A Permissionless Proof-of-Stake Blockchain with Best-Possible Unpredictability
(or, How to Mimic Bitcoin via Proof-of-Stake) *

Lei Fan† Jonathan Katz‡ Phuc Thai§ Hong-Sheng Zhou¶

May 25, 2021

Abstract

To eliminate the unnecessary waste of energy and computing power in Bitcoin, in this paper, we develop a novel proof-of-stake consensus in the permissionless setting. Among other features, our design achieves the "best possible" unpredictability for permissionless proof-of-stake protocols. As shown by Brown-Cohen et al (EC 2019), unpredictability property is critical for proof-of-stake consensus in the rational setting; the flip side of unpredictability property, i.e., predictability can be abused by the attackers for launching strengthened version of multiple attacks such as selfish-mining and bribing, against proof-of-stake systems.

We are inspired by Bitcoin's "block-by-block" design, and we show that a direct and natural mimic of Bitcoin's design via proof-of-stake is secure if the majority 73% of stake is honest. Our result relies on an interesting upper bound of extending proof-of-stake blockchain we establish: players (who may extend all chains) can generate blockchain at most $2.72\times$ faster than playing the basic strategy of extending the longest chain.

We introduce a novel strategy called "D-distance-greedy" strategy, which enables us to construct a class of secure proof-of-stake blockchain protocols, against an *arbitrary* adversary, even assuming much smaller (than 73% of) stake is honest. To enable a thorough security analysis in the cryptographic setting, we develop several new techniques: for example, to show the chain growth property, we represent the chain extension process via a Markov chain, and then develop a random walk on the Markov chain; to prove the common prefix property, we introduce a new concept called "virtual chains", and then present a reduction from the regular version of common prefix to "common prefix w.r.t. virtual chains".

Finally, we note that, ours is the first "block-by-block" style of proof-of-stake in the permissionless setting, naturally mimicking Bitcoin's design; it turns out that this feature, again allows us to achieve the "best possible" unpredictability property. Other existing provably secure permissionless proof-of-stake solutions are all in an "epoch-by-epoch" style, and thus cannot achieve the best possible unpredictability.

*This is a replacement of earlier versions of the work [30, 31, 29]. Several technical issues in previous versions have been addressed; please see Section 6.3 for details.

†Shanghai Jiaotong University, Email: fanlei@sjtu.edu.cn

‡University of Maryland, Email: jkatz@cs.umd.edu

§Virginia Commonwealth University, Email: thaipd@vcu.edu

¶Virginia Commonwealth University, Email: hszhou@vcu.edu

Contents

1 Introduction

Cryptocurrencies like Bitcoin [51] have proven to be a phenomenal success. The underlying techniques hold huge promise to change the future of financial transactions, and eventually the way people and companies compute, collaborate, and interact. At the heart of these cryptocurrency systems are distributed *blockchain* protocols, jointly executed by a large-scale peer-to-peer network of nodes (players) via the so-called *proof-of-work* (PoW) mechanism [26, 4]. These blockchain protocols implement a highly trustworthy, append-only, and always-available public ledger, which can then be used to implement a global payment system (as in Bitcoin) or a global computer (as in Ethereum [15]). Bitcoin's design has unique *permissionless* features: the protocol can be executed in an *open* network environment in which all miners/players are allowed to join/leave the protocol execution at any moment they want. The protocol has very low communication complexity and *can scale* to a large network of nodes.

Proof-of-work mechanism. In Bitcoin, the players follow a consensus protocol to maintain a growing list of records, called *transactions*. The transactions are stored in blocks that are via cryptographic hashes. More concretely, the player, who can find *a valid solution* (a random nonce) to solve *the hash-based proof-of-work puzzle* will become the block producer and has the right to generate the next block. The newly generated block will be appended to *the longest chain* by including the hash value of the last block on the longest chain. Here, to find the solutions of the hash-based proof-of-work puzzles, proof-of-work based system has "wasted" a huge amount of computing resources over the past several years.

From proof-of-work to proof-of-stake. It is definitely desirable to utilize alternative resources such as *coins (also called stakes)* to secure a blockchain. If successful, the new system will be "green" in the sense that it does not require a huge amount of 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., [1, 47, 59, 9]). In a nutshell, in a proof-of-stake based blockchain protocol, players are expected to prove ownership of a certain number of stakes (coins); only the players that can provide such a proof are allowed to participate in the process of maintaining the blockchain. In order to extend the chain, the players make attempt to find the solutions of *the hash-based proof-of-stake puzzles*. Here, the solutions are generated based on the information of the stakes, a time step (round number), and a *public randomness*. Thus, the computational cost to find the solutions are very "cheap".

From ad hoc to rigorous designs. While proof-of-stake (PoS) mechanisms have been widely discussed in the cryptocurrency community (e.g., [1, 47, 59, 9]), these designs are carried out in an *ad hoc* style. We note that, this is the same for proof-of-work based blockchain design: the original Bitcoin design is indeed in an *ad hoc* style. Recent trend is to follow a rigorous approach: security concerns are carefully defined and the designed protocols are mathmatically analyzed.

Notable efforts include the work by Garay et al [33] (and later improved by Pass et al [53]), for analyzing proof-of-work based blockchain in Bitcoin in the *cryptographic setting* where the malicious players could deviate from the protocol arbitrarily, while honest players always stick to the protocol instructions. There, it has been demonstrated that Bitcoin blockchain can achieve important security properties called common prefix, chain quality, and chain growth; see Section 2.2 for the definitions. In addition, research efforts have been made for proof-of-stake based blockchain protocols; e.g., [23, 22].

In the rational setting, rigorous research efforts have been carried out, too. For example, in this setting, Brown-Cohen et al. [14], recently studied the security property called unpredictability for proof-of-stake protocols. Note that, now all players are rational in the sense of seeking the maximum benefit. Intuitively, the flip side of this security property, i.e., predictability means that (certain) protocol players are aware that they will be selected to generate blocks of blockchain, *before* they actually generate the blocks. This "power" of predictability can be abused by the attackers so that they can reduce the difficulty/cost of performing many attacks such as selfish-mining, or bribing. As a result, the consensus protocol becomes more vulnerable. Ideally, we expect a

proof-of-stake protocol can achieve the (best possible) unpredictability property so that the attacks based on the predictability can be addressed as much as possible.

At this point, we ask the following research question:

Is that possible to design a permissionless proof-of-stake blockchain which achieve

1. *the fundamental security properties such as common prefix, chain quality, and chain growth, in the* cryptographic setting*, and*
2. *the best possible unpredictability in the* rational setting*,*

at the same time ?

1.1 Our result

We give an affirmative answer to the above research question. We are inspired by Bitcoin's design of "per block", or "block-by-block" consensus. We remark that, except that our design here, good proof-of-stake solutions already exist in the permissionless setting, however, they all in the "per epoch", or "epoch-by-epoch" format. This immediately implies that, these solutions cannot achieve best possible unpredicatability in the rational setting.

While mimicking Bitcoin's "per block" design may give us a good starting point, it turns out it is non-trivial to develop a "per block" version for proof of stake. We thus highlight some design and analysis techniques here, and in next subsection we will illustrate the details. We show that a direct and natural mimic of Bitcoin's design via proof-of-stake is secure if the majority 73% of stake is honest. Our result relies on an interesting upper bound of extending proof-of-stake blockchain we establish: players (who may extend all chains) can generate blockchain at most 2.7 times faster than playing the basic strategy of extending the longest chain.

We introduce a novel design strategy called "D-distance-greedy" strategy, which enables us to construct a class of secure proof-of-stake blockchain protocols, against an *arbitrary* adversary, even assuming much smaller amount of stake is honest. To enable a thorough security analysis in the cryptographic setting, we develop several new analysis techniques: to show the chain growth property, we represent the chain extension process via a Markov chain, and then develop a random walk on the Markov chain; to prove the common prefix property, we introduce a new concept called "virtual chains", and then present a reduction from the regular version of common prefix to "common prefix w.r.t. virtual chains";

1.2 Technical roadmap

Warm-up: Bitcoin's design and proof-of-work (PoW) based core-chain. We first briefly review Bitcoin's design ideas [51]. The blockchain consists of a chain of ordered blocks $B_0, B_1, B_2, \ldots$, and PoW-players in each round (or time slot) attempt to extend the blockchain with a new block by solving hash-based proof-of-work puzzles [26, 4]. The puzzle for each miner is defined by (1) the "context", i.e., the latest block in the longest blockchain in the miner's view, and (2) the "payload", i.e., the set of valid transactions to be included in the new block; and a valid *puzzle solution*[1] to the problem is defined by a hash inequality. More concretely, assume the longest blockchain for a miner consists of $B_0, B_1, B_2, \ldots, B_i$, and B_i is the latest block. The miner now attempts to find a valid puzzle solution *nonce* which can satisfy the following hash inequality: $\mathsf{H}(\mathsf{hash}(B_i), payload, nonce) < \mathtt{T}$, where $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\kappa$ and $\mathsf{hash} : \{0,1\}^* \to \{0,1\}^\kappa$ are two hash functions, $payload \in \{0,1\}^*$ denotes the set of valid transactions to be included in the new block, and $\mathtt{T} \in [1..2^\kappa]$ denotes the target of proof-of-work puzzle difficulty (which specifies how difficult to identify a puzzle solution). In the case that a new valid solution, *nonce*, is identified, such a solution can be used for defining a new valid block B_{i+1} as follows: $B_{i+1} := \langle h_i, payload, nonce \rangle$, where $h_i := \mathsf{hash}(B_i)$. Then the new block B_{i+1} will be revealed by the miner,

[1] The payload can be considered as part of the solution.

and broadcasted to the network and then accepted by the remaining miners in the system. (Note that, the above description is oversimplified.)

We may consider a 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 $\mathsf{H}(\mathsf{hash}(B_i), \mathit{nonce}) < \mathtt{T}$, and the new block B_{i+1} is defined as $B_{i+1} := \langle h_i, \mathit{nonce} \rangle$. (We often call the blocks in a blockchain protocol, *blocks*, while the blocks in a core-chain protocol, *block-cores*.)

Next, we describe our construction ideas. To make our presentation accessible, we start with a direct mimic of Bitcoin's design, and present the basic version of our proof-of-stake based core-chain protocol, Π^{core}; then we improve this basic version step by step.

Step 1, Π^{core}: Proof-of-stake (PoS) based core-chain, the basic version. We intend to mimic Bitcoin's design; our proof-of-stake based protocol will be maintained by PoS-players (i.e., stakeholders). We first consider the basic strategy that all honest players make attempts to extend the longest chain with a new block. Our design is very similar to that in Bitcoin: the context here consists of the latest block-core in the longest core-chain, the payload in the core-chain is empty, the puzzle solution consists of the current time, a PoS-player's verification key and his signature of the context. 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 $\mathtt{r}$ denote the current time step (or round number); consider a *unique*[2] digital signature scheme $(\mathsf{uKeyGen}, \mathsf{uKeyVer}, \mathsf{uSign}, \mathsf{uVerify})$, 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: $\mathsf{H}(\mathsf{hash}(B_i), \mathtt{r}, \mathrm{PK}, \sigma) < \mathtt{T}$, where $\sigma := \mathsf{uSign}_{\mathrm{SK}}(h_i, \mathtt{r})$, and $h_i := \mathsf{hash}(B_i)$. Here, $\langle \mathtt{r}, \mathrm{PK}, \sigma, \rangle$ can be considered as the solution of the hash-based proof-of-work puzzle. The new block-core B_{i+1} is defined as $B_{i+1} := \langle h_i, \mathtt{r}, \mathrm{PK}, \sigma \rangle$.

