifies the player whether he is selected or not. Next, if the party P is elected

FUNCTIONALITY $\mathcal{F}_{\mathsf{rCERT}}^{\mathbf{I}}$

The functionality is parameterized by a difficulty parameter p, a security parameter κ, initial setup information $\mathbf{I}$, and interacts with a set $\mathcal{P}$ of parties, as well as an adversary.

Stake Resource Registration.

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

For each round, set $\phi_{\mathrm{P}} := 0$ for every registered party $\mathrm{P} \in \mathcal{P}$, then proceed as follows.

Stake Election: Upon receiving (ELECT, P, *context*) from a PoS-player P, proceed as follows.

1. If $(\mathrm{P}, \mathbb{b}_{\mathrm{P}})$ is recorded where $\mathbb{b}_{\mathrm{P}} = 1$ and $\phi_{\mathrm{P}} = 0$, (the party P registered and granted one unit of stake)
   - with probability p, set $\phi_{\mathrm{P}} := 1$ and $\mathsf{f} := 1$, send (ELECTED, P, f) to P, and record the entry (P, *context*) (the party P is elected.)
   - with probability $1-p$, set $\phi_{\mathrm{P}} := 1$ and $\mathsf{f} := 0$, and send (ELECTED, P, f) to P (the party P is not elected.)
2. Otherwise, if any of the following cases occur:
   - $(\mathrm{P}, \mathbb{b}_{\mathrm{P}})$ is not recorded, (the party P is not registered yet)
   - or $(\mathrm{P}, \mathbb{b}_{\mathrm{P}})$ is recorded and $\mathbb{b}_{\mathrm{P}} = 0$ (the party P registered and then unregistered),
   - or $\phi_{\mathrm{P}} = 1$, (the party P already used the granted stake unit)

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

Signature Generation:
Upon receiving (CORE-SIGN, P, *context*) from a party P, send (CORE-SIGN, P, *context*) to the adversary.
Upon receiving (SIGNATURE, P, *context*, σ) from the adversary, verify that no entry (P, *context*, σ, 0) is recorded. If it is, then output an error message (ERROR) to P and halt. Else, output (CORE-SIGNED, P, *context*, σ) to P, and record the entry (P, *context*, σ, 1).

Stake Verification: Upon receiving (CORE-VERIFY, P, *context*, σ) 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 (CORE-VERIFY, P, *context*, σ) to the adversary.
   Upon receiving (CORE-VERIFIED, P, *context*, ϕ) from the adversary, do:
   - If (P, *context*, σ, 1) is recorded, then set $f := 1$.
   - Else, if P is not corrupted, and no entry (P, *context*, σ', 1) for any σ' is recorded, then set $f := 0$ and record the entry (P, *context*, σ, f).
   - Else, if there is an entry (P, *context*, σ, f'), then set $f := f'$.
   - Else, set $f := \phi$, and record the entry (P, *context*, σ, f).

   Output (CORE-VERIFIED, P, *context*, 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 (CORE-VERIFIED, P, *context*, f) to the party P'.

Figure 3: Resource certification functionality $\mathcal{F}_{\mathsf{rCERT}}^{\mathbf{I}}$.

as the leader, the elected party then asks the functionality $\mathcal{F}_{\mathsf{rCERT}}^{\mathbf{I}}$ to provide the signature of *context*. The functionality therefore requests the adversary to produce the signature by command $(\textsc{Core-Sign}, \mathrm{P}, \mathit{context})$, and then waits until the adversary responds by a signature σ. The functionality after that checks if no entry $(\mathrm{P}, \mathit{context}, \sigma, 0)$ has been recorded. Note that, the indicator 0 implies that this is not a valid signature (the verification fails). If this entry is not recorded, then the functionality simply passes the signature σ to P and stores $(\mathrm{P}, \mathit{context}, \sigma, 1)$. If entry $(\mathrm{P}, \mathit{context}, \sigma, 0)$ is already recorded, this implies no signature has been generated for *context* yet. Then, the functionality outputs an error and halts.

Next, the verification process of $\mathcal{F}_{\mathsf{rCERT}}^{\mathbf{I}}$ proceeds as follows. Upon receiving a verification request, the functionality then asks the adversary to verify the signature. The functionality, upon receiving the verification decision from the adversary, would ensure the completeness, unforgeability, and guarantees consistency properties of the signature scheme.

How to implement functionality $\mathcal{F}_{\mathsf{rCERT}}^{\mathbf{I}}$? Functionality $\mathcal{F}_{\mathsf{rCERT}}^{\mathbf{I}}$ can be easily implemented. As noticed in [DFZ16], functionality $\mathcal{F}_{\mathsf{rCERT}}^{\mathbf{I}}$ can be a "resource" analog of the *multi-session* version of certificate functionality $\mathcal{F}_{\mathsf{CERT}}$ in [Can03]. Note that $\mathcal{F}_{\mathsf{CERT}}$ can be implemented in the $\{\mathcal{F}_{\mathsf{CA}}, \mathcal{F}_{\mathsf{SIG}}\}$-hybrid model [Can03]. We can follow the approach to implement our functionality $\mathcal{F}_{\mathsf{rCERT}}^{\mathbf{I}}$. Here, the functionality $\mathcal{F}_{\mathsf{rCERT}}^{\mathbf{I}}$ can be instantiated in the $\{\mathcal{F}_{\mathsf{rCA}}^{\mathbf{I}}, \hat{\mathcal{F}}_{\mathsf{SIG}}\}$-hybrid model, where $\hat{\mathcal{F}}_{\mathsf{SIG}}$ is a *multi-session* signature functionality [CR03] and the resource certificate authority functionality $\mathcal{F}_{\mathsf{rCA}}^{\mathbf{I}}$ can be implemented in the $\{\hat{\mathcal{F}}_{\mathsf{CA}}, \mathcal{F}_{\mathsf{RO}}\}$-hybrid model. In addition, the multi-session certificate authority functionality $\hat{\mathcal{F}}_{\mathsf{CA}}$ can be implemented by a "mature" blockchain. Please refer to Appendix A.2 for more details about resource certificate authority functionality $\mathcal{F}_{\mathsf{rCA}}^{\mathbf{I}}$ and multi-session signature functionality$\hat{\mathcal{F}}_{\mathsf{SIG}}$; note that, $\mathcal{F}_{\mathsf{rCA}}^{\mathbf{I}}$ can be viewed as a "resource" analogy of the multi-session version of certificate authority functionality $\mathcal{F}_{\mathsf{CA}}$ in [Can03]. Note that $\mathcal{F}_{\mathsf{rCA}}^{\mathbf{I}}$ can be instantiated in the $\{\hat{\mathcal{F}}_{\mathsf{CA}}, \mathcal{F}_{\mathsf{RO}}\}$-hybrid model (details can be found in [DFZ16]). Then $\hat{\mathcal{F}}_{\mathsf{CA}}$ can be implemented by a mature blockchain. Please refer to Figure 4 for a hierarchical implementation of our $\mathcal{F}_{\mathsf{rCERT}}^{\mathbf{I}}$.

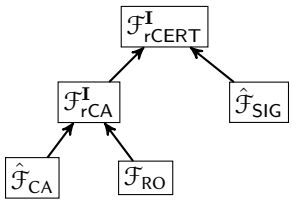


Figure 4: A hierarchical implementation of $\mathcal{F}_{\mathsf{rCERT}}^{\mathbf{I}}$.

3.2 Our core-chain protocol

We now describe the core-chain protocol Π^{core}. Each PoS-player P, initially sets his local core-chain $\mathcal{C} := \mathbf{I}$. Once activated by the environment on $(\textsc{Input-Stake}, \mathrm{P})$ at round $\mathtt{round}$, and received a core-chain set $\mathbb{C}$ from $\mathcal{F}_{\mathsf{NET}}$, the party P finds the best valid core-chain $\mathcal{C}_{\mathrm{best}}$ by running the subroutine BestCore (in Figure 6), and then updates its local core-chain $\mathcal{C} := \mathcal{C}_{\mathrm{best}}$.

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 $\mathit{context} := \langle h_\ell, \mathtt{round}_{\ell+1}\rangle$ where $\mathcal{C}[\ell]$ is the latest block-core in the longest core-chain $\mathcal{C}$, and $h_\ell := \mathsf{hash}(\mathcal{C}[\ell])$, and $\mathtt{round}_{\ell+1}$ denotes the current time. The PoS-player P may query $\mathcal{F}_{\mathsf{rCERT}}^{\mathbf{I}}$ by command $(\textsc{Elect}, \mathrm{P}, \mathit{context})$ to see if he is selected. If the PoS-player P is selected (with certain probability p), he would receive a message $(\textsc{Elected}, \mathrm{P}, \mathsf{f})$ from $\mathcal{F}_{\mathsf{rCERT}}^{\mathbf{I}}$ such that $\mathsf{f} = 1$; then the PoS-player P queries the functionality to generate a signature for $\langle h_\ell, \mathtt{round}_{\ell+1}\rangle$ via command $(\textsc{Core-Sign}, \mathrm{P}, \langle h_\ell, \mathtt{round}_{\ell+1}\rangle)$. Once receiving the signature σ from the functionality, the PoS-player P defines a

new block-core $B := \langle\langle h_\ell, \mathtt{round}_{\ell+1}\rangle, \mathrm{P}, \sigma\rangle$, updates his local core-chain $\mathcal{C}$ and then broadcasts the local core-chain to the network. Please refer to Figure 5 for more details of our core-chain protocol.