Note that, the unforgeability of the unique signature scheme will ensure that, a malicious PoS-player will not be able to forge a signature for a given context. Further, the uniqueness of the unique signature scheme will ensure that the malicious PoS-player will generate exactly one valid signature for the given context. When the adversary (1) follows the basic strategy, i.e., extending the single chain, and (2) has all stakes registered without being aware of the state of protocol execution, then our protocol can be viewed as a proof-of-stake analogy of Bitcoin; in each round, a PoS player, which can be malicious or honest, will generate exactly one solution for a given context. The security properties i.e., chain growth, chain quality, and common prefix can be demonstrated similarly as in Bitcoin consensus [33, 54].

However, in the proof-of-stake setting, it is very cheap to extend a chain. Some proof-of-stake players may follow a strategy to extend all chains, expecting to obtain additional advantage for extending the best chain. This introduces difficulty for security analysis. Furthermore, an *arbitrary* adversary can choose to extend some critical chains and seek a more sophisticated attack. For example, the adversary may extend the weaker/shorter chains only; this could create a scenario that, two (or multiple) chains take turns to be the best chain; as a consequence, the common prefix property may not be achieved. Thus, it is extremely challenging to defend against an arbitrary adversary.

Interestingly, we demonstrate a very useful upper bound: if a PoS player extends all chains, then the player can improve his/her chance of extending chains with a factor at most e where $e \approx 2.718$. Intuitively, we model the chain extension of the adversary as a random tree. To bound the growth rate of the chain that a generated by the adversary, we first bound the number of branches in the random tree. Then, based on the number of branches and the growth rate of each branch, we can bound the maximum length of all branches in the random tree. This upper bound allows us to develop secure core-chain protocols, and we have the following theorem. See Section 3 for more details.

Theorem 1.1 (informal). *Consider core-chain protocol Π^{core} where all honest players follow the basic strategy of extending the longest chain, while malicious players follow an arbitrary strategy; in addition, all players have their*

[2] Roughly speaking, in a unique signature scheme, for every possible verification key PK, every message m, there is a unique signature σ. Please see Section 6.5.1 of Goldreich's textbook [38]; we also include a version of the definition in Appendix G.

stakes registered without being aware of the state of the protocol execution. Assume the involved signature scheme is a secure unique digital signature scheme. If more than 73% *stakes are honest, then the protocol* Π^{core} *can achieve chain growth, chain quality, and common prefix properties.*

Step 2, $\Pi^{\mathrm{core}\circ}$**: Improved version via distance-greedy strategy.** We now show how to deal with an ***arbitrary adversary*** via smaller portion of honest stakes. Our key observation is that, *honest players play a carefully designed strategy of extending the chains.* This is non-trivial: as discussed before, an arbitrary adversary could break the common prefix property by extending the weaker chains.

We introduce a novel, "D-distance-greedy" (where D is a positive integer) strategy. A D-distance-greedy player will make attempts to extend a *set of best chains*; these chains are very "close" to the best chain. We say a chain is "close" to the best chain iff after removing the last D blocks from the best chain, we can obtain a prefix of that chain. In other words, all the chains in the set of best chains share a common prefix that can be obtained by removing the last D blocks from the best chain. Note that, by following the D-distance-greedy strategy, the honest miners extend the chain set that share the same prefix. The D-distance-greedy strategy can have common prefix property enabled, which can effectively defend against an arbitrary adversary. If majority of stakes are honest, then the protocol can achieve the security properties. We can have the following theorem. See Section 4 for more details.

Theorem 1.2 (informal)**.** *Consider core-chain protocol* $\Pi^{\mathrm{core}\circ}$ *where honest players follow the* 2*-distance-greedy strategy (resp.,* 0*-distance-greedy strategy) while adversarial players follow an arbitrary strategy; in addition, all players have their stakes registered without being aware of the state of the protocol execution. Assume the involved signature scheme is a secure unique digital signature scheme. If more than* 66.2% *stakes (resp.,* 73% *stakes) are honest, then the protocol* $\Pi^{\mathrm{core}\circ}$ *can achieve chain growth, chain quality, and common prefix properties.*

We remark that the security analysis of our protocol is highly non-trivial. For chain growth property, in the existing protocols, as the honest players only extend a single longest chain, the analysis of the chain growth is quite straightforward. Roughly speaking, the chain growth equals the number of blocks that are generated by honest players. In our protocol, the honest players are allowed to extend multiple chains that are "close" to the longest chain. While this helps to defend against nothing-at-stake attacks, it also causes difficulty in the analysis. Even when the honest players generate new blocks, the length of the longest chain may not necessarily increase. To analyze the chain growth property, we develop a random work in a Markov chain that consists of multiple states, where each state represents the number rounds that has passed since the previous longest chain is generated. Intuitively, the set of best chains, that is extended by the honest players, grows bigger through time. Thus, after each round, we move to a state that represents a bigger set of best chains in the Markov chain. Until a new longest chain is generated, some of the chains will be removed from the set of best chains (since the length of the best chain increases, after removing the last D blocks from the best chain, we cannot obtain a prefix of those chains). Note that, the probability that the honest players can create a new longest chain is proportional to the number of *good chains*, i.e., the chain that have the same length with the current longest chain in the set of best chain. Thus, we can use the Markov chain to approximate the chain growth of the honest players. Furthermore, to capture the network delay, we introduce "delayed states" into the Markov chain.

In the previous analysis for Bitcoin's proof-of-work consensus [33, 53], the key observation to prove common prefix property is that the honest players only contribute at most one block at a block height. Thus, to break common prefix property, the adversary must generate more blocks than the honest players do. This is impossible since the honest players control the majority of mining power. However, our proof-of-stake protocol allows the players to extend from multiple chains to defend against nothing-at-stake attacks. This will make the analysis more difficult since the honest players may contribute more than one block at a block height. Thus, we introduce a new concept of *virtual block set* and *virtual chains*. Jumping ahead, we can prove the common prefix of the virtual chain due to the fact that honest players contribute at most one virtual block set in a block height. In detail, a virtual block set consists of multiple blocks with the same height that are "close" to each other. Here, we say two

blocks are "close" if the chains from the genesis block to those two blocks are "close", i.e., after removing the last $\mathtt{D}$ blocks from a chain, we can obtain a prefix of the other chain. Similar to the normal chain, the virtual chain consists of multiple virtual block sets that are linked together. As we mentioned above, on each block height, there is at most one honest virtual block set. (Here, we say a virtual block set is honest if the first generated block in the virtual block set is generated by an honest player.) Thus, to break the common prefix property, the adversary needs to generate more virtual block sets than the honest players. This requires the adversary to control the majority of the stakes (which contradicts to our assumption). Finally, since the blocks in the same virtual block set are "close", we can show common prefix w.r.t. the virtual chains implies the regular common prefix property.

Step 3, Π^{main}: From the core-chain to a blockchain. In this step, we will "upgrade" the core-chain protocol to a regular blockchain protocol so that payload (e.g., the transactions) can be included. Intuitively, the core-chain can be viewed as a (biased) randomness beacon; we can use the beacon to select a PoS-player to generate a new block so that the blockchain can be extended. More concretely, once a new block-core B_{i+1} is generated by a PoS-player (in the core-chain protocol), then the PoS-player is selected for generating the new block $\tilde{B}_{i+1}$, in the following format: $\tilde{B}_{i+1} = \langle \mathsf{hash}(\tilde{B}_i), B_{i+1}, X_{i+1}, \tilde{\mathrm{PK}}, \tilde{\sigma} \rangle$, where $\tilde{\sigma} \leftarrow \mathsf{Sign}_{\tilde{\mathrm{SK}}}(\tilde{h}_i, B_{i+1}, X_i)$, X_{i+1} is payload and $\tilde{h}_i := \mathsf{hash}(\tilde{B}_i)$, and $B_{i+1} := \langle h_i, \mathtt{r}, \mathrm{PK}, \sigma \rangle$. Here the PoS-player holds two pairs of keys, i.e., $(\mathrm{SK}, \mathrm{PK})$ of the unique signature scheme $(\mathsf{uKeyGen}, \mathsf{uSign}, \mathsf{uVerify})$, and $(\tilde{\mathrm{SK}}, \tilde{\mathrm{PK}})$ of a regular[3] digital signature scheme $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$. Now we attach each block to the core-chain via the corresponding block-core; we can reduce the security of the blockchain protocol to the security of the core-chain protocol. Please also see Figure 8 in Appendix for a pictorial illustration. Finally, we have the theorem; See Section C for more details.

Step 4, $\Pi^{\mathrm{main}\bullet}$: Securing the core-chain further, against adaptive stake registration. The protocol $\Pi^{\mathrm{core}\circ}$ above is expected to be executed in a less realistic setting where all players must have their stakes registered without being aware of the state of the protocol execution. Recall that the hash inequality $\mathsf{H}(\mathit{context}, \mathit{solution}) < \mathtt{T}$ is used in the process of extending the chains. In reality, an adversary may have a stake registered *based on the state* of the protocol execution. More concretely, the adversary can play a "rejection re-sampling" strategy to generate keys, and then have his/her stake registered adaptively: the adversary first runs the key generation algorithm to obtain a key-pair $(\mathrm{PK}, \mathrm{SK})$, and then checks if the corresponding (PK, σ) is a valid solution to the hash inequality; if not, the adversary re-samples a new key-pair. This adaptive stake registration strategy enables the adversary (to be selected) to extend the chains with much higher probability. To address this concern, we introduce new ideas to our protocol design: to extend the chains with new blocks, a player must have his/her stake registered a specified number of rounds earlier. Now, the players do not know about the state of the blockchain when they can start extending the chain. Thus, they cannot perform the "rejection re-sampling" strategy to increase their chance of creating new blocks. See Section D for more details.

Best possible unpredictability. We now move to the security analysis from the cryptographic setting to the rational setting in which all players are seeking the maximum utility. Brown-Cohen et al have demonstrated that it is important to achieve unpredictability property [14], as we discussed before. In Section 5, we show that our design can achieve the best possible unpredictability. Our design, is the first "block-by-block" style of proof-of-stake in the permissionless setting, closely mimicking Bitcoin's design; this feature, allows us to achieve the "best possible" unpredictability property. In contrast, other existing provably secure permissionless proof-of-stake protocols (e.g., [45, 23, 5, 7, 22]), are all in a "epoch-by-epoch" style; the players can be c-predictable, where c is the number of blocks in an epoch, and thus they cannot achieve the best possible unpredictability. Indeed, at the beginning of any epoch, the public randomness is known by all players. Thus, the players can predict whether or not they can generate new blocks in that epoch.

Organization. The remaining of the paper is organized as follows. In Section 2, we introduce an analysis

[3] To achieve adaptive security, this regular digital signature scheme will be replaced by a forward-secure digital signature scheme [8]. We remark that, the core-chain protocols, i.e., the protocols without having payload included, in previous steps are adaptively secure.

framework for proof-of-stake protocols. In Section 3, we construct the basic version of our proof-of-stake based core-chain protocol, and then provide the security analysis ideas. In Section 4, we investigate greedy strategies, and develop a modified proof-of-stake based core-chain protocol to defend against an arbitrary adversary. Then we provide more details for the analysis of the modified core-chain protocol against an arbitrary adversary. In Section 5, discuss the predictability of our protocol, compare with existing protocols, and some predictability-based attacks. In Appendix C, we upgrade the core-chain protocol to a full-fledged blockchain protocol. In Appendix D, we improve the modified core-chain protocol further; the players are allowed to register their key-pairs adaptively. In Appendix E, many extensions are provided. In Appendix F, we provide discussions on other attacks on our design.

2 Security Model (in the Cryptographic Setting)

In order to study the security of Bitcoin-like proof-of-work based protocols, Garay et al. [33] proposed a cryptographic framework and showed that (a simplified version of) Bitcoin protocol can achieve several important security properties. Then, Pass et al. [53] strengthened Garay et al.'s analysis by considering a more realistic communication network (i.e., partially synchronous network) in which messages from honest players can be delayed with a bounded number of rounds. Below we define a framework for analyzing proof-of-stake based blockchain protocols. We note that we take many formulation ideas from the previous framework [33, 53].

2.1 Blockchain protocol executions

The execution of proof-of-stake blockchain protocol Following Canetti's formulation of the "real world" executions [17], we present an abstract model for proof-of-stake (PoS) blockchain protocol Π in the hybrid world of the partially synchronous network communication functionality and some setup functionality.

We consider the execution of blockchain protocol Π that is directed by an environment $\mathcal{Z}(1^\kappa)$ (where κ is a security parameter), which activates a set $\mathcal{P}$ of PoS-players. Agreement on the first, so-called *genesis block*, is a necessary condition in all common blockchains for the parties to achieve eventual consensus. This block includes the identities (e.g., public keys) and stake distribution of the player at the beginning of the protocol execution. During the protocol execution the stake distribution can be changed when transactions are added to the ledger.

The environment $\mathcal{Z}$ can "manage" protocol players through an adversary $\mathcal{A}$ that can dynamically corrupt honest parties. More concretely, the protocol execution proceeds as follows. Each party in the execution is initialized with an initial state including all initial public information e.g., a genesis block. The environment $\mathcal{Z}$ first actives the adversary $\mathcal{A}$ and provides instructions for the adversary. The execution proceeds in rounds, and in each round, a protocol party could be activated by the environment or the functionalities. Players are equipped with (roughly synchronized) clocks that indicate the current round.

In each round r, each PoS-player $P \in \mathcal{P}$, with a local state $state_r$ (note that $state$ originally includes the initial state), proceeds as follows. When PoS-player P is activated by the environment $\mathcal{Z}$ with the input x from the environment, and potentially P receives subroutine output message $(\textsc{Message}, P', m)$ for any $P' \in \mathcal{P}$, from the network, the PoS-player P interacts with the setup functionality and receives some output y.

Next, the PoS-player P executes the protocol Π on input its local state $state$, the current round r, an input from the environment x, the value y received from the setup functionality, and the message m received from the network; and then P obtains an updated local state $state_{r+1}$ and an outgoing message m', i.e.,$\{state_{r+1}, m'\} \leftarrow \Pi(state_r, r, x, y, m)$. After that, P broadcast the message m' to the network. Note that, messages are broadcasted by an honest player are guaranteed to arrive at all other honest players within a maximum delay of Δ.

At any round r of the execution, $\mathcal{Z}$ can send message $(\textsc{Corrupt}, P)$, where $P \in \mathcal{P}$, to adversary $\mathcal{A}$. Then $\mathcal{A}$ will have access to the party's local state and control P. Let $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$ be a random variable denoting the joint $\mathtt{VIEW}$ of all parties (i.e., all their inputs, random coins and messages received) in the above protocol execution; note that this joint view fully determines the execution.

For simplicity, we focus on the idealized "flat" model where all PoS-players have the same number of stakes. In section E, we discuss how to extend our main results in the idealized flat, static difficulty model to the more realistic non-flat, adaptive difficulty setting. Protocol players are allowed to join the protocol execution $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$. In the current version of our modeling, we assume that when (honest) PoS-players leave the protocol execution, they will *erase* their own local internal information.[4]

2.2 Security properties

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 $\mathsf{len}(\mathcal{C})$ to denote *blockchain length*, i.e., the number of blocks in blockchain $\mathcal{C}$; and here $\mathsf{len}(\mathcal{C}) = \ell$. We use sub blockchain (or subchain) for referring to segment of a chain; here for example, $\mathcal{C}[0,\ell]$ refers to an entire blockchain, whereas $\mathcal{C}[j,m]$, with $j \geq 0$ 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 $\mathcal{C}$; here i denotes the *block height* of B_i in chain $\mathcal{C}$. If blockchain $\mathcal{C}$ is a prefix of another blockchain $\mathcal{C}_1$, we write $\mathcal{C} \preceq \mathcal{C}_1$. If a chain $\mathcal{C}$ is truncated the last κ blocks, we write $\mathcal{C}[\neg\kappa]$.

For some $\mathcal{A}, \mathcal{Z}$, consider some $\mathtt{VIEW}$ in the support of $\mathsf{EXEC}^{(\Pi^V,\mathcal{C})}(\mathcal{A},\mathcal{Z},\kappa)$. We use the notation $|\mathtt{VIEW}|$ to denote the number of rounds in the execution, $\mathtt{VIEW}^r$ to denote the prefix of $\mathtt{VIEW}$ up until round r, $state_i(\mathtt{VIEW})$ denote the local state of player i in $\mathtt{VIEW}$, $\mathcal{C}_i(\mathtt{VIEW}) = \mathcal{C}(state_i(\mathtt{VIEW}))$ and $\mathcal{C}_i^r(\mathtt{VIEW}) = \mathcal{C}_i(\mathtt{VIEW}^r)$.

Chain growth, common prefix, and chain quality. Previously, several fundamental security properties for proof-of-work blockchain protocols have been defined: *common prefix property* [33, 53], *chain quality property* [33], and *chain growth property* [43]. Intuitively, the chain growth property states that the chains of honest players should grow linearly to the number of rounds. The common prefix property indicates the consistency of any two honest chains except the last κ blocks. The chain quality property, aims at indicating the number of honest blocks' contributions that are contained in a sufficiently long and continuous part of an honest chain. Specifically, for parameters $\ell \in \mathbb{N}$ and $\mu \in (0,1)$, the ratio of honest input contributions in a continuous part of an honest chain has a lower bounded μ. We follow the same path to define the security properties for proof-of-stake blockchain protocols. The definitions for these properties are formally given as follows.

Definition 2.1 (Chain growth)**.** *Consider a blockchain protocol* Π *with a set* $\mathcal{P}$ *of players. The chain growth property with parameter* $g \in \mathbb{R}$*, states: for any honest player* P_1 *with local chain* $\mathcal{C}_1$ *at round* r_1*, and honest player* P_2 *with local chain* $\mathcal{C}_2$ *at round* r_2*, where* $P_1, P_2 \in \mathcal{P}$ *and* $r_2 > r_1$*, in the execution* $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$*, it holds that* $\mathsf{len}(\mathcal{C}_2) - \mathsf{len}(\mathcal{C}_1) \geq g(r_2 - r_1)$.

Definition 2.2 (Common prefix)**.** *Consider a blockchain protocol* Π *with a set* $\mathcal{P}$ *of players. The common prefix property states the following: for any honest player* P_1 *adopting local chain* $\mathcal{C}_1$ *at round* r_1*, and honest player* P *adopting local chain* $\mathcal{C}$ *at round* r*, in the execution* $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$*, where* $P_1, P \in \mathcal{P}$ *and* $r \leq r_1$*, it holds that* $\mathcal{C}[\neg\kappa] \preceq \mathcal{C}_1$.

Definition 2.3 (Chain quality)**.** *Consider a blockchain protocol* Π *with a set* $\mathcal{P}$ *of players. The chain quality property with parameters* μ, ℓ*, where* $\mu \in \mathbb{R}$ *and* $\ell \in \mathbb{N}$*, states: for any honest player* $P \in \mathcal{P}$*, with local chain* $\mathcal{C}$ *in round* r*, in* $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$*, it holds, for large enough* ℓ *consecutive blocks of* $\mathcal{C}$*, the ratio of honest blocks is at least* μ.

3 Proof-of-stake Core-chain, the Basic Version

In this section, we describe the basic version of our core-chain protocol. Similar to Bitcoin design, to generate new blocks, the players make attempts to solve hash-based PoS puzzles, where the context consists of the latest

[4] Players may sell their own secrete keys; this is out of scope of this paper.

block-core in the longest core-chain and the solution consists of the current time, a PoS-player's verification key and the signature of the context.

In the analysis, we prove that our basic protocol is secure if 73% of the stake is honest. Note that, all players including the adversary, must have their stake registered before the protocol execution starts. (Jumping ahead, in Section 4 we present a better protocol, which is secure if 66.2% of the stake is honest. In Appendix D, we show how to allow new players to join the system during the protocol execution.)

3.1 Our core-chain protocol

We now describe the core-chain protocol Π^{core}; Please also refer to Algorithm 1 for details. In our design, the following building blocks are used: 1) a unique digital signature scheme $(\mathsf{uKeyGen}, \mathsf{uKeyVer}, \mathsf{uSign}, \mathsf{uVerify})$; 2) hash functions H, hash for implementing hash inequality based puzzle and pointing to the previous block, respectively. In high level, a PoS-player in the initialization phase, has itself registered to the system; then the player participates in the process of extending the blockchain, along with other players.

Initialization. Given an (initial) group of PoS-players $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$, a security parameter κ, and a unique digital signature scheme $(\mathsf{uKeyGen}, \mathsf{uKeyVer}, \mathsf{uSign}, \mathsf{uVerify})$, the initialization is as follows: each PoS-player $P_j \in \mathcal{P}$ generates a key pair $(\text{SK}_j, \text{PK}_j) \leftarrow \mathsf{uKeyGen}(1^\kappa)$, publishes the public key PK_j and keeps SK_j secret. The public keys are stored in the genesis block of the blockchain system; let B_0 denote the genesis block. In addition, an independent randomness $\mathsf{rand} \in \{0,1\}^\kappa$ will also be stored in B_0. That is $B_0 = \langle(\text{PK}_1, \text{PK}_2, \cdots, \text{PK}_n), \mathsf{rand}\rangle$. Note that, in this section, we consider that only the players who have been registered in the initialization (i.e., those whose public keys have been recorded on the genesis block) can extend the chain. Jumping ahead, in Appendix D, we will turn to consider how to have new players to join the system during the protocol execution.

Algorithm 1: Protocol Π^{core}

State : At round $\mathtt{r}$, the PoS-player $P \in \mathcal{P}$, with key pair (SK, PK) and local state *state*, proceeds as follows.