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

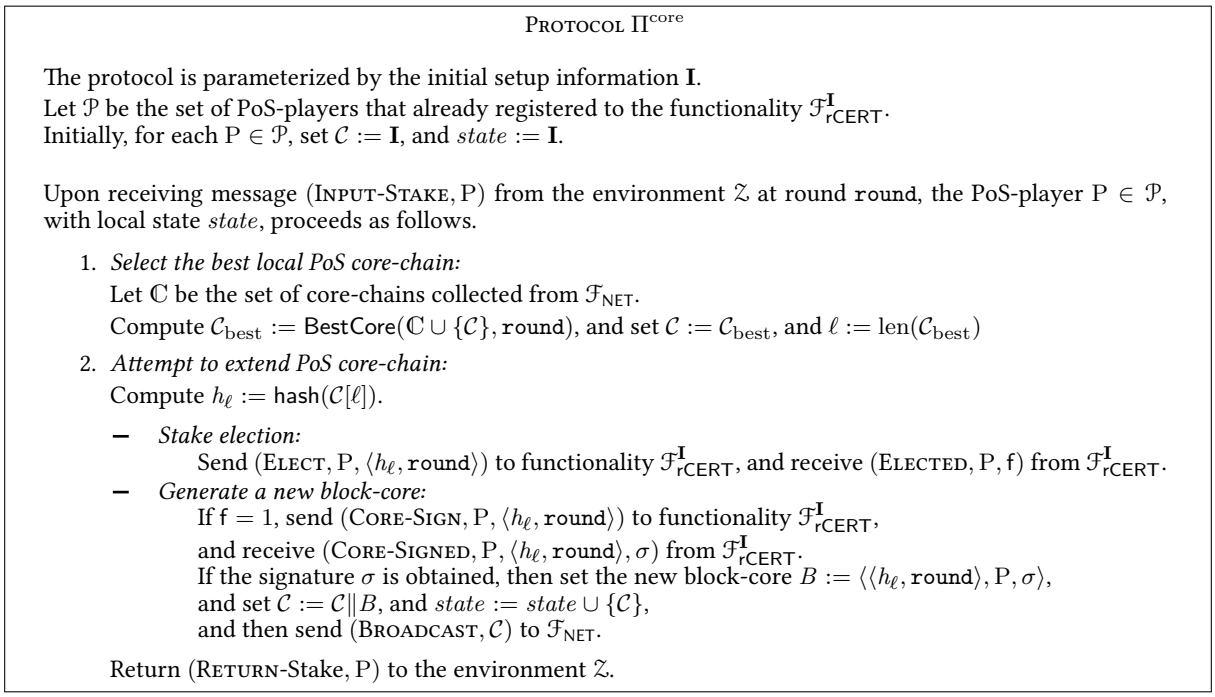


Figure 5: Our proof-of-stake core-chain protocol Π^{core} in the $\{\mathcal{F}_{\mathsf{rCERT}}^{\mathbf{I}}, \mathcal{F}_{\mathsf{NET}}\}$-hybrid model. (See Figure 6 for the subroutine BestCore.)

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 is parameterized by the initial setup information $\mathbf{I}$, and 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 signature generated by that PoS-player can be verified (by interacting with $\mathcal{F}_{\mathsf{rCERT}}^{\mathbf{I}}$). After the validation, the best valid core-chain is the longest one. Please refer to Figure 6 for more details.

4 Security analysis for core-chain

4.1 Preliminary

Our core-chain protocol Π^{core} is in the "flat, static difficulty" model in which each PoS-player holds 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

SUBROUTINE BestCore

The subroutine BestCore is parameterized by an initial setup information $\mathbf{I}$, 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 ℓ to 0, verify block-core $\mathcal{C}[i]$, as follows.
   - Parse $\mathcal{C}[i]$ into $\langle\langle h_{i-1}, \mathtt{round}_i\rangle, \mathrm{P}_i, \sigma_i\rangle$.
   - If either $i = \ell$ and $\mathtt{round}_i < \mathtt{round}'$, or $i < \ell$ and $\mathtt{round}_i < \mathtt{round}_{i+1}$ and $\mathtt{round}_{i+1} < \mathtt{round}'$, then execute:
     - If $h_{i-1} \neq \mathsf{hash}(\mathcal{C}[i-1])$, then remove this core-chain $\mathcal{C}$ from $\mathbb{C}'$.
     - Else if $h_{i-1} = \mathsf{hash}(\mathcal{C}[i-1])$, send $(\textsc{Core-Verify}, \mathrm{P}_i, \langle h_{i-1}, \mathtt{round}_i\rangle, \sigma_i)$ to $\mathcal{F}^{\mathbf{I}}_{\mathsf{rCERT}}$. Upon receiving message $(\textsc{Core-Verified}, \mathrm{P}_i, \langle h_{i-1}, \mathtt{round}_i\rangle, \sigma_i, f_i)$ from $\mathcal{F}^{\mathbf{I}}_{\mathsf{rCERT}}$, 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 6: The core-chain set validation subroutine BestCore.

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.

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.[2] Additionally, we assume that $\alpha_0 = \lambda_0\beta_0$ where $\lambda_0 \in (1, \infty)$. We are now ready to state our theorems for our core-chain protocol Π^{core}.

Theorem 4.1 (Chain growth). *Consider core-chain protocol Π^{core} in Section 3, 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*

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

where $t = r'' - r'$, $g = (1-\delta)\alpha$, and $\delta > 0$.

Theorem 4.2 (Chain quality). *Consider $\alpha = \lambda\beta$, $\lambda > 1$, and $\delta > 0$. Consider core-chain protocol Π^{core} in Section 3, and an honest PoS-player with PoS core-chain $\mathcal{C}$. Consider that ℓ consecutive block-cores of $\mathcal{C}$ are generated in s rounds, where ℓ_g block-cores are generated by honest PoS-players. Then we have*

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

where $\mu = 1 - (1+\delta)\frac{1}{\lambda}$.

Theorem 4.3 (Common prefix). *Consider $\alpha = \lambda\beta$, $\lambda > 1$, and $\delta > 0$. Consider core-chain protocol Π^{core} in Section 3. and 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 $\ell = \mathrm{len}(\mathcal{C}) - \Theta(\kappa)$.

Theorem 4.4 (Chain soundness). *Consider for every round, $\alpha = \lambda\beta$, $\lambda > 1$, and $\delta > 0$. Consider core-chain protocol Π^{core} in Section 3. and two honest PoS-players, P' and P'' in round r, with the local best PoS 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 $\mathcal{C}''[\neg\kappa] \preceq \mathcal{C}'$*

[2] In Section 6, we will discuss how to tune the target difficulty in the hash inequality in an adaptive fashion.

Before the security analysis, we introduce some terms.

Definition 4.5 (Honest successful round)**.** *We say a round r is an* honest successful *round, if in round r, at least one honest PoS-player are allowed to extend the core-chain.*

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

Definition 4.6 (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.*

4.2 Proof ideas

Before providing the proof details in the next subsections, 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 all of 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.

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 next subsection.) 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.

Chain quality. In order to reduce the core-chain quality, the best strategy for malicious parties is to generate as more 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 $1 - \frac{\beta}{\alpha}$.

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.

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.

4.3 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.

4.3.1 Hybrid expriment

To analyze the best strategy of the adversary, and the worst scenario that may happen to the honest players, we 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

1. ω is the randomness in the execution,
2. 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 $\mathsf{REAL}(\omega)$:

1. The randomness is fixed to ω as in $\mathsf{HYB}^r(\omega)$,
2. $\mathcal{F}_{\mathsf{NET}}$ delays all messages generated by *honest* PoS-players to exact Δ rounds,
3. 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,
4. Whenever some message is being delayed, no *honest* PoS-players query the functionality $\mathcal{F}^{\mathbf{I}}_{\mathsf{rCERT}}$ until the message is delivered.

In the $\mathsf{REAL}(\omega)$ executions, 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.

Lemma 4.7. *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 $\mathcal{C}'_{\mathrm{hybrid}}$ denote the PoS core-chain of P at round r' in the $\mathsf{HYB}^r(\omega)$. We then have $\mathrm{len}(\mathcal{C}') \geq \mathrm{len}(\mathcal{C}'_{\mathrm{hybrid}})$.*

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 $\mathrm{len}(\mathcal{C}) \geq \mathrm{len}(\mathcal{C}_{\mathrm{hybrid}})$. We suppose it holds for all players before round $s-1$. The only case that $\mathrm{len}(\mathcal{C}^s) < \mathrm{len}(\mathcal{C}^s_{\mathrm{hybrid}})$ is the player P received a new core-chain to extend $\mathcal{C}^s_{\mathrm{hybrid}}$ 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 $\mathrm{len}(\mathcal{C}^s_{\mathrm{hybrid}}) = \mathrm{len}(\mathcal{C}^{s-\Delta}_{\mathrm{hybrid}}) + 1$. 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 $\mathrm{len}(\mathcal{C}') \geq \mathrm{len}(\mathcal{C}'^{\Delta})$. □

4.3.2 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 $\mathsf{HYB}^r(\omega)$.

Lemma 4.8. *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 $\mathcal{F}^{\mathbf{I}}_{\mathsf{rCERT}}$ 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 $\mathtt{VIEW}^r$ denote the $\mathtt{VIEW}$ at round r in $\mathsf{REAL}(\omega)$ where $r > 0$. Let $\mathrm{len}(\mathtt{VIEW}^r)$ denote the length of the best public PoS core-chain in $\mathtt{VIEW}^r$. 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 $\mathsf{HYB}^r(\omega)$.