Let $\mathbb{C}$ be the set of core-chains in the local state *state*
Compute $\mathcal{C}_{\text{best}} := \mathsf{BestCore}(\mathbb{C}, \mathtt{r})$
$\ell := \mathsf{len}(\mathcal{C}_{\text{best}})$, $prev := \mathsf{hash}(\mathcal{C}_{\text{best}}[\ell])$
$\sigma := \mathsf{uSign}(\text{SK}, \langle prev, \mathtt{r}\rangle)$
if $\mathsf{H}(prev, \mathtt{r}, pk, \sigma) < \mathtt{T}$ **then**
  Create new block $B := \langle prev, \mathtt{r}, \text{PK}, \sigma\rangle$
  $\mathcal{C} := \mathcal{C}_{\text{best}} \| B$
  Broadcast $\mathcal{C}$

Blockchain extension. Based on the genesis block B_0, a blockchain in the format of $B_0 \| B_1 \| B_2 \| \ldots \| B_i$, can be generated, round by round. The honest players always select the longest chain to extend. Let $\mathtt{r}$ denote the current time (or round number). If the PoS-player P is chosen, then the following hash inequality holds: $\mathsf{H}(h_i, \mathtt{r}, \text{PK}, \sigma) < \mathtt{T}$, where $\sigma := \mathsf{uSign}(\text{SK}, \langle h_i, \mathtt{r}\rangle)$, and $h_i := \mathsf{hash}(B_i)$. The new block-core B_{i+1} is defined as $B_{i+1} := \langle h_i, \mathtt{r}, \text{PK}, \sigma\rangle$. In our protocol, the chain must be associated with a strictly increasing sequence of round. More concretely, let ℓ be the length of core-chain $\mathcal{C}_{\text{best}}$. In our design, only the elected PoS-players are allowed to generate new block-cores (to extend the core-chain). Now, each registered PoS-player P will work on the right "context" which consists of the *latest block-core in the longest core-chain and the current time*; formally $context := \langle prev, \mathtt{r}\rangle$ where $prev$ is the hash value of the last block in the longest core chain, and $\mathtt{r}$ denotes the current time. The PoS-player P, that has a key pair (SK, PK), checks if the following hash inequality is satisfied: $\mathsf{H}(prev, \mathtt{r}, \text{PK}, \sigma) < \mathtt{T}$, where $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\kappa$ is a hash function, $\sigma := \mathsf{uSign}(\text{SK}, \langle prev, \mathtt{r}\rangle)$ is the solution, and $\mathtt{T}$ is the difficulty parameter. If the PoS-player P solves the hash inequality, it creates a new block $B = \langle prev, \mathtt{r}, \text{PK}, \sigma\rangle$, updates his local core-chain $\mathcal{C}$ and then broadcasts the local core-chain to the network. We

note that, in our protocol, we use a unique signature scheme to guarantee that, each player can only generate exactly one solution in each round.

Algorithm 2: PROCEDURE BestCore

Input : A chain set $\mathbb{C}$, round $\mathtt{r}$.
Output: The best chain $\mathcal{C}_{\text{best}}$.

for $\mathcal{C} \in \mathbb{C}$ **do**
  Parse $\mathcal{C}$ into $B_0 \| B_1 \| \cdots \| B_\ell$
  for $i \in [1, \ell]$ **do**
    Parse B_i into $\langle prev_i, \mathtt{r}_i, \mathtt{PK}_i, \sigma_i \rangle$
    if $h(B_{i-1}) \neq prev_i$ *or* $\mathsf{H}(prev_i, \mathtt{r}_i, \mathtt{PK}_i, \sigma_i) \geq \mathtt{T}$ *or* $\mathsf{uVerify}(\mathtt{PK}_i, \langle prev_i, \mathtt{r}_i \rangle, \sigma_i) = 0$ *or* $\mathtt{r}_i > \mathtt{r}$ **then**
      Remove $\mathcal{C}$ from $\mathbb{C}$
Set $\mathcal{C}_{\text{best}}$ be the longest core-chain in $\mathbb{C}'$

The best core-chain procedure. Our proof-of-stake core-chain protocol Π^{core} uses the procedure 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 procedure BestCore takes as input, a core-chain set $\mathbb{C}$ and the current time information $\mathtt{r}$. Intuitively, the procedure 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 $\mathtt{r}$, and for each core-chain $\mathcal{C}$, the procedure then evaluates every core-block 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 procedure (1) ensures that $\mathcal{C}[i]$ is linked to the previous block-core $\mathcal{C}[i-1]$ correctly, (2) tests if the hash inequality is correct, and (3) tests if the signature generated by that PoS-player is correct. After the validation, procedure BestCore set the best valid core-chain as the longest one. (See Algorithm 2 for more details.)

3.2 Security analysis, high-level ideas

Let $p = \frac{\mathtt{T}}{2^\kappa}$ be the probability that a given (honest or malicious) player creates a new block from a given chain in a round. Recall that in protocol Π^{core}, for a given chain, each player use the unique signature scheme to generate a signature on the *current context* which consists of the last block of the chain and the round number. Then, based on the signature, the player checks whether or not the hash inequality is satisfied. If yes, the player can generate a new block. Note that, a malicious player could make attempts to generate *multiple* "qualified" signatures; once succeeded, he can provide the hash inequalities with "qualified" inputs, for multiple times, with the goal of increasing his probability of generating a new block. Fortunately, the unforgeability of the unique signature scheme ensures that the malicious player cannot forge any "qualified" signatures, and the uniqueness property of the unique signature scheme ensures that the malicious player can generate *exactly one* valid signature on the current context. Thus, all players, including the honest and the malicious players, have (approximately) the same probability (i.e., p) to create a new block for the current context of a given chain. Please see Appendix A.1 for more details.

Let n be the number of players and ρ be the fraction of malicious players in the protocol execution. The probability that honest players create a new block in a round is $\alpha_0 = 1 - (1-p)^{n(1-\rho)}$. Similarly, the probability that the adversary creates a new block from a given chain is $\beta = 1 - (1-p)^{n\rho}$. We remark that, α_0 and β are proportional to the stakes of the honest and of the malicious players, respectively. Indeed, as $np \ll 1$, we have, $\alpha_0 \approx n(1-\rho)p$ and $\beta \approx n\rho p$. Thus, $\frac{\alpha_0}{\beta} \approx \frac{1-\rho}{\rho}$.

Note that, in our model, it takes up to Δ rounds for an honest player to broadcast a message to all other honest players. Thus, the probability that honest players create a new block may slightly reduce. We denote $\alpha = \frac{\alpha_0}{1+\Delta\alpha_0}$ as the discounted probability of α_0. (We will briefly discuss the discounted probability in the Chain growth paragraph of this section. The detailed calculation steps can be found in Appendix A.3.1.)

Now, we turn to consider the adversarial strategies. An adversary may consider different strategies to break the security of the blockchain system. Since the computational cost to extend the chain in proof-of-stake protocol is "very cheap", an adversary can choose an arbitrary set of chain to extend in each round. Note that, an arbitrary adversary cannot extend the chain faster than an adversary who extends all possible chains in each round.

Let β° be the effective stake of the adversary, i.e., the probability that the adversary can extend a new block on the longest chain in each round. Recall that the computational cost to find the solutions are very cheap. Thus, the adversary may extend from multiple chain to amplify its stake. Jumping ahead, we will later show in Lemma 3.2 that $\beta^\circ \leq 2.718\beta$. Under the assumption that $\alpha > \beta^\circ$ (i.e., 73% of total stakes is honest), protocol Π^{core} is secure. We are ready to state our theorem for our core-chain protocol Π^{core}.

Theorem 3.1 (Theorem 1.1, restated)**.** *Consider core-chain protocol Π^{core} in the presence of an arbitrary adversary, and assume* $(\mathsf{uKeyGen}, \mathsf{uKeyVer}, \mathsf{uSign}, \mathsf{uVerify})$ *is a unique digital signature scheme. If $\alpha = \lambda\beta^\circ$, $\lambda > 1$, then the protocol Π^{core} can achieve chain growth, chain quality, and common prefix properties.*

Here, we first demonstrate an upper bound to show the comparison between players who extend all chains and "naive" players (who extend only one chain in each round). Note that, depend on the strategy, the adversary only extend a small set of chains instead of all chains. However, the arbitrary adversary cannot extend the chain faster than a player, who extend all chains. Next, using similar arguments as in [54], we can prove Theorem 3.1.

3.2.1 A bound for effective stake of the adversary

First, we show a very interesting lemma (i.e., Lemma 3.2) that the effective stake of any arbitrary strategy is bounded by a factor e (the base of natural logarithm), compare with the stake of a restricted adversary, who only extends one chain in one round.

Lemma 3.2. *Consider core-chain protocol $\Pi^{\text{core}\circ}$. Assume that malicious players can generate a new block with probability β in a round. Assume that the malicious players could follow any arbitrary strategy to extend a core-chain $\mathcal{C}_1$ at round r_1 into $\mathcal{C}_2$ at round r_2, where $r_2 > r_1$. For some $\epsilon > 0$, we have $\Pr\left[\mathsf{len}(\mathcal{C}_2) - \mathsf{len}(\mathcal{C}_1) < (1+\epsilon)\beta^\circ \cdot T\right] \geq 1 - e^{-\Omega(T)}$ where $T = r_2 - r_1$, $\beta^\circ = e\beta$, and $e = 2.72$ is the base of natural logarithm.*

Intuitively, if protocol players follow any arbitrary strategy and extend all chains, one of the relatively shorter chains will become the longest chain with certain probability; that means, the longest chain will be extended faster. However, we note that, the shorter the chain is, the lower the probability of being extended into the longest chain is; collectively, the longest chain will strictly increase but will be bounded by a constant factor. We model the chain extension of the adversary as a random tree. To bound the growth rate of the chain, we first bound the number of branches in the random tree. Then, based on the number of branches and the growth rate of each branch, we can bound the maximum length of all branches in the random tree. We present the detail proof in Appendix A.2.

3.3 Proofs ideas of the security analysis

Here, we describe the high-level proof ideas to achieve the security properties. The proof details can be found in Appendix A.3.

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 Appendix A.3.1.) 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. (Please see more detail on Appendix A.3.2)

Lemma 3.3 (Chain growth). *Consider an execution of core-chain protocol Π^{core}, where an honest PoS-player P_1 is with best local core-chain $\mathcal{C}_1$ in round r_1, and an honest PoS-player P_2 with best local core-chain $\mathcal{C}_2$ in round r_2, and $r_2 > r_1$. Then we have $\Pr\left[\mathsf{len}(\mathcal{C}_2) - \mathsf{len}(\mathcal{C}_1) \geq g \cdot t\right] \geq 1 - e^{-\Omega(t)}$ where $t = r_2 - r_1$, $g = (1-\delta)\alpha$, and $\delta > 0$.*

Common prefix Note 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, so that other chain will not grow as fast as the longest chain. Together, we have that the public best chain is longer than any other core-chains after a sufficient long time period. All of the honest players will converge on the best public chain with high probability except the last several block-cores. (Please see more detail on Appendix A.3.3)

Lemma 3.4 (Common prefix). *Assume that $\alpha = \lambda\beta^{\circ}$, and $\lambda > 1$. Consider an execution of core-chain protocol Π^{core}, where two honest PoS-players, P in round r and P_1 in round r_1, with the local best core-chains $\mathcal{C}$ and $\mathcal{C}_1$, respectively, where $r_1 \geq r$. Then we have $\Pr\left[\mathcal{C}[\neg\kappa] \preceq \mathcal{C}_1\right] \geq 1 - e^{-\Omega(\kappa)}$.*

Chain quality In order to reduce the chain quality, the best strategy for malicious parties is to generate as many block-cores as they can. When the honest players generate and broadcast a new block-core, they will try to send out another one to compete with the honest one. We focus on the worst case that the malicious players win all of the competition. During any t consecutive rounds, the chain growth rate is αt on average. The malicious players will contribute $\beta^{\circ} t$ block-cores. The chain quality will remain at least $1 - \frac{\beta^{\circ}}{\alpha}$. (Please see more detail on Appendix A.3.4)

Lemma 3.5 (Chain quality). *Assume $\alpha = \lambda\beta^{\circ}$, and $\lambda > 1$. Consider an execution of core-chain protocol Π^{core}, where an honest PoS-player is with core-chain $\mathcal{C}$. If among ℓ consecutive block-cores in $\mathcal{C}$, there are ℓ_{good} block-cores that are generated by honest PoS-players, then we have $\Pr\left[\frac{\ell_{\text{good}}}{\ell} \geq \mu\right] \geq 1 - e^{-\Omega(\ell)}$ where $\mu = 1 - (1+\delta)\frac{1}{\lambda}$, and $\delta > 0$.*

4 Greedy Strategies, and an Improved Version

In the previous section (Section 3), we have described a protocol that is secure if 73% of the stake is honest (and 27% is malicious). In this section, we will present a better protocol, which is secure if 66.2% of the stake is honest. Similar to that in the previous section, we assume all protocol players have their stakes registered before the protocol execution starts. Jumping ahead, in Section D, this assumption will be eliminated, and we will show how to improve our core-chain design further so that it can be executed in a more realistic setting.

We remark that the security analysis on chain growth and common prefix properties of our protocol is highly non-trivial. (Based on the security analysis of chain growth, the security analysis on chain quality will be similar to the existing analysis in [54].) For chain growth property, we model the chain extension as a Markov chain in which arbitrary adversarial behaviors and network delays can be captured. For common prefix property, we introduce a new concept called "virtual chains", and we show the common prefix for the virtual chains, and further show this will imply the regular common prefix property.

Before going to the technical details, we describe the organization of this section: In subsection 4.1 we introduce a *distance-greedy strategy* in which protocol players make attempts to extend a set of chains that are "close" to the best chain. Then, in subsection 4.2 we describe the details of our protocol using the distance-greedy strategy. Finally, in subsection 4.3, we analyze the security of our protocol.

4.1 Greedy strategies

We remark that, in a proof-of-stake protocol, the computational cost to extend the chains can be "very cheap". Thus, a proof-of-stake players may take a *greedy* strategy to extend the core-chains in a protocol execution: instead of extending a single best chain, the players are allowed to extend *a set of best chains*, expecting to extend the best chain faster. Note that, the set of best chains should be carefully chosen; otherwise, the protocol may not secure. In our greedy strategy, the honest player extend the set of chains that share the same common prefix after removing the last few blocks. With this strategy, the security of the protocol is guaranteed. Next, we will formally study greedy strategy.

Distance-greedy strategies. We first introduce *distance-greedy strategies* for honest protocol players. Consider a blockchain protocol execution. In each player's local view, there are multiple chains, which can be viewed as a tree. More concretely, the genesis block is the root of the tree, and each path from the root to another node is essentially a chain. The tree will "grow": the length of each existing chain may increase, and new chains may be created, round after round. Before giving the formal definition for distance-greedy strategies, we define the "distance" between two chains in a tree.

Definition 4.1 (Distance between two chains)**.** *Here we define the distance between two chains $\mathcal{C}$ and $\mathcal{C}_1$. Let chain $\mathcal{C}$ with length ℓ, be a targeted chain, and let $\mathcal{C}_1$ with length ℓ_1, be a branch chain. We use* $\mathsf{distance}$*(branch chain $\to$ target chain), i.e.,* $\mathsf{distance}(\mathcal{C}_1 \to \mathcal{C})$ *to denote the distance between the target chain $\mathcal{C}$ and the branch $\mathcal{C}_1$. More formally, if d is the smallest non-negative integer so that $\mathcal{C}_1[0, \ell_1 - d] \preceq \mathcal{C}$, then we say the distance let d be a non-negative integer. We say the distance between the target chain $\mathcal{C}$ and the branch $\mathcal{C}_1$ is d, and we write* $\mathsf{distance}(\mathcal{C}_1 \to \mathcal{C}) = d$.

Remark 4.2. *We note that the distance of chain $\mathcal{C}$ from chain $\mathcal{C}_1$ is different from the distance of $\mathcal{C}_1$ from $\mathcal{C}$, and it is very possible that* $\mathsf{distance}(\mathcal{C} \to \mathcal{C}_1) \neq \mathsf{distance}(\mathcal{C}_1 \to \mathcal{C})$*. For example, in Figure 1, the distance of $\mathcal{C}$ from $\mathcal{C}_1$ is 4, i.e.* $\mathsf{distance}(\mathcal{C} \to \mathcal{C}_1) = 4$*, while the distance of $\mathcal{C}_1$ from $\mathcal{C}$ is 2, i.e.,* $\mathsf{distance}(\mathcal{C}_1 \to \mathcal{C}) = 2$*. We also note that the distance of $\mathcal{C}$ from itself is always 0, i.e.,* $\mathsf{distance}(\mathcal{C} \to \mathcal{C}) = 0$.

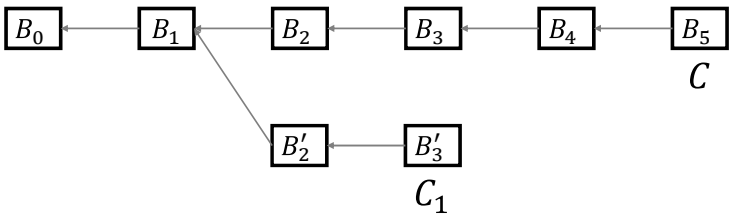

Figure 1: The distance between two chains $\mathcal{C} = B_0\|B_1\|B_2\|B_3\|B_4\|B_5$ and $\mathcal{C}_1 = B_0\|B_1\|B'_2\|B'_3$. (The distance from $\mathcal{C}_1$ to $\mathcal{C}$ is $\mathsf{distance}(\mathcal{C}_1 \to \mathcal{C}) = 2$; the distance from $\mathcal{C}$ to $\mathcal{C}_1$ is $\mathsf{distance}(\mathcal{C} \to \mathcal{C}_1) = 4$).

After explaining the concept of the *distance between two chains*, we are ready to introduce the distance-greedy strategies. Intuitively, a player who plays a distance-greedy strategy will make attempts to extend a *set of chains* in which all chains have the following properties: (1) the chain should be very "close" to the best chain, i.e., the distance from the chain to the best chain must be small; (2) the chain should not fall behind the best chain too much, i.e., the length of the chain must be big. More formally, we have the following definition.

Definition 4.3 ($(\mathtt{D}, \mathtt{F})$-distance-greedy strategy)**.** *Consider a blockchain protocol execution. Let P be a player of the protocol execution, and T be a tree which consists of chains with the same genesis block, in player P's local view. Let $\mathcal{C}_{\text{best}}$ be the longest chain at round r, where $\ell = \mathsf{len}(\mathcal{C}_{\text{best}})$. Consider parameters $\mathtt{D}$ and $\mathtt{F}$. Define a chain set $\mathbb{C}_{\text{best}}$ as*

$$\mathbb{C}_{\text{best}} = \left\{\mathcal{C} \;\middle|\; \mathsf{distance}(\mathcal{C}_{\text{best}} \to \mathcal{C}) \leq \mathtt{D} \bigwedge \mathsf{len}(\mathcal{C}) \geq \ell - \mathtt{F}\right\}$$

We say the player is $(\mathtt{D}, \mathtt{F})$-distance-greedy *if, for all* r*, the player makes attempts to extend all chains* $\mathcal{C} \in \mathbb{C}_{\text{best}}$.

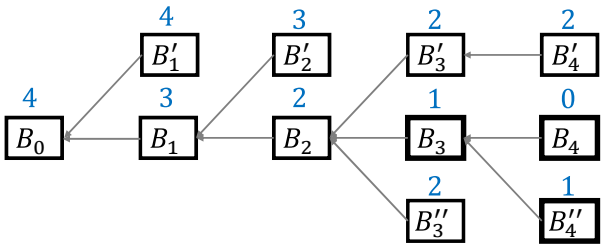

Figure 2: 1-distance-greedy strategy. The best chain is $\mathcal{C}_{\text{best}} = B_0\|B_1\|B_2\|B_3\|B_4$. The blue number on top of each block denotes the distance from the best chain $\mathcal{C}_{\text{best}}$ to the chain from the genesis block B_0 to that block. For example, let $\mathcal{C} = B_0\|B_1\|B_2\|B'_3\|B'_4$, we have $\mathsf{distance}(\mathcal{C}_{\text{best}} \to \mathcal{C}) = 2$. The honest players will extend from the bold blocks (B_3, B_4, B''_4).

Remark 4.4 ($\mathtt{D}$-distance-greedy strategy)**.** *In the above definition,* $\mathtt{F} \leq \mathtt{D}$*. In the remaining of the paper, for simplicity we set* $\mathtt{F} := \mathtt{D}$*, and define the best chain set as*

$$\mathbb{C}_{\text{best}} = \left\{\mathcal{C} \;\middle|\; \mathsf{distance}(\mathcal{C}_{\text{best}} \to \mathcal{C}) \leq \mathtt{D}\right\}$$

and call it $\mathtt{D}$*-distance-greedy strategy. In Figure 2, pictorial illustration for the example of 1-distance-greedy strategies can be found. The honest players will extend the bold blocks* (B_3, B_4, B''_4)*. We also note that many useful variants of the above distance greedy strategies can be designed; a fundamental rule here is that, honest players will only extend the chains which share a common prefix after removing the last few blocks.*

4.2 The modified core-chain protocol $\Pi^{\text{core}\circ}$

Next, we present a new core-chain protocol $\Pi^{\text{core}\circ}$ with the goal of defending against adversaries who play any *arbitrary* strategies.

Initialization. Similar to the initialization in protocol Π^{core} in Section 3, the public keys of an (initial) group of PoS-players $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ are stored in the genesis block B_0, alongside with a randomness rand, i.e., $B_0 = \langle(\mathtt{PK}_1, \mathtt{PK}_2, \cdots, \mathtt{PK}_n), \mathsf{rand}\,\rangle$.

Blockchain extension. This new protocol is based on the core-chain protocol Π^{core} in Section 3 but now the players follow the $\mathtt{D}$-distance-greedy strategy. (See the pseudocode in Algorithm 3). Instead of extending the longest chain, the players extend a set of chains $\mathbb{C}_{\text{best}}$ in which, for all chain $\mathcal{C} \in \mathbb{C}_{\text{best}}$, the distance from the best chain $\mathcal{C}_{\text{best}}$ to the chain $\mathcal{C}$ does not exceed $\mathtt{D}$, i.e., $\mathsf{distance}(\mathcal{C}_{\text{best}} \to \mathcal{C}) \leq \mathtt{D}$.

As the honest players extend from multiple chains, the chain growth of honest players will be amplified. In addition, note that the distances from the best chain to the chains in the set of best chains, are all small than $\mathtt{D}$; those chains share a common prefix that can be obtained by removing the last $\mathtt{D}$ blocks from the best chain. Jumping ahead, this property of blockchain extension process, can be used for proving the security of the protocol $\Pi^{\text{core}\circ}$.

The "best set of core-chains" procedure. The procedure $\mathtt{D}$-$\mathsf{BestCore}^\circ$ will output *a set of best chains* including the longest (i.e., the best) chain, and several chains that are very close to the longest chain. First, the procedure $\mathtt{D}$-$\mathsf{BestCore}^\circ$ uses the procedure $\mathsf{BestCore}$ in the protocol Π^{core} as a sub-procedure to find the best chain. Then, it iterates through the set of chains in the local state of the player to find all the chains in which the distances from the best chain to those chains do not exceed $\mathtt{D}$.

Algorithm 3: PROTOCOL $\Pi^{\text{core}\circ}$

State : At round $\mathtt{r}$, the PoS-player $P \in \mathcal{P}$, with key pair (SK, PK) and local state $state$, proceeds as follows.