Lemma 4.9. *Consider* $\mathsf{HYB}^r(\omega)$. *For any honest successful round* s, *where* $s > r$, *it holds that*

$$\mathrm{len}(\mathtt{VIEW}^{s+\Delta}) - \mathrm{len}(\mathtt{VIEW}^s) \geq 1$$

Proof. By Definition 4.5, there is at least one honest PoS-player producing a PoS block-core at round s. Let $\mathcal{C}^s_{\mathrm{hybrid}}$ be the PoS core-chain that is extended by the PoS-player at round s. We have $\mathrm{len}(\mathcal{C}^s_{\mathrm{hybrid}}) \geq \mathrm{len}(\mathtt{VIEW}^s)$. At the end of round s the honest player will broadcast the extended chain with length $\mathrm{len}(\mathcal{C}^s_{\mathrm{hybrid}}) + 1$. At the end of round $s + \Delta$, all honest players will receive the extended core-chain, we have $\mathrm{len}(\mathtt{VIEW}^{s+\Delta}) \geq \mathrm{len}(\mathcal{C}^{s+\Delta}_{\mathrm{hybrid}}) = \mathrm{len}(\mathcal{C}^s_{\mathrm{hybrid}}) + 1$. Putting them together, we have $\mathrm{len}(\mathtt{VIEW}^{s+\Delta}) - \mathrm{len}(\mathtt{VIEW}^s) \geq 1$.

□

Corollary 4.10. *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*

$$\mathrm{len}(\mathtt{VIEW}^{r+t+\Delta}) - \mathrm{len}(\mathtt{VIEW}^r) \geq h$$

Proof. Let r_k be the kth honest successful round where $r < \mathtt{round}_k < r+t$ and $1 \leq k \leq h$. From Lemma 4.9, we have $\mathrm{len}(\mathtt{VIEW}^{\mathtt{round}_k+\Delta}) - \mathrm{len}(\mathtt{VIEW}^{\mathtt{round}_k}) \geq 1$. Then we have $\mathrm{len}(\mathtt{VIEW}^{r+t}) - \mathrm{len}(\mathtt{VIEW}^r) \geq \sum_{i=1}^{h}(\mathrm{len}(\mathtt{VIEW}^{\mathtt{round}_k+\Delta}) - \mathrm{len}(\mathtt{VIEW}^{\mathtt{round}_k})) \geq h$. □

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

4.4 Achieving chain growth property

We here demonstrate that our core-chain protocol satisfies the growth property (Definition 2.2). The concrete statement to be proved can be found in Theorem 4.1. We next first develop some useful lemmas.

Lemma 4.11. *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 Lemma 4.8, 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 $\Pr[X > (1 - \delta)\alpha t] > 1 - e^{-\delta^2\alpha t/2} = 1 - e^{-\Omega(t)}$. □

Lemma 4.12. *Consider* $\mathsf{HYB}^{r}(\omega)$ *and* $\delta > 0$. *Consider an honest PoS-player* P *with the best PoS core-chain* $\mathcal{C}_{\mathrm{hybrid}}$ *in round* r, *and an honest PoS-player* P' *with the best PoS core-chain* $\mathcal{C}'_{\mathrm{hybrid}}$ *in round* r', *respectively, where* $r' - r \gg \Delta$. *Then we have*

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

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

Proof. First, we note that $\mathcal{C}_{\mathrm{hybrid}}$ will be received by all honest players no later than round $r + \Delta$ because player P is honest. We have $\mathrm{len}(\mathcal{C}_{\mathrm{hybrid}}) \leq \mathrm{len}(\mathtt{VIEW}^{r+\Delta})$. 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 Lemma 4.11, 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 Lemma 4.9 and Corollary 4.10, we have $\mathrm{len}(\mathtt{VIEW}^{r'}) - \mathrm{len}(\mathtt{VIEW}^{r+\Delta}) \geq (1-\delta)\alpha t$. Chain $\mathcal{C}'_{\mathrm{hybrid}}$ is an valid PoS core-chain accepted by an honest PoS-player P' at round r'. We have $\mathrm{len}(\mathcal{C}'_{\mathrm{hybrid}}) \geq \mathrm{len}(\mathtt{VIEW}^{r'})$. Put them together, $\mathrm{len}(\mathcal{C}'_{\mathrm{hybrid}}) - \mathrm{len}(\mathcal{C}_{\mathrm{hybrid}}) \geq \mathrm{len}(\mathtt{VIEW}^{r'}) - \mathrm{len}(\mathtt{VIEW}^{r+\Delta}) \geq (1-\delta)\alpha t$ with probability at least $1 - e^{-\Omega(t)}$. The corresponding growth rate is $g = (1-\delta)\alpha$. □

Reminder of Theorem 4.1. *Consider core-chain protocol* Π^{core} *in Section 3, 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*

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

where $t = r'' - r'$, $g = (1-\delta)\alpha$, *and* $\delta > 0$.

Proof. In order to distinguish the notation clearly, we use $\mathcal{C}'_{\mathrm{hybrid}}$ and $\mathcal{C}''_{\mathrm{hybrid}}$ 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 Lemma 4.12, we have $\Pr[\mathrm{len}(\mathcal{C}''_{\mathrm{hybrid}}) \geq \mathrm{len}(\mathcal{C}'^{)}_{\mathrm{hybrid}} + g \cdot t] \geq 1 - e^{-\Omega(t)}$ where $t = r'' - r'$, in $\mathsf{HYB}^{r}(\omega)$. We now turn to the core-chain growth property in $\mathsf{EXEC}_{\Pi^{\mathrm{core}},\mathcal{A},\mathcal{Z}}$. From the definition of hybrid execution, we know that all honest players have same initial status at round r'. We have $\mathrm{len}(\mathcal{C}') = \mathrm{len}(\mathcal{C}'_{\mathrm{hybrid}})$. By Lemma 4.7, we have $\mathrm{len}(\mathcal{C}'') \geq \mathrm{len}(\mathcal{C}''_{\mathrm{hybrid}})$. It follows that,

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

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

4.5 Achieving chain quality property

The chain-quality property (Definition 2.4) 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.

Lemma 4.13. *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.

Lemma 4.14. *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 Theorem 4.1, 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 Theorem 4.2. *Consider* $\alpha = \lambda\beta$*,* $\lambda > 1$*, and* $\delta > 0$*. Consider core-chain protocol* Π^{core} *in Section 3, and an honest PoS-player with PoS core-chain* $\mathcal{C}$*. Consider that* ℓ *consecutive block-cores of* $\mathcal{C}$ *are generated in* s *rounds, where* ℓ_g *block-cores are generated by honest PoS-players. Then we have*

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

where $\mu = 1 - (1+\delta)\frac{1}{\lambda}$*.*

Proof. Let t be the rounds that the ℓ block-cores are generated. From Lemma 4.13, we have $\Pr[t > (1-\delta)c\ell] > 1 - e^{-\Omega(\ell)}$. From Lemma 4.14, 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. □

4.6 Achieving common prefix property

We now turn to proving the common prefix property (Definition 2.3) for the core-chain protocol Π^{core}. The concrete statement can be found in Theorem 4.3. 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 4.15 (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 $\ell = max\{\ell', \ell''\}$.*

Lemma 4.16. *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 $\Pr[\mathrm{len}(\mathcal{C}) - \mathrm{len}(\mathcal{C}') > (1-\delta)\ell] > 1 - e^{-\Omega(\ell)}$.*

Proof. Suppose the last common block-core of $\mathcal{C}$ and $\mathcal{C}'$ is generated in round $s = r - t$. With Lemma 4.13, we have $t > (1-\delta)\frac{\ell}{\alpha+\beta}$ with probability no less than $1 - e^{-\Omega(\ell)}$. Let $X = \mathrm{len}(\mathcal{C}) - \mathrm{len}(\mathcal{C}^s)$ be the length growth of best public core-chain in the t rounds, with Theorem 4.1, 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 $1 - e^{-\Omega(t)}$. Put them 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 $\Pr[\mathrm{len}(\mathcal{C}) - \mathrm{len}(\mathcal{C}')] = \Pr[X - (Y+Z) > (1-\delta)\ell] > 1 - e^{-\Omega(\ell)}$. This completes the proof. □

Lemma 4.17. *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. With Lemma 4.16, 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 Theorem 4.3. *Consider $\alpha = \lambda\beta$, $\lambda > 1$, and $\delta > 0$. Consider core-chain protocol Π^{core} in Section 3. and 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 $\ell = \mathrm{len}(\mathcal{C}) - \Theta(\kappa)$.

Proof. Let $\mathcal{C}^r_{\mathrm{public}}$ be the best public core-chain in round r. For $\mathcal{C}$ is accepted by a player it is must be better than $\mathcal{C}^r_{\mathrm{public}}$. Let ℓ be the divergent length of $\mathcal{C}$ and $\mathcal{C}^r_{\mathrm{public}}$, from Lemma 4.16 we have $\ell < \kappa$. Otherwise, $\mathcal{C}$ is better than $\mathcal{C}^r_{\mathrm{public}}$ 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 $\mathcal{C}^r_{\mathrm{public}}$ and $\mathcal{C}'^r$. From Lemma 4.17, we have $\ell < \kappa$. Otherwise, $\mathcal{C}'$ is better than public core-chain in round r' with a low

probability. Put them together, 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. □

4.7 Achieving chain soudness property

We now turn to proving the chain soudnness property (Definition 2.5) for the core-chain protocol Π^{core}. The concrete statement can be found in Theorem 4.4. Before providing our formal proof, we here give some informal proof ideas. Our best chain strategy is longest chain as in PoW blockchain. As we discussed above, the malicious players can not create a chain which grows faster than the best public chain. For a new spawn player, the malicious players can not mislead him by provide a longer chain.