Let $\mathbb{C}$ be the set of core-chains in the local state $state$;
Compute $\mathbb{C}_{\text{best}} := \mathtt{D}\text{-}\mathsf{BestCore}^{\circ}(\mathbb{C}, \mathtt{r})$;
for $\mathcal{C} \in \mathbb{C}_{\text{best}}$ **do**
  $\ell := \mathsf{len}(\mathcal{C})$; $prev := h(\mathcal{C}[\ell])$;
  $\sigma := \mathsf{uSign}(\text{SK}, \langle prev, \mathtt{r} \rangle)$;
  if $\mathsf{H}(prev, \mathtt{r}, \text{PK}, \sigma) < \mathtt{T}$ **then**
    Create new block $B := \langle prev, \mathtt{r}, \text{PK}, \sigma \rangle$;
    $\mathcal{C}_1 := \mathcal{C} \| B$;
    Broadcast $\mathcal{C}_1$;

Algorithm 4: PROCEDURE $\mathtt{D}\text{-}\mathsf{BestCore}^{\circ}$

Input : A chain set $\mathbb{C}$, round $\mathtt{r}$.
Output: The best chain set $\mathbb{C}_{\text{best}}$.

$\mathcal{C}_{\text{best}} := \mathsf{BestCore}(\mathbb{C}, \mathtt{r})$, $\mathbb{C}_{\text{best}} := \emptyset$
for $\mathcal{C} \in \mathbb{C}$ **do**
  if $\mathsf{distance}(\mathcal{C}_{\text{best}} \to \mathcal{C}) \leq \mathtt{D}$ **then**
    $\mathbb{C}_{\text{best}} := \mathbb{C}_{\text{best}} \cup \{\mathcal{C}\}$
Return $\mathbb{C}_{\text{best}}$

We note that, as the greedy parameter $\mathtt{D}$ increases, the set of best chains will become bigger. Thus, the honest players have a better opportunities to generate a new longest chain. However, the (computation and storage) complexity of the protocol is proportional to the greedy parameter $\mathtt{D}$. In practice, we can choose $\mathtt{D} = 2$.

4.3 Security analysis

As in previous section, assuming the underlying scheme $(\mathsf{uKeyGen}, \mathsf{uKeyVer}, \mathsf{uSign}, \mathsf{uVerify})$ is a unique digital signature scheme, a malicious player for a given context, can create *exactly one* signature. Note that in previous section, the security properties of protocol Π^{core} have been proven under the assumption of honest majority of stakes based on the honest stake α and the effective stake β° of the adversary. Now, we can prove the security properties of the modified core-chain protocol $\Pi^{\text{core}\circ}$ but under the assumption of honest majority of *effective stakes* based on α° and β°. When $\mathtt{D} = 2$, $\alpha^{\circ} = 1.39\alpha$, protocol $\Pi^{\text{core}\circ}$ is secure if the at least 66.2% of total stakes is honest. We have:

Theorem 4.5 (Theorem 1.2, restated)**.** *Consider an execution of core-chain protocol $\Pi^{\text{core}\circ}$: all honest players follow the $\mathtt{D}$-distance-greedy strategy while adversarial players could follow any arbitrary strategy; in addition, all players have their stakes registered without being aware of the state of the protocol execution. Assume that $(\mathsf{uKeyGen}, \mathsf{uKeyVer}, \mathsf{uSign}, \mathsf{uVerify})$ is a unique digital signature scheme. If $\alpha^{\circ} = \lambda\beta^{\circ}$, $\lambda > 1$, then the protocol $\Pi^{\text{core}\circ}$ can achieve chain growth, chain quality, and common prefix properties.*

As we mentioned before, it is highly non trivial to analyze the security properties, *chain growth* and *common prefix* for our protocol; however, the security analysis for the chain quality of our design is similar to that in [54].

To analysis chain growth property, we construct Markov chain that consists of multiple states, where each state represents the protocol execution after a given number of rounds after the previous longest chain is generated. In each state of the Markov chain, the probability that an honest player can generate a new best chain are different. Note that, in our analysis, we consider the worst case for the chain growth in which when a new best chain is generated, the set of best chains only consists of the best chain and its prefix. At this state, probability that an honest player can generate a new best chain is α (this equals the probability that an honest player can

extend a new block from a single best chain). After each round, if the new best chain is not generated, the set of best chain grows bigger with some certain probability. In this case, we move to the next state in which the probability that an honest player can generate a best chain is bigger. If the new best chain is not generated, we move back to the state where the probability that an honest player can generate a best chain is α. In our analysis, we start with a Markov in synchronous network model, i.e., all messages can be delivered to the honest player in the end of each round. To capture the network delay, we add some delayed states into the Markov chain. After the new best chain is generated, we move to the delayed states. At the delayed states, the honest players will not extend the best chain, i.e., the probability that an honest player can generate a best chain is 0.

For common prefix property, we introduce a new concept of *virtual block set* and *virtual chain*. A virtual block set consists of multiple blocks with the same height that are "close" to each other. Here, we say two blocks are "close" if the chains from the genesis block to those two blocks are "close", i.e., after removing the last $\mathtt{D}$ blocks from a chain, we can obtain a prefix of the other chain. The intuition here is that at each block height, the honest players extend the blocks from the same virtual block set. The virtual chain is formed by the virtual block sets. We say a virtual block set is the previous virtual block set of another virtual block set on the virtual chain if a block in the first virtual block set is the previous block of a block in the other virtual block set. Note that, since the blocks in the same virtual block set are "close", each virtual block set only has one previous virtual block set. In this sense, the virtual chain is very similar to the normal chain. We then prove the common prefix for the virtual chains based on the *honest virtual block set*. Here, we say a virtual block set is honest if the first generated block in the virtual block set is generated by an honest player. On each block height, there is at most one honest virtual block set. Thus, to break the common prefix property, the adversary needs to generate more virtual block sets than the honest players. This requires the adversary to control the majority of the stakes (this contradicts our assumption). Finally, since the blocks in the same virtual block set are "close", we can show common prefix for the virtual chains implies the regular common prefix property.

4.3.1 Chain growth

In order to analyze the chain growth property, we construct a Markov chain that experiences transitions between *states* according to certain *transition probabilities*, and then develop a *random walk* on the constructed Markov chain. More concretely, in the process of blockchain extension, new blocks are generated and published, and in some cases, a new longest chain will be generated and published (i.e., certain chain turns into the longest one in the system after the generation of a new block). In the Markov chain that we plan to construct, we consider and focus on *the total number of rounds that has passed since the previous longest chain has been generated and published*. That is, we use state ⓪ to denote the state in which, *a new longest chain has just been generated and published*; equivalently state ⓪ reflects that the protocol execution is now *"0 round after" the previous longest chain has been generated and published*. Similarly, we use state $\textcircled{i}$, where $i \in \mathbb{N}$, to denote that the protocol execution is now "i number of rounds after" the previous longest chain has been generated and published.

It is easy to see, when the blockchain execution moves from state ⓪ to state ①, or from $\textcircled{i}$ to state $\textcircled{j}$, where $i \in \mathbb{N}$ and $j = i + 1$, no new longest chain will be generated and the length of the longest chain in the system remains the same. However, when the protocol execution moves *back* to ⓪ (say from $\textcircled{i}$ or even from ⓪), a new longest chain will be generated and the length of the longest chain will increase by 1. The chain growth can be approximated as the stationary probability of state ⓪, i.e., the expected number of occurrences of a particular state ⓪ over a long sequence of transitions on the random walk.

Representing chain-extension via Markov chain. We now describe in detail the states and the transition probabilities on the Markov chain. The Markov chain consists of $m + 1$ states ⓪, ①, …,$\textcircled{m}$, where $m = a_{\mathtt{D}}/\alpha$ is an integer and $a_{\mathtt{D}} \geq 1$ is a constant (see Fig. 3). As we mentioned above, state $\textcircled{i}$ represents the protocol execution i rounds after the previous longest chain has been generated and published. Note that, it is possible that the new longest chain is not published after m rounds. Thus, the state $\textcircled{m}$ represents the represents the protocol execution at all the rounds in which the previous longest chain is generated at least m rounds ago. Note that, here, Here,

we analyze the chain growth in the execution where all messages from honest players can be broadcasted to all other honest nodes in $\Delta = 1$. In Appendix B.2, we will present the analysis for arbitrary Δ by adding Δ "delayed states" in the Markov chain.

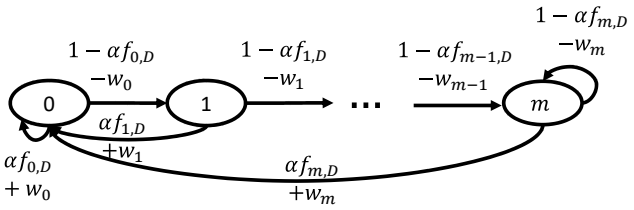

Figure 3: The Markov chain that represents the chain extension process. State i represents protocol execution i rounds after the previous longest chain has been generated and published. At state i, with probability $\alpha f_{i,\mathtt{D}}$, the honest players can extend the longest chain and we move to state ⓪. Additionally, with some probability $w_i \geq 0$, the adversary publishes a new longest chain and we move to state ⓪.

We recall that the honest players extend all the chains in the set of best chains. At a given round, the probability to create a new longest chain is proportional with the number of *good chains*, i.e., the chain that have the same length with the current longest chain in the set of best chain. To be precise, let round r be the round where the first chain of length ℓ is generated. At round r, the number of good chains is *one* (the only good chain is the longest chain). As the time goes by, the number of good chains slowly increases until a longest chain of length $\ell + 1$ is generated. The number of good chain again become one. This process keeps repeating during protocol execution. Note that, the number of longest chain increases faster as $\mathtt{D}$ increases. We denote $f_{i,\mathtt{D}}$ (for any integer $i \geq 0$) as the expected number of chains of length ℓ at round $r + 1 + i$. Thus, the probability that the honest players create a new chain of length $\ell + 1$ at round $r + 1 + i$ is $\alpha f_{i,\mathtt{D}}$.

Furthermore, the adversary may publish a new longest chain with non-negative probability in each round. We denote w_i (for any integer $i \geq 0$) as the probability that the adversary publishes a new chain of length $\ell + 1$ at round $r + 1 + i$. Note that, when the adversary publish a new longest chain, the best chain set may has less chains than when an honest player generate a new longest chain. Here, we always consider the worst case where the best chain set only consists of the longest chain and its prefix.

For each state ⓘ, the transition probability of going from state ⓘ to state ⓪ is $\alpha f_{i,\mathtt{D}} + w_i$, where $w_i \geq 0$. Here, the adversary publish a new longest chain at state i with probability $w_i \geq 0$. Plus, the probability that an honest player generates new longest chain is $\alpha f_{i,\mathtt{D}}$. As we mentioned above, when a new longest chain is generated, we move to the state 0 in the next round. With probability $1 - \alpha f_{i,\mathtt{D}} - w_i$, there is no new longest chain is generated. Thus, for any state i (where $0 \leq i < m$), the transition probability of going from state ⓘ to state ⓙ (where $j = i + 1$) is $1 - \alpha f_{i,\mathtt{D}} - w_i$. Recall that, the state ⓜ represents the represents the protocol execution at all the rounds in which the previous longest chain is generated at least m rounds ago. Thus, the transition probability of going from state ⓜ to state ⓜ is $1 - \alpha f_{m,\mathtt{D}} - w_m$.

We remark that, our Markov chain model also can capture the chain growth of the protocol in which the honest players only extend a single chain [33, 53] (or our protocol Π^{core} with $\mathtt{D} = 0$). In this case, we have $f_{i,\mathtt{D}} = 1, \forall i \in [0..m]$. If the adversary does not help to extend the chain, i.e., $w_i = 0, \forall i \in [0..m]$, at any state, the probability that a new longest chain is generated is α. The chain growth now equals α.

Chain growth as a random walk on the Markov chain. We now model the honest chain extension process from round $r' + 1$ to round r'' ($r'' - r' = t > 0$) as a random walk $s_1, \cdots, s_t$. We start at round $r' + 1$ at state s_1. At round $r' + j$, state s_j is randomly selected based on the transition probabilities of going from state s_{j-1} to other states.

Let $\mathsf{Q} = [\mathsf{q}_0, \cdots \mathsf{q}_m]$ be the stationary distribution, where q_i be the stationary probability of the state ⓘ. We

obtain the following equations for the stationary distribution.

$$\begin{cases} \sum_{i=0}^{\infty} \mathsf{q}_i = 1, \\ \mathsf{q}_0 = \sum_{i=0}^{\infty} \mathsf{q}_i \cdot (w_i + \alpha \cdot f_{i,\mathtt{D}}). \\ \mathsf{q}_i = \mathsf{q}_{i-1} \cdot (1 - (1 - w_{i-1} - \alpha \cdot f_{i-1,\mathtt{D}})), \\ \qquad\qquad\qquad\qquad \forall 1 \le i < m, \\ \mathsf{q}_m = \mathsf{q}_{m-1} \cdot (1 - (1 - w_{m-1}\alpha \cdot f_{m-1,\mathtt{D}})) \\ \qquad\quad + \mathsf{q}_m \cdot (1 - (1 - wm - \alpha \cdot f_{m,\mathtt{D}})) \end{cases} \tag{1}$$

From Eq. 1, we have $\mathsf{q}_0 \ge 1.39\alpha$ when $\mathtt{D} = 2$. Please see Appendix B.1 for more detail.

4.3.2 Common prefix

In order to analyze the common prefix property, we first introduce the notion of *virtual chains*, and then define the common prefix property with respecting to the virtual chains. Then we show that the standard common prefix property can be reduced to common prefix w.r.t. virtual chains.

Defining virtual chains. Based on the definition of D-distance-greedy strategy (as in Definition 4.3), we group all blocks, that are bounded by a distance D, to a single virtual block set. The virtual chain is formed by the virtual block sets. More concretely, consider a local state of a player, we define the virtual block set and virtual chain as follows.

Definition 4.6 (Virtual block sets)**.** *Let $\mathbb{B}$ be the set of all blocks in the local state of the player. We construct a set $\mathbb{V}$ of virtual block sets as follows.*

- *First, set $\mathbb{V} = \emptyset$.*
- *For each block $B \in \mathbb{B}$ such that $\nexists V' \in \mathbb{V} : B \in V'$, let $\mathcal{C}^{(B)}$ be the chain from the genesis block to the block B. We build a virtual block V based on the block B as follows.*
  - *Initialize that $V := \{B\}$;*
  - *for any block $B' \in \mathbb{B}$ such that $\mathsf{len}(\mathcal{C}^{(B')}) = \mathsf{len}(\mathcal{C}^{(B)})$ (B' and B have the same block height) and $\mathsf{distance}(\mathcal{C}^{(B')} \to \mathcal{C}^{(B)}) \le \mathtt{D}$, we set $V := V \cup \{B'\}$.*

We say V_i is the previous virtual block set of V_{i+1} iff there exists a block $B_{i+1} \in V_{i+1}$ and a block $B_i \in V_i$ such that B_i is the previous block of B_{i+1}

Definition 4.7 (Virtual chain)**.** *The corresponding virtual chain consists of multiple consecutive virtual block sets, i.e., $\mathcal{V} = V_0 \| V_1 \| \cdots \| V_\ell$ (where ℓ is a non-negative integer) in which for all $i \in [\ell]$, V_{i-1} is the previous virtual block set of V_i.*

We say a chain $\mathcal{C} = B_0 \| B_1 \| \cdots \| B_\ell$ belongs to the virtual chain $\mathcal{V} = V_0 \| V_1 \| \cdots \| V_\ell$ if and only if for all $i \in [0..\ell]$, $B_i \in V_i$. We write $\mathcal{C} \in \mathcal{V}$.

Definition 4.8 (Best virtual chain)**.** *The best virtual chain $\mathcal{V}_{\text{best}}$ is the virtual chain that the best chain $\mathcal{C}_{\text{best}}$ belongs to, i.e., $\mathcal{C}_{\text{best}} \in \mathcal{V}_{\text{best}}$. (Please see Figure 4 for an example of virtual block sets and the best virtual chain when $\mathtt{D} = 2$.)*

Common prefix property w.r.t. virtual chain. To achieve common prefix property, we first prove the common prefix wrt virtual chain. We introduce a new notion of *honest virtual block set*, where the first published block in the virtual block set is honest. We present the formal definition of an honest virtual block set as follows.

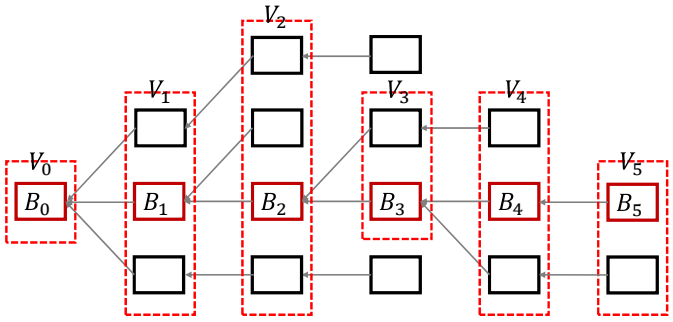

Figure 4: The best virtual chain with $\mathtt{D} = 2$. The best chain is the red solid blocks, i.e., $\mathcal{C}_{\text{best}} = B_0 \| B_1 \| \cdots \| B_5$. Each block B_i on the best chain belongs to a virtual block set V_i which is represented by a dash rectangle. Each virtual block set V_i consists of multiple blocks at the same height that are "close" to the block B_i, i.e., $\forall B_i' \in V_i$, $\mathsf{distance}(\mathcal{C}^{(B_i)} \to \mathcal{C}^{(B_i')}) \leq \mathtt{D}$. Here, for all $i \in [5]$, the virtual block set V_{i-1} is the previous virtual block set of V_i since B_{i-1} is the previous block of B_i. Thus, the best virtual chain is $\mathcal{V}_{\text{best}} = V_0 \| V_1 \| \cdots \| V_5$.

Definition 4.9 (Honest virtual block sets)**.** *Let* $\mathsf{round}(B')$ *be the round number in the block* B'*. Consider a virtual block set* V*, let* $B = \arg\min_{B' \in V} \mathsf{round}(B')$ *be the earliest block in* V*. We say* V *is the honest iff the earliest block* B *is generated by an honest player.*

Now, we will show that for most of the time, there is at most *one* honest virtual block set in each block height. Indeed, since the honest players only extend on the chains that are close to the best chain, unless a new longest chain is generated, the honest players will not create a new block in which a new virtual block set is created. (Please see Fig. 5 for an example.) To break common prefix wrt virtual chain, the adversary must grow a virtual chain as fast as the virtual chain of honest players. This contradicts the assumption that the honest players control more stakes than the adversary.

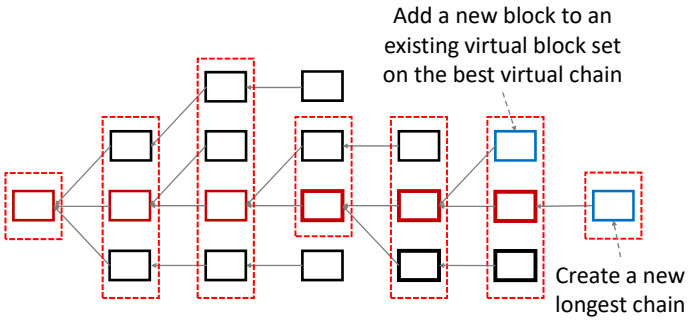

Figure 5: Players extend the set of best chain from Fig. 4, using 2-distance-greedy strategy. The blue blocks denote the new blocks. Here, the players generate either a new block to create a new longest chain (that is longer than the current longest chain) or a new block that is added to an existing virtual block set.

From common prefix w.r.t. virtual chain, to the standard common prefix property. Next, we prove common prefix property from common prefix wrt virtual chain. Consider the set of chains where the latest blocks in those chain belong to a virtual block set. By the definition of the virtual block set, all of those chains share the same common prefix after pruning the last $\mathtt{D}$ blocks. Thus, if a protocol achieves common prefix wrt virtual chain, it also achieves common prefix property by pruning extra $\mathtt{D}$ blocks. Please see Appendix B.3 for the full proof of common prefix property.

4.3.3 Chain quality

At high level ideas, the malicious players cannot extend the chain as fast as the chain growth. Thus, for any κ consecutive blocks, the must be at least one block that is created by an honest player. Please see Appendix B.4 for more detail.

5 (Un)Predictability in the Rational Setting

In this section, we will investigate the unpredictability property [14] of our design in the rational setting; note that, now all players are rational in the sense of seeking the maximum benefit. We further note that, the players do not collaborate and exchange their local states as in [14].

Intuitively, predictability means that (certain) protocol players are aware that they will be selected to generate blocks of blockchain, *before* they actually generate the blocks. This "power" of predictability reduces the difficulty for attackers to perform many attacks such as selfish-mining, or bribing. As a result, the consensus protocol becomes more vulnerable. Note, the predictability definitions here are essentially taken from [14], but rephrased in our terminology .

Global (un)predictability. We first present the formal definition of the global predictability. At high-level idea, global predictability captures that all players in the protocol can predict when a player will be able to produce a block in the future.

Definition 5.1 (L-global predictable)**.** *Consider a blockchain protocol Π with a set $\mathcal{P}$ of players, and integer $L > 0$. Consider a player $P \in \mathcal{P}$ at round r and let $\mathcal{C}$ be the best valid chain in the view of P at round r. We say player P is* L-global predictable *if at round r, any player $P' \in \mathcal{P}$ can efficiently predict whether or not player P can generate the L-th block in the future. In other words, any player P' can predict whether or not player P can extend the chain $\mathcal{C}'$ in which $\mathsf{len}(\mathcal{C}') \geq \mathsf{len}(\mathcal{C}) + L - 1$ and $\mathcal{C}'$ is the public chain at some round $r' \geq r$.*

Several existing protocols [45, 23, 5, 7] (and our protocols) are well designed, and do not suffer from global predictability. Indeed, if digital signature is used as part of solution to the hash inequality puzzle, in the process of selecting players to produce blocks, then, since the remaining protocol players are not aware of who will be selected, global *un*predictability can be achieved.

Local (un)predictability. A weaker version, called *local predictability* has also been introduced.Intuitively, local predictability captures that a miner can predict when she/he will be able to produce a block in the future. As our main focus is local predictability, for simplicity, without otherwise mentioned, we refer to predictability as local predictability. The formal definition of the (local) predictability is given as follows.

Definition 5.2 (L-(un)predictable)**.** *Consider a blockchain protocol Π with a set $\mathcal{P}$ of players, and integer $L > 0$. Consider a player $P \in \mathcal{P}$ at round r and let $\mathcal{C}$ be the best valid chain in the view of P at round r. We say player P is* L-predictable *if at round r, P can efficiently predict whether or not she/he can extend the chain $\mathcal{C}'$ in which $\mathsf{len}(\mathcal{C}') \geq \mathsf{len}(\mathcal{C}) + L - 1$ and $\mathcal{C}'$ is the public chain at some round $r' \geq r$. Otherwise, we say P is* L-unpredictable.