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

Reminder of Theorem 4.4. *Consider for every round, $\alpha = \lambda\beta$, $\lambda > 1$, and $\delta > 0$. Consider core-chain protocol Π^{core} in Section 3. and two honest PoS-players, P' and P'' in round r, with the local best PoS 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 imply both P' and P'' have received $\mathcal{C}$. Let ℓ' be the divergent length of $\mathcal{C}'$ and $\mathcal{C}$. From the Lemma 4.16, 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 contradict that P_i take $\mathcal{C}'$ as the best chain. We get $\mathcal{C}'[\neg\kappa] \preceq \mathcal{C}$. Similarly, we get $\mathcal{C}''[\neg\kappa] \preceq \mathcal{C}$. Furthermore, because $\mathrm{len}(\mathcal{C}') \geq \mathrm{len}(\mathcal{C})$, we have the divergent length of $\mathcal{C}$ is shorter than $\mathcal{C}'$. We get $\mathcal{C}[\neg\kappa] \preceq \mathcal{C}'$. Similarly, we get $\mathcal{C}[\neg\kappa] \preceq \mathcal{C}''$. Put them together, we get $\mathcal{C}'[\neg\kappa] \preceq \mathcal{C}''$ and $\mathcal{C}''[\neg\kappa] \preceq \mathcal{C}'$. □

5 From core-chain to blockchain

In this section, we start to extend the core-chain protocol in Section 3 to a blockchain protocol in which payload (transactions) will be included. 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 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 blockchain protocol), then the PoS-player is selected for generating the new block $\tilde{B}_{i+1}$, in the following format

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

where $\tilde{\sigma} \leftarrow \mathsf{Sign}_{\tilde{\mathrm{SK}}}(\tilde{h}_i, B_{i+1}, \tilde{X}_i)$ and $\tilde{h}_i := \mathsf{hash}(\tilde{B}_i)$, and $B_{i+1} := \langle h_i, \mathtt{round}, \mathrm{PK}, \sigma \rangle$. Here we note that in our blockchain protocol design, the PoS-player holds two pairs of keys, $(\mathrm{SK}, \mathrm{PK})$ of the strengthened unique signature scheme $(\mathsf{uKeyGen}, \mathsf{uSign}, \mathsf{uVerify})$, and $(\tilde{\mathrm{SK}}, \tilde{\mathrm{PK}})$ of a regular digital signature scheme $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$. Here we note that in our blockchain protocol design, the PoS-player holds two pairs of keys, $(\mathrm{SK}, \mathrm{PK})$ of the strengthened unique signature scheme $(\mathsf{uKeyGen}, \mathsf{uSign}, \mathsf{uVerify})$, and $(\tilde{\mathrm{SK}}, \tilde{\mathrm{PK}})$ 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 the security of the blockchain protocol.

In the formal description of our blockchain protocol below, we will use a slightly augmented setup functionality $\tilde{\mathcal{F}}^{\mathbf{I}}_{\mathsf{rCERT}}$ 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 schemes.

5.1 Setup functionality $\tilde{\mathcal{F}}^{\mathbf{I}}_{\mathsf{rCERT}}$

In our blockchain protocol design, we will use the setup functionality, $\tilde{\mathcal{F}}^{\mathbf{I}}_{\mathsf{rCERT}}$ (in Figure 7), which is an augmented version of the resource certification functionality in Section 3.1. The first two phases, "Stake Resource Registration" and "Stake Election", are the same. But the remaining two phases, "Signature Generation" and "Stake Verification", have been extended for both *context* and *msg* signing and verification. (Note that, in $\mathcal{F}^{\mathbf{I}}_{\mathsf{rCERT}}$, only *context* signing and verification can be supported.)

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

The functionality is parameterized by a difficulty parameter p, a security parameter κ, initial setup information $\mathbf{I}$, and interacts with a set $\mathcal{P}$ of parties, as well as an adversary.

Stake Resource Registration.

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

For each round, set $\phi_{\mathrm{P}} := 0$ for every registered party $\mathrm{P} \in \mathcal{P}$, then proceed as follows.

Stake Election: Upon receiving (ELECT, P, *context*) from a PoS-player P, proceed as follows.

1. If $(\mathrm{P}, \mathbb{b}_{\mathrm{P}})$ is recorded where $\mathbb{b}_{\mathrm{P}} = 1$ and $\phi_{\mathrm{P}} = 0$, (the party P registered and granted one unit of stake)
   - with probability p, set $\phi_{\mathrm{P}} := 1$ and $\mathsf{f} := 1$, send $(\text{ELECTED}, \mathrm{P}, \mathsf{f})$ to P, and record the entry $(\mathrm{P}, \mathit{context})$ (the party P is elected.)
   - with probability $1 - p$, set $\phi_{\mathrm{P}} := 1$ and $\mathsf{f} := 0$, and send $(\text{ELECTED}, \mathrm{P}, \mathsf{f})$ to P (the party P is not elected.)
2. Otherwise, if any of the following cases occur:
   - $(\mathrm{P}, \mathbb{b}_{\mathrm{P}})$ is not recorded, (the party P is not registered yet)
   - or $(\mathrm{P}, \mathbb{b}_{\mathrm{P}})$ is recorded and $\mathbb{b}_{\mathrm{P}} = 0$ (the party P registered and then unregistered),
   - or $\phi_{\mathrm{P}} = 1$, (the party P already used the granted stake unit)

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

Signature Generation:
Upon receiving (CORE-SIGN, P, *context*) from a party P, send (CORE-SIGN, P, *context*) to the adversary.
Upon receiving (SIGNATURE, P, *context*, σ) from the adversary, verify that no entry $(\mathrm{P}, \mathit{context}, \sigma, 0)$ is recorded. If it is, then output an error message (ERROR) to P and halt. Else, output (CORE-SIGNED, P, *context*, σ) to P, and record the entry $(\mathrm{P}, \mathit{context}, \sigma, 1)$.

Upon receiving (BLOCK-SIGN, P, *msg*) from a party P, send (BLOCK-SIGN, P, *msg*) to the adversary.
Upon receiving (SIGNATURE, P, *msg*, $\tilde{\sigma}$) from the adversary, verify that no entry $(\mathrm{P}, \mathit{msg}, \tilde{\sigma}, 0)$ is recorded. If it is, then output an error message (ERROR) to P and halt. Else, output (BLOCK-SIGNED, P, *msg*, $\tilde{\sigma}$) to P, and record the entry $(\mathrm{P}, \mathit{msg}, \tilde{\sigma}, 1)$.

Stake Verification:
Upon receiving (CORE-VERIFY, P, *context*, σ) 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 (CORE-VERIFY, P, *context*, σ) to the adversary. Upon receiving (CORE-VERIFIED, P, *context*, ϕ) from the adversary, do:
   - If $(\mathrm{P}, \mathit{context}, \sigma, 1)$ is recorded, then set $f := 1$.
   - Else, if P is not corrupted, and no entry $(\mathrm{P}, \mathit{context}, \sigma', 1)$ for any σ' is recorded, then set $f := 0$ and record the entry $(\mathrm{P}, \mathit{context}, \sigma, f)$.
   - Else, if there is an entry $(\mathrm{P}, \mathit{context}, \sigma, f')$, then set $f := f'$.
   - Else, set $f := \phi$, and record the entry $(\mathrm{P}, \mathit{context}, \sigma, f)$.

   Output (CORE-VERIFIED, P, *context*, 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 (CORE-VERIFIED, P, *context*, f) to the party P'.

Upon receiving (BLOCK-VERIFY, P, *msg*, $\tilde{\sigma}$) from a party $\mathrm{P}' \in \mathcal{P}$,

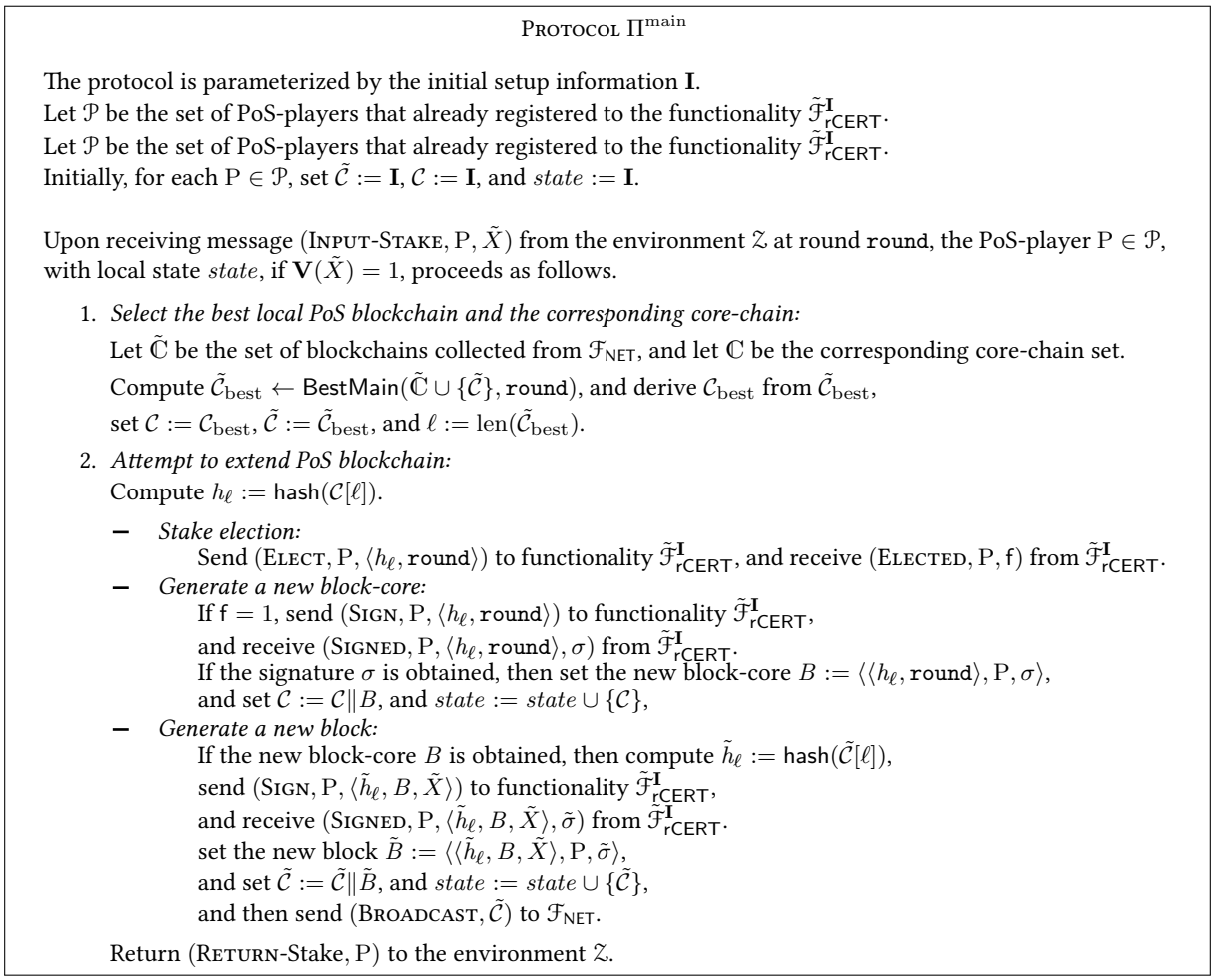
1. If there exists a record of the form $(\mathrm{P}, \cdot)$, (the party P is elected) then send (CORE-VERIFY, P, *msg*, $\tilde{\sigma}$) to the adversary. Upon receiving (BLOCK-VERIFIED, P, *msg*, $\tilde{\phi}$) from the adversary, do:
   - If $(\mathrm{P}, \mathit{msg}, \tilde{\sigma}, 1)$ is recorded,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

   Output (BLOCK-VERIFIED, 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 (BLOCK-VERIFIED, P, *msg*, f) to the party P'.

Figure 7: Augmented resource certification functionality $\tilde{\mathcal{F}}^{\mathbf{I}}_{\mathsf{rCERT}}$.

5.2 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 3, and now it uses the augmented resource certificate functionality $\tilde{\mathcal{F}}_{\mathsf{rCERT}}^{\mathbf{I}}$ as setup functionality.



PROTOCOL Π^{main}

The protocol is parameterized by the initial setup information $\mathbf{I}$.
Let $\mathcal{P}$ be the set of PoS-players that already registered to the functionality $\tilde{\mathcal{F}}_{\mathsf{rCERT}}^{\mathbf{I}}$.
Let $\mathcal{P}$ be the set of PoS-players that already registered to the functionality $\tilde{\mathcal{F}}_{\mathsf{rCERT}}^{\mathbf{I}}$.
Initially, for each $\mathrm{P} \in \mathcal{P}$, set $\tilde{\mathcal{C}} := \mathbf{I}$, $\mathcal{C} := \mathbf{I}$, and $state := \mathbf{I}$.

Upon receiving message $(\text{INPUT-STAKE}, \mathrm{P}, \tilde{X})$ from the environment $\mathcal{Z}$ at round $\mathtt{round}$, the PoS-player $\mathrm{P} \in \mathcal{P}$, with local state $state$, if $\mathbf{V}(\tilde{X}) = 1$, proceeds as follows.

1. *Select the best local PoS blockchain and the corresponding core-chain:*
   Let $\tilde{\mathbb{C}}$ be the set of blockchains collected from $\mathcal{F}_{\mathsf{NET}}$, and let $\mathbb{C}$ be the corresponding core-chain set.
   Compute $\tilde{\mathcal{C}}_{\text{best}} \leftarrow \mathsf{BestMain}(\tilde{\mathbb{C}} \cup \{\tilde{\mathcal{C}}\}, \mathtt{round})$, and derive $\mathcal{C}_{\text{best}}$ from $\tilde{\mathcal{C}}_{\text{best}}$,
   set $\mathcal{C} := \mathcal{C}_{\text{best}}$, $\tilde{\mathcal{C}} := \tilde{\mathcal{C}}_{\text{best}}$, and $\ell := \text{len}(\tilde{\mathcal{C}}_{\text{best}})$.
2. *Attempt to extend PoS blockchain:*
   Compute $h_\ell := \mathsf{hash}(\mathcal{C}[\ell])$.
   - *Stake election:*
     Send $(\text{ELECT}, \mathrm{P}, \langle h_\ell, \mathtt{round}\rangle)$ to functionality $\tilde{\mathcal{F}}_{\mathsf{rCERT}}^{\mathbf{I}}$, and receive $(\text{ELECTED}, \mathrm{P}, \mathsf{f})$ from $\tilde{\mathcal{F}}_{\mathsf{rCERT}}^{\mathbf{I}}$.
   - *Generate a new block-core:*
     If $\mathsf{f} = 1$, send $(\text{SIGN}, \mathrm{P}, \langle h_\ell, \mathtt{round}\rangle)$ to functionality $\tilde{\mathcal{F}}_{\mathsf{rCERT}}^{\mathbf{I}}$,
     and receive $(\text{SIGNED}, \mathrm{P}, \langle h_\ell, \mathtt{round}\rangle, \sigma)$ from $\tilde{\mathcal{F}}_{\mathsf{rCERT}}^{\mathbf{I}}$.
     If the signature σ is obtained, then set the new block-core $B := \langle\langle h_\ell, \mathtt{round}\rangle, \mathrm{P}, \sigma\rangle$,
     and set $\mathcal{C} := \mathcal{C} \| B$, and $state := state \cup \{\mathcal{C}\}$,
   - *Generate a new block:*
     If the new block-core B is obtained, then compute $\tilde{h}_\ell := \mathsf{hash}(\tilde{\mathcal{C}}[\ell])$,
     send $(\text{SIGN}, \mathrm{P}, \langle \tilde{h}_\ell, B, \tilde{X}\rangle)$ to functionality $\tilde{\mathcal{F}}_{\mathsf{rCERT}}^{\mathbf{I}}$,
     and receive $(\text{SIGNED}, \mathrm{P}, \langle \tilde{h}_\ell, B, \tilde{X}\rangle, \tilde{\sigma})$ from $\tilde{\mathcal{F}}_{\mathsf{rCERT}}^{\mathbf{I}}$.
     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 $(\text{BROADCAST}, \tilde{\mathcal{C}})$ to $\mathcal{F}_{\mathsf{NET}}$.

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

Figure 8: Our proof-of-stake blockchain protocol Π^{main} in the $\{\tilde{\mathcal{F}}_{\mathsf{rCERT}}^{\mathbf{I}}, \mathcal{F}_{\mathsf{NET}}\}$-hybrid model. (See Figure 9 for the subroutine BestMain.)

As in the core-chain protocol, each PoS-player P, initially sets his local blockchain $\tilde{\mathcal{C}} := \mathbf{I}$. Once activated by the environment on $(\text{INPUT-STAKE}, \mathrm{P})$ at round $\mathtt{round}$, 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 BestMain (in Figure 9), and then updates its local blockchain $\tilde{\mathcal{C}} := \tilde{\mathcal{C}}_{\text{best}}$. Note that, the i-the 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 $\tilde{\mathcal{C}}$.

Then each registered PoS-player P attempt to extend the core-chain. Let ℓ be the length of blockchain $\tilde{\mathcal{C}}$. (The core-chain $\mathcal{C}$ is also with length ℓ.) In our design, only the elected PoS-players are allowed to generate new block-cores (to extend the blockchain). The PoS-player P queries $\tilde{\mathcal{F}}_{\mathsf{rCERT}}^{\mathbf{I}}$ by command $(\text{ELECT}, \mathrm{P}, context)$, where $context := \langle h_\ell, \mathtt{round}_{\ell+1}\rangle$, to see if he is selected; note that here $h_\ell := \mathsf{hash}(\mathcal{C}[\ell])$, and $\mathcal{C}[\ell]$ is the newest block-core in the best core-chain.