Best-possible unpredictability. We now show that our protocol achieves best-possible unpredictability in the rational setting. Recall from Observation 1 in [14] shows that in any proof-of-stake protocol, all players can *always* predict whether or not they can generate the next block, i.e., they are 1-predictable. In our protocol, with probability almost 1, the players can *only* predict whether or not they can generate the next block, i.e., they are 2-unpredictable. Indeed, a player is 2-unpredictable if and only if she/he is the one that generates the next blocks. The event occurs with probability $\frac{1}{n}$, where n is the number of players.

Lemma 5.3. *Consider $\Pi^{\text{core}\circ}$ protocol, at any round r, a rational player P is 2-unpredictability with the probability of $1 - \frac{1}{n}$, where n is the number of players.*

Proof. Let $\mathcal{C}$ be the best chain at round r. From Def. 5.2, P is 2-unpredictability at round r iff P can predict whether or not she/he can extend the chain $\mathcal{C}'$ in which $\mathsf{len}(\mathcal{C}') = \mathsf{len}(\mathcal{C}) + 1$ and $\mathcal{C}'$ is the public chain at some round $r' \geq r$. In other words, at round r, P needs to know about the chain $\mathcal{C}'$ at round r'. Since players do not collaborate and exchange their local states, this event only happens when the player P generates the next block on the chain $\mathcal{C}$ to form the chain $\mathcal{C}'$. Since all players have the same probability to extend the chain. This event happens with probability of at most $\frac{1}{n}$. □

On the other hand, in epoch-based proof-of-stake protocols [45, 23, 5, 7, 22], the players can be c-predictable, where c is the number of blocks in an epoch. (In [45, 23, 5, 22], $c = \Omega(\kappa)$, where κ is the security parameter. In [7], c is a predefined parameter). Indeed, at the beginning of any epoch, the public randomness is known by all players. Thus, the players can predict whether or not they can generate new blocks in that epoch.

Predictability-based attacks. We now describe the attacks where the attackers that rely of the power of the predictability. We remark that, here, we mainly discuss the attacks on the protocols that only allow local predictability. We also briefly discuss how to improve the attacks on the protocols that allow global predictability.

Predictable selfish mining attacks. In a selfish mining attack, a rational attacker may not publish its blocks immediately to the rest of the network. This greatly endangers the fairness of blockchain. Since the proof-of-stake protocol allows the players to predict weather or not they can generate a few block in the future, proof-of-stake protocols are more vulnerable to selfish mining attacks. Brown-Cohen et al. [14] presents a predictable selfish mining attack in PoS protocols. In this attack, for a given integer $u > 0$, the rational players predict a time period t_u is the time it takes for them to generate the next u blocks. If the probability, that other players cannot generate u blocks in the time period t_u, is big enough, the rational players will hide those u blocks until the last block (i.e., the u-th block) is generated. With this strategy, with high probability, the blocks of the rational players will belong to the longest chain.

For the protocols that allow global predictability, the attackers know when other players can generate new block. Thus, the attacks only hides its blocks when other players cannot generate u blocks in the time period t_u. Hence, the attacks can always successfully perform selfish-mining attack. As the rational players can predict more blocks in the future, they have more chance to successfully perform predictable selfish mining attacks, thus increasing the fraction of their blocks on the longest chain. Since our protocol only allow minimal predictability, we can minimize the affect of predictable selfish mining attacks.

Predictable bribing attacks. In bribery attacks [7], the attacker pays players to work on specific chains, aiming at benefiting the attacker, e.g., supporting double spending or censorship attack . In proof-of-stake protocol, those attacks will be more dangerous if the players can predict further in the future. In epoch-based proof-of-stake protocol, the players can predict whether or not they can create new blocks in the future. Thus, at the beginning of each epoch the attacker make attempts to bribe those players, who can new blocks in the future round of the current epoch. If the attacker can bribe enough players, it can control the majority of the blocks that are generated in the epoch. We consider two cases as follows. (Here, *confirmation time* is defined as the time elapsed between the moment a transaction is included on the blockchain and the time it is finally recorded into a confirmed block.)

Case 1: *The confirmation time is smaller than the length of each epoch.* In this case, the attacker can perform a double spending attack by issuing some transactions at the beginning of the epoch. At the same time, it hides all of its blocks. At the end of the epoch, those transactions are confirmed on the best public chain. The attacker now publishes its hidden blocks and reverts the transactions that are issued at the beginning.

Case 2: *The confirmation time is bigger than the length of each epoch.* The attacker can perform a censorship attacks to prevent certain transactions from becoming confirmed. In each epoch, the attacker perform a predictable bribing attack to control majority of the blocks, i.e., controlling the longest chain. Since all blocks on the longest chain belong to the attacker, it can prevent any transaction from being added to the blockchain.

For the protocols that allow global predictability, if the attacker cannot bribe a player, it can perform a DoS attack on that player to prevent the player to produce the new block (since the attacker know when the player will generate the block). Thus, even when the attacker cannot bribe the majority of the players that will generate blocks in the future, it can still successfully perform the attack.

In our block-based proof-of-stake protocol, most of the time, the players can only predict 1 block in the future. Thus, the attacker does not know which players to bribe. Hence, it cannot perform the predictable bribing attacks.

6 Related Work

6.1 Cryptocurrency and proof-of-work

The first decentralized currency system, Bitcoin [51], was introduced in 2008, based on moderately-hard cryptographic puzzles (also called proof-of-work puzzles [26, 4]). Please refer to the online course [52] and the survey [13].

The security of Bitcoin system has been analyzed in the rational setting, e.g., [28, 27, 56, 57, 42, 14]. In the cryptographic setting, efforts have been made, e.g., [33, 54, 58, 43, 6, 34, 46, 24]. Garay et al [33], and then Pass et al [54] initialize the rigorous security analysis for the Bitcoin consensus. Several important cryptographic properties, *common prefix* [33, 54], *chain quality* [33], and *chain growth* [43], have been considered for proof-of-work protocols. Badertscher et al [6] provide the analysis in the universal composability framework [18].

6.2 Proof-of-stake

Using coins/stakes to construct cryptocurrency has been intensively considered. Since the inception of the idea in an online forum [10], several proof-of-stake proposals have been introduced and/or implemented (e.g., [1, 47, 59, 16, 9]). We remark that these proposals are *ad hoc* without formal security, and it is not clear how to formally prove the security of these proposals. Very recently, several provably secure proof-of-stake based blockchain proposals have been developed. More details can be found below.

6.2.1 Bitcoin-like proof-of-stake.

We first review Bitcoin-like longest chain based, proof-of-stake consensus proposals; these are closely related to the results in the current writeup. We note that, all these related proposals are in the format of "epoch by epoch" which are different from Bitcoin's format of "block by block"; our proposals are the first such proof of stake protocols, in the format of "block by block". These related proposals include Snow White [22] by Pass and Shi, Ouroboros Praos [23] and Ouroboros Genesis [5] by Kiayias et al, and a proposal [7] by Bagaria et al.

In Snow White [22], the protocol execution is divided into epochs, where each epoch consists of $\Omega(\kappa)$ blocks (where, κ is the security parameter). The players are selected to generate new blocks based on the public key, the current round number, and the randomness of the current epoch (via a hash inequality). If the players are elected to generate new blocks, they also need to provide some randomness that will be used to generate the randomness in the next epoch. The Snow White protocol can only defend against a "mildly adaptive adversary", i.e., after the adversary corrupt some players, they remain honest for a mild corruption delay period. We remark that, the Snow White protocol is based on Pass and Shi's early proposal, the Sleepy protocol [55] (in which the new players are not allowed to join the system during the execution) as a starting point. In the Snow White protocol, new players are allowed to join the system but relies on external trust (e.g., a set of trusted players). Jumping ahead, this limitation in joining has been eliminated in later proposals, Ouroboros Genesis [5], and our earlier version [32].

In Ouroboros Praos [23], similar to Snow White, the protocol execution is divided into epochs of $\Omega(\kappa)$ blocks. In each round of an epoch, the player queries a verifiable random function (VRF) [50] to determine whether it

can generate a new block; note that the input of VRF consists of the current round, the public key of the player, and the randomness of the current epoch. Here, the randomness of the epoch is computed based on the output of the VRF in the previous epoch. Ouroboros Praos adopt an erasure model to achieve "fully adaptive security", i.e., the adversary can instantly corrupt the players. Note that, the protocol of Ouroboros Praos does not allow new players to join the system after the protocol execution starts. In their follow-up work, Ouroboros Genesis [5], new players are allowed to join the protocol execution securely.

In Bagaria et al. [7], similar to Ouroboros Praos, the players use a VRF to determine whether or not they can generate new blocks. However, here, the length of each epoch can be arbitrary. Jumping ahead, this allows a trade-off between nothing-at-stake attacks and predictability attacks [14]. We will discuss more in the security analysis in rational setting paragraph. The authors also adopt the technique in [5] and our earlier version [32] to allow new players to join the system.

Security analysis in cryptographic setting Note that, based on the analysis in [22, 23], the consistency (i.e. common prefix property) is guaranteed with error $e^{-\Omega(\kappa)}$ by removing the last $O(\kappa^2)$ blocks. While in Bitcoin, the consistency is guaranteed with error $e^{-\Omega(\kappa)}$ by removing only the last $O(\kappa)$ blocks. Blum et al. [11] improves the analysis for the consistency (i.e. common prefix property) of proof-of-stake based blockchain protocols in cryptographic setting. Now, similar to Bitcoin, the consistency is guaranteed with error $e^{-\Omega(\kappa)}$ by removing only the last $O(\kappa)$ blocks. However, in [11], the "multiply honest" rounds (the rounds that have multiple honest players that can generate new blocks) are treated as "malicious" rounds (the rounds that have at least one malicious players that can generate new blocks). Kiayias et al. [44] extends the result from [11]. Here, the "multiple honest" rounds are treated as "unique honest" rounds (the rounds that have exactly one honest player that can generate a new block). Dembo et al. [24] introduces a new technique to analyze the blockchain protocols (including Bitcoin and proof-of-stake based protocols). The analysis shows that the best strategy for the adversary to break consistency is private "double-spend attack", i.e., the adversary does not contribute to the public best chain and aims to extend a private chain that is longer than the public best chain. We remark that the analysis in [11, 44, 24] can only be applied for the protocols in which the players only extend a single best chain. In contrast, the players in our protocol extend a set of best chains instead of the single best chain. Thus, we introduce a new analysis strategy to analyze the security of our protocol.

Security analysis in rational setting Brown-Cohen et al. [14] exploit the security of proof-of-stake based protocols in a rational setting. Contrary to Bitcoin, the proof-of-stake based protocols allow the players to predict whether or not they can create new blocks in the future. Indeed, in proof-of-work based protocols, the randomness is in some sense external to the blockchain. Thus, the players cannot predict whether or not they can create new blocks in the future. On the other hand, in proof-of-stake based protocols, the randomness comes from the blockchain itself. Hence, the players can predict whether or not they can create a few next block in the future. We refer to this as predictability. The predictability allows the adversary to perform many rational attacks such as predictable selfish mining and predictable bribing.

In [55, 22, 23, 5], the protocol execution consists of multiple epoches, and in each epoch multiple ($\Omega(\kappa)$) blocks will be generated. Here, the adversary is restricted and cannot amplify its stakes in each epoch. However, for each player, he can predict all blocks he will generate in the current epoch. Thus, their protocols are more vulnerable to predictability attacks [14]. In [7], the length of the epoch can be arbitrary. This allows a trade-off between nothing-at-stake attacks and predictability attacks. If the length of the epoch increases, the protocol is more secure against nothing-at-stake attacks. However, it allows the players to predict more blocks in the future, i.e., the protocol