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

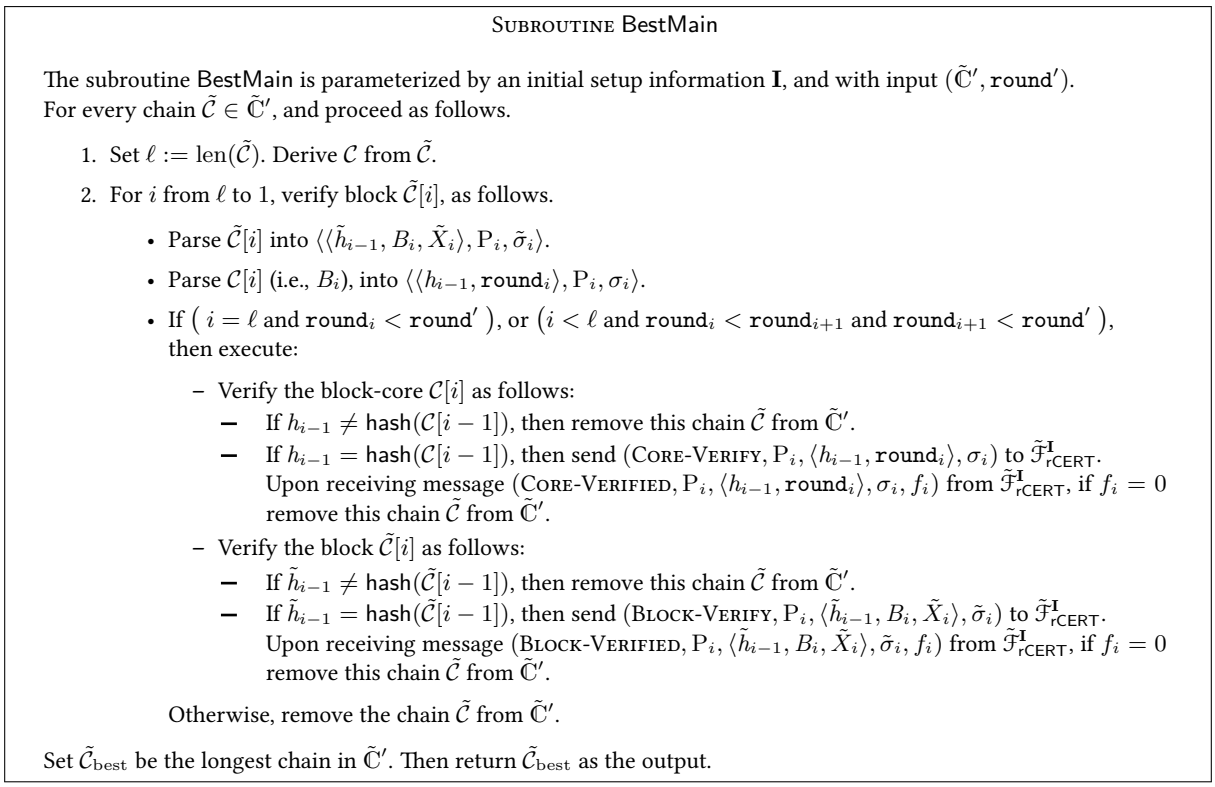
core-chain $\mathcal{C}$.

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 8 for more details of our blockchain protocol.

The best blockchain strategy. In Section 3, 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. Here we describe a similar strategy, subroutine BestMain, to single out the best blockchain from a set of blockchains. The subroutine BestMain here is a slightly augmented version of the subroutine BestCore in our core-chain protocol.

Intuitively, a blockchain is the best one if it is the *current longest valid* blockchain. The BestMain subroutine is parameterized by the initial setup information $\mathbf{I}$, and 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, 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 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 $\tilde{\mathcal{F}}^{\mathbf{I}}_{\mathsf{rCERT}}$). Then for every block-core $\tilde{\mathcal{C}}[i]$, for all $i \in [\ell]$, in the blockchain $\tilde{\mathcal{C}}$, the 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 $\tilde{\mathcal{F}}^{\mathbf{I}}_{\mathsf{rCERT}}$). After the validation, the best valid blockchain is the longest one. Please refer to Figure 9 for more details.



SUBROUTINE BestMain

The subroutine BestMain is parameterized by an initial setup information $\mathbf{I}$, 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. For i from ℓ to 1, verify block $\tilde{\mathcal{C}}[i]$, as follows.
   - Parse $\tilde{\mathcal{C}}[i]$ into $\langle \langle \tilde{h}_{i-1}, B_i, \tilde{X}_i \rangle, \mathrm{P}_i, \tilde{\sigma}_i \rangle$.
   - Parse $\mathcal{C}[i]$ (i.e., B_i), into $\langle \langle h_{i-1}, \mathtt{round}_i \rangle, \mathrm{P}_i, \sigma_i \rangle$.
   - If $\big(i = \ell$ and $\mathtt{round}_i < \mathtt{round}' \big)$, or $\big(i < \ell$ and $\mathtt{round}_i < \mathtt{round}_{i+1}$ and $\mathtt{round}_{i+1} < \mathtt{round}' \big)$, then execute:
     - Verify the block-core $\mathcal{C}[i]$ as follows:
       - If $h_{i-1} \neq \mathsf{hash}(\mathcal{C}[i-1])$, then remove this chain $\tilde{\mathcal{C}}$ from $\tilde{\mathbb{C}}'$.
       - If $h_{i-1} = \mathsf{hash}(\mathcal{C}[i-1])$, then send $(\textsc{Core-Verify}, \mathrm{P}_i, \langle h_{i-1}, \mathtt{round}_i \rangle, \sigma_i)$ to $\tilde{\mathcal{F}}^{\mathbf{I}}_{\mathsf{rCERT}}$. Upon receiving message $(\textsc{Core-Verified}, \mathrm{P}_i, \langle h_{i-1}, \mathtt{round}_i \rangle, \sigma_i, f_i)$ from $\tilde{\mathcal{F}}^{\mathbf{I}}_{\mathsf{rCERT}}$, if $f_i = 0$ remove this chain $\tilde{\mathcal{C}}$ from $\tilde{\mathbb{C}}'$.
     - Verify the block $\tilde{\mathcal{C}}[i]$ as follows:
       - If $\tilde{h}_{i-1} \neq \mathsf{hash}(\tilde{\mathcal{C}}[i-1])$, then remove this chain $\tilde{\mathcal{C}}$ from $\tilde{\mathbb{C}}'$.
       - If $\tilde{h}_{i-1} = \mathsf{hash}(\tilde{\mathcal{C}}[i-1])$, then send $(\textsc{Block-Verify}, \mathrm{P}_i, \langle \tilde{h}_{i-1}, B_i, \tilde{X}_i \rangle, \tilde{\sigma}_i)$ to $\tilde{\mathcal{F}}^{\mathbf{I}}_{\mathsf{rCERT}}$. Upon receiving message $(\textsc{Block-Verified}, \mathrm{P}_i, \langle \tilde{h}_{i-1}, B_i, \tilde{X}_i \rangle, \tilde{\sigma}_i, f_i)$ from $\tilde{\mathcal{F}}^{\mathbf{I}}_{\mathsf{rCERT}}$, if $f_i = 0$ remove this chain $\tilde{\mathcal{C}}$ from $\tilde{\mathbb{C}}'$.

     Otherwise, remove the chain $\tilde{\mathcal{C}}$ from $\tilde{\mathbb{C}}'$.

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

Figure 9: The chain set validation subroutine BestMain.

5.3 Analysis of blockchain protocol

Here we provide security analysis for our blockchain protocol. As mentioned before, our blockchain protocol can be viewed as an augmented version of our core-chain protocol in Section 3; each security property of our blockchain protocol can be reduced to the corresponding property of the core-chain protocol.

Chain growth. The proof idea for achieving chain growth property is very clear. 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 5.1 (Chain growth)**.** *Consider the blockchain protocol Π^{main}. Consider $\alpha = \lambda\beta$, $\lambda > 1$, 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*

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

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

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, $\text{len}(\tilde{\mathcal{C}}') = \text{len}(\mathcal{C}')$ and $\text{len}(\tilde{\mathcal{C}}) = \text{len}(\mathcal{C})$. That means, $\text{len}(\tilde{\mathcal{C}}') - \text{len}(\tilde{\mathcal{C}}) = \text{len}(\mathcal{C}') - \text{len}(\mathcal{C})$. From the Theorem 4.1, we have

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

This completes the proof. □

Chain quality. Similarly, the proof idea for achieving chain quality property is very clear. 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 5.2 (Chain quality)**.** *Consider the blockchain protocol Π^{main}. Consider $\alpha = \lambda\beta$, $\lambda > 1$, 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_g blocks are generated by honest PoS-players. Then we have*

$$\Pr[\frac{\ell_g}{\ell} \geq \mu] \geq 1 - e^{-\Omega(\ell)}$$

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 ℓ_g^{core} be the number of block-cores from honest stakeholders on core-chain $\mathcal{C}$. Let ℓ_g^{main} 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 $\ell^{\text{core}} = \ell_g^{\text{main}} = \ell_g$. We also have that $\text{len}(\mathcal{C}) = \text{len}(\tilde{\mathcal{C}}) = \ell$. From the Theorem 4.2 we have $\Pr[\frac{\ell_g}{\ell} \geq \mu] \geq 1 - e^{-\Omega(\ell)}$, 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 5.3 (Common prefix). *Consider the blockchain protocol* Π^{main}. *Consider* $\alpha = \lambda\beta$, $\lambda > 1$, *and* $\delta > 0$. *and 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 $\ell = \mathrm{len}(\mathcal{C}) - \Theta(\kappa)$.

Proof. As we discussed, $\tilde{\mathcal{C}}$ and $\tilde{\mathcal{C}}'$ are associated with core-chains $\mathcal{C}$ and $\mathcal{C}'$ respectively. From the Theorem 4.3 we know that $\Pr[\mathcal{C}[1,\ell] \preceq \mathcal{C}'] \geq 1 - e^{-\Omega(\kappa)}$.

Based on the assumption that $\alpha = \lambda\beta$ 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 $\ell = \mathrm{len}(\mathcal{C}) - \Theta(\kappa)$. □

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 5.4 (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 the Theorem 4.4 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 $\tilde{\mathcal{C}}''[\neg\kappa] \preceq \tilde{\mathcal{C}}'$. □

6 Extensions and Discussions

In order to make our design practical, additional mechanisms are needed. As mentioned before, our design is a natural mimic of Nakamoto's but via proof-of-stake. This is a unique feature: we can easily "borrow" many of Nakamoto's ideas (and also follow-up ideas), to our design. In this section, we mention a few of them.

Blockchain with adaptive difficulty. In Bitcoin, in order to keep 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 the actual time span of the i-th epoch, and T_i be the target difficulty in the i-th epoch. We have the target difficulty in the $(i+1)$-th epoch as follows:

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

From the equation above we can observe that, if $t_i > \mathsf{t}$ then $\mathsf{T}_{i+1} > \mathsf{T}_i$ 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 $\mathsf{H}(\mathsf{hash}(B_i), \mathtt{round}, \mathrm{PK}, \sigma) < \mathsf{T}_i$. A player will test if he is qualified to sign a PoS-block based on the current target difficulty T_i.

Blockchain in the non-flat model. Our ideas in Section 3 and in Section 5 are described in the "flat" model, where all PoS-players are assumed to hold the same amount of stake (and they are selected as the wining 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 (PK, 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 the follows:

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

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.

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., [PS16]) 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. In stead of straightforwardly including the entire "payload" $\tilde{X}_i$ in the block $\tilde{B}_i$ (as in Section 5, and in [GKL14, PSS17]), we can store a Merkle root in $\tilde{B}_i$. New ideas (e.g., [RMCI16]) can also be used in our design.

Acknowledgement: We thank Tuyet Duong and Jonathan Katz for helpful discussions about scalable blockchain protocols in the open network setting.

References

[Bac02] Adam Back. Hashcash — A denial of service counter-measure. 2002. http://hashcash.org/papers/hashcash.pdf.

[BGM16] Iddo Bentov, Ariel Gabizon, and Alex Mizrahi. Currencies without proof of work. In *Bitcoin Workshop*, 2016.

[Bit11] 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.

[BLMR14] Iddo Bentov, Charles Lee, Alex Mizrahi, and Meni Rosenfeld. Proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract]. *SIGMETRICS Perform. Eval. Rev.*, 42(3):34–37, December 2014.

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Heidelberg, December 2001.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, CCS '93, pages 62–73. ACM, 1993.

[But14] Vitalik Buterin. A next-generation smart contract and decentralized application platform. 2014. https://github.com/ethereum/wiki/wiki/White-Paper.

[But15] Vitalik Buterin. Understanding serenity, part 2: Casper. 2015. https://blog.ethereum.org/2015/12/28/understanding-serenity-part-2-casper/.

[Can00a] Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[Can00b] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000. http://eprint.iacr.org/2000/067.

[Can03] Ran Canetti. Universally composable signatures, certification and authentication. Cryptology ePrint Archive, Report 2003/239, 2003. http://eprint.iacr.org/2003/239.

[CDFZ17] Alexander Chepurnoy, Tuyet Duong, Lei Fan, and Hong-Sheng 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.

[Cha82] David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *CRYPTO'82*, pages 199–203. Plenum Press, New York, USA, 1982.

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

[CR03] Ran Canetti and Tal Rabin. Universal composition with joint state. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 265–281. Springer, Heidelberg, August 2003.

[Cry14] CryptoManiac. Proof of stake. *NovaCoin wiki*, 2014. https://github.com/novacoin-project/novacoin/wiki/Proof-of-stake.

[DFZ16] Tuyet Duong, Lei Fan, and Hong-Sheng Zhou. 2-hop blockchain: Combining proof-of-work and proof-of-stake securely. In *Cryptology ePrint Archive, Report 2016/716*, 2016. https://eprint.iacr.org/2016/716.

[DFZ17] Tuyet Duong, Lei Fan, and Hong-Sheng Zhou. 2-hop blockchain: Combining proof-of-work and proof-of-stake securely. 2017. Manuscript.

[DHLW10] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Cryptography against continuous memory attacks. In *51st FOCS*, pages 511–520. IEEE Computer Society Press, October 2010.

[DN93] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 139–147. Springer, Heidelberg, August 1993.

[DPS17] Phil Daian, Rafael Pass, and Elaine Shi. Snow white: Robustly reconfigurable consensus and applications to provably secure proofs of stake. In *Cryptology ePrint Archive, Report 2016/919*, April 2017. http://eprint.iacr.org/2016/919.

[DZ17] Tuyet Duong and Hong-Sheng Zhou. A note on proof-of-work blockchains in the open setting. 2017. Manuscript.

[ES14] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In Nicolas Christin and Reihaneh Safavi-Naini, editors, *FC 2014*, volume 8437 of *LNCS*, pages 436–454. Springer, Heidelberg, March 2014.

[Eya15] Ittay Eyal. The miner's dilemma. In *2015 IEEE Symposium on Security and Privacy*, pages 89–103. IEEE Computer Society Press, May 2015.

[GKL14] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. Cryptology ePrint Archive, Report 2014/765, 2014. http://eprint.iacr.org/2014/765.

[GKL15] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 281–310. Springer, Heidelberg, April 2015.

[GKL17] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol with chains of variable difficulty. In *CRYPTO*, 2017. https://eprint.iacr.org/2016/1048.

[HMQ04] Dennis Hofheinz and Jörn Müller-Quade. Universally composable commitments using random oracles. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 58–76. Springer, Heidelberg, February 2004.

[Int16] Intel. Proof of elapsed time (poet). 2016. https://intelledger.github.io/introduction.html.

[KKKT16] Aggelos Kiayias, Elias Koutsoupias, Maria Kyropoulou, and Yiannis Tselekounis. Blockchain mining games. In *Proceedings of the 2016 ACM Conference on Economics and Computation (EC)*, pages 365–382, 2016.

[KN12] Sunny King and Scott Nadal. PPCoin: Peer-to-peer crypto-currency with proof-of-stake. 2012. https://peercoin.net/assets/paper/peercoin-paper.pdf.

[KP15] Aggelos Kiayias and Giorgos Panagiotakos. Speed-security tradeoffs in blockchain protocols. Cryptology ePrint Archive, Report 2015/1019, 2015. http://eprint.iacr.org/2015/1019.

[KP16] Aggelos Kiayias and Giorgos Panagiotakos. On trees, chains and fast transactions in the blockchain. Cryptology ePrint Archive, Report 2016/545, 2016. http://eprint.iacr.org/2016/545.

[KRDO17] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *CRYPTO*, 2017. http://eprint.iacr.org/2016/889.

[Kwo14] Jae Kwon. Tendermint: Consensus without mining. 2014. https://tendermint.com/static/docs/tendermint.pdf.

[Lit11] Litecoin. 2011. https://litecoin.org.

[Lys02] Anna Lysyanskaya. Unique signatures and verifiable random functions from the DH-DDH separation. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 597–612. Springer, Heidelberg, August 2002.

[MJS+14] Andrew Miller, Ari Juels, Elaine Shi, Bryan Parno, and Jonathan 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.

[MO16] Tal Moran and Ilan Orlov. Proofs of space-time and rational proofs of storage. Cryptology ePrint Archive, Report 2016/035, 2016. http://eprint.iacr.org/2016/035.

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

[NKMS15] Kartik Nayak, Srijan Kumar, Andrew Miller, and Elaine Shi. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. Cryptology ePrint Archive, Report 2015/796, 2015. http://eprint.iacr.org/2015/796.

[NXT14] NXT Community. Nxt whitepaper. 2014. https://www.dropbox.com/s/cbuwrorf672c0yy/NxtWhitepaper_v122_rev4.pdf.

[PPK+15] Sunoo Park, Krzysztof Pietrzak, Albert Kwon, Joël Alwen, Georg Fuchsbauer, and Peter Gaži. Spacemint: A cryptocurrency based on proofs of space. Cryptology ePrint Archive, Report 2015/528, 2015. http://eprint.iacr.org/2015/528.

[PS16] Rafael Pass and Elaine Shi. FruitChains: A fair blockchain. Cryptology ePrint Archive, Report 2016/916, 2016. http://eprint.iacr.org/2016/916.

[PSS17] Rafael Pass, Lior Seeman, and Abhi Shelat. Analysis of the blockchain protocol in asynchronous networks. In *EUROCRYPT*, 2017. https://eprint.iacr.org/2016/454.

[RMCI16] Leonid Reyzin, Dmitry Meshkov, Alexander Chepurnoy, and Sasha Ivanov. Improving authenticated dynamic dictionaries, with applications to cryptocurrencies. Cryptology ePrint Archive, Report 2016/994, 2016. http://eprint.iacr.org/2016/994.

[SBBR16] Okke Schrijvers, Joseph Bonneau, Dan Boneh, and Tim Roughgarden. Incentive compatibility of bitcoin mining pool reward functions. In *Bitcoin Workshop*, 2016.

[SSZ16] Ayelet Sapirstein, Yonatan Sompolinsky, and Aviv Zohar. Optimal selfish mining strategies in bitcoin. In *Financial Cryptography and Data Security (FC)*, 2016.

[SZ15] Yonatan Sompolinsky and Aviv Zohar. Secure high-rate transaction processing in bitcoin. In Rainer Böhme and Tatsuaki Okamoto, editors, *FC 2015*, volume 8975 of *LNCS*, pages 507–527. Springer, Heidelberg, January 2015.

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

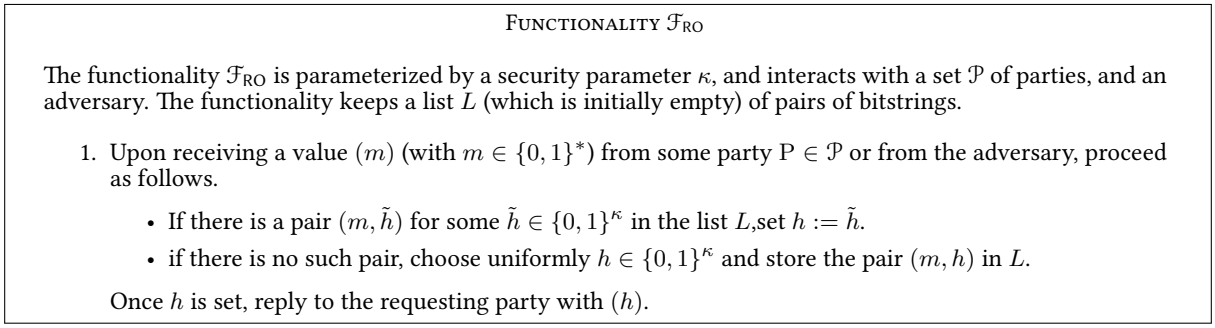
[Woo14] Gavin Wood. Ethereum: A secure decentralized transaction ledger. 2014. http://gavwood.com/paper.pdf.

A Supporting Materials

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

A.1 Random Oracle Functionality $\mathcal{F}_{\mathsf{RO}}$

The random oracle model (e.g., [BR93]) captures an idealization of a hash function. We here present the random oracle functionality $\mathcal{F}_{\mathsf{RO}}$ that has been defined in [HMQ04].



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.

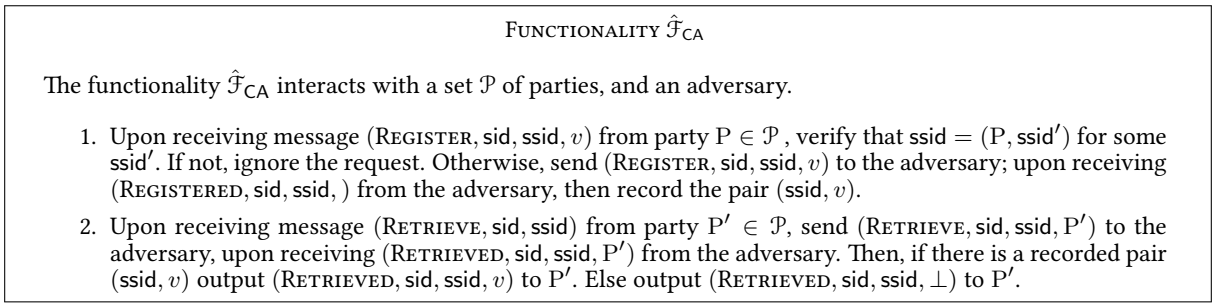
1. Upon receiving a value (m) (with $m \in \{0,1\}^*$) from some party $\mathrm{P} \in \mathcal{P}$ or from the adversary, proceed as follows.
   - If there is a pair $(m, \tilde{h})$ for some $\tilde{h} \in \{0,1\}^\kappa$ in the list L,set $h := \tilde{h}$.
   - if there is no such pair, choose uniformly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

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

Figure 10: Random oracle functionality $\mathcal{F}_{\mathsf{RO}}$.

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

We present the certificate authority functionality following the modeling of [Can03, CR03].



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

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

We present the multi-session version of the digital signature functionality in [Can03].

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

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

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

Signature Generation: Upon receiving a message $(\textsc{Sign}, \mathsf{sid}, \mathsf{ssid}, m)$ from a signer $P \in \{S_1, \ldots, S_k\}$, verify that $\mathsf{ssid} = (P, \mathsf{ssid}')$ for some ssid'. If not, ignore the request. Otherwise, send $(\textsc{Sign}, \mathsf{sid}, \mathsf{ssid}, m)$ to the adversary. Upon receiving $(\textsc{Signature}, \mathsf{sid}, \mathsf{ssid}, m, \sigma)$ from the adversary, verify that no entry $(\mathsf{ssid}, m, \sigma, \textsc{pk}, 0)$ 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 12: Multi-session signature functionality $\hat{\mathcal{F}}_{\mathsf{SIG}}$.

Strengthened unique signature scheme. Unique signature scheme was introduced in [Lys02]. Here we consider a strengthened version which consists of four algorithms, a randomized key generation algorithm $\mathsf{uKeyGen}$, a deterministic key verification algorithm $\mathsf{uKeyVer}$, a deterministic signing algorithm uSign, and a deterministic verification algorithm $\mathsf{uVerify}$. Essentially, $(\mathsf{uKeyGen}, \mathsf{uKeyVer})$ can be view a variant of one-way relation [DHLW10], and 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.1. *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

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

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*

$$\Pr\left[(\mathit{PK}, \mathit{SK}_1, \mathit{SK}_2) \leftarrow \mathcal{A}(1^\kappa) \ : \ \mathsf{uKeyVer}(\mathit{PK}, \mathit{SK}_1) = 1 \wedge \mathsf{uKeyVer}(\mathit{PK}, \mathit{SK}_1) = 1 \wedge \mathit{SK}_1 \neq \mathit{SK}_2\right] \leq \mathsf{negl}(\kappa)$$

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

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

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

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

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

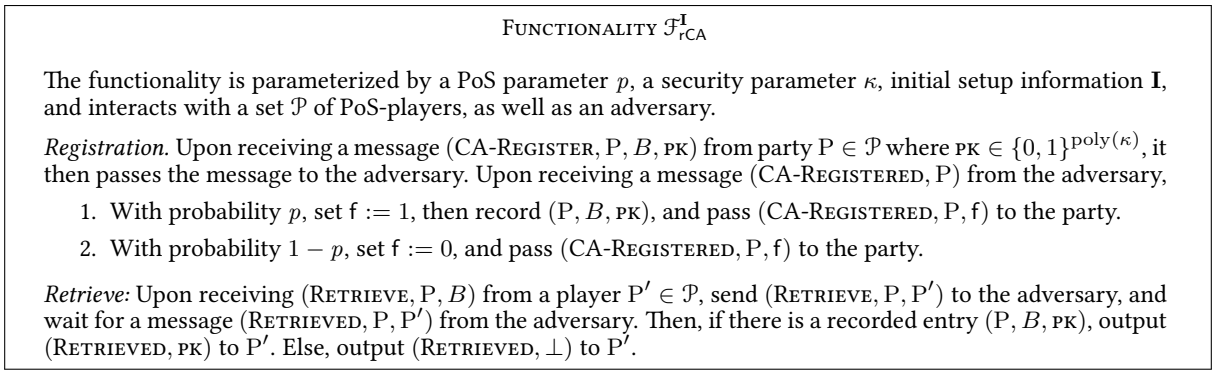
$$\Pr\left[(pk, sk) \leftarrow \mathsf{uKeyGen}(1^\kappa); (x, \sigma) \leftarrow \mathcal{A}^{\mathsf{uSign}(sk,\cdot)}(1^\kappa) \ : \ \mathsf{uVerify}(pk, x, \sigma) = 1 \wedge (x, \sigma) \notin Q\right] \leq \mathsf{negl}(\kappa)$$

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

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

A.4 Resource Certificate Authority Functionality $\mathcal{F}_{\mathsf{rCA}}^{\mathbf{I}}$

At any time step, a PoS-player P could send a register command (CA-Register, P, B, pk) to ask for registration. The functionality then records (P, B, pk) (if permitted by the adversary), with probability p. Then, for each execution round, a different player P could request the functionality retrieving the message registered by P, the functionality then returns the record of P if it permitted by the adversary. Otherwise, the player $\mathrm{P}_{j'}$ will not receive pk. The formal description of $\mathcal{F}_{\mathsf{rCA}}^{\mathbf{I}}$ is given in Figure 13.



Functionality $\mathcal{F}_{\mathsf{rCA}}^{\mathbf{I}}$

The functionality is parameterized by a PoS parameter p, a security parameter κ, initial setup information $\mathbf{I}$, and interacts with a set $\mathcal{P}$ of PoS-players, as well as an adversary.

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

1. With probability p, set $\mathsf{f} := 1$, then record (P, B, pk), and pass (CA-Registered, P, f) to the party.
2. With probability $1 - p$, set $\mathsf{f} := 0$, and pass (CA-Registered, P, f) to the party.

Retrieve: Upon receiving (Retrieve, P, B) from a player $\mathrm{P}' \in \mathcal{P}$, send (Retrieve, P, P′) to the adversary, and wait for a message (Retrieved, P, P′) from the adversary. Then, if there is a recorded entry (P, B, pk), output (Retrieved, pk) to P′. Else, output (Retrieved, $\perp$) to P′.

Figure 13: Resource certificate authority functionality $\mathcal{F}_{\mathsf{rCA}}^{\mathbf{I}}$.

There are multiple ways to instantiate $\mathcal{F}_{\mathsf{rCA}}^{\mathbf{I}}$. Intuitively, in our main application scenario, $\mathcal{F}_{\mathsf{rCA}}^{\mathbf{I}}$ is implemented by a protocol in $\{\hat{\mathcal{F}}_{\mathsf{CA}}, \mathcal{F}_{\mathsf{RO}}\}$-hybrid model, and then multi-session certificate authority functionality $\hat{\mathcal{F}}_{\mathsf{CA}}$ can be implemented by an already "mature" blockchain (i.e., Bitcoin). Please see [DFZ16] for details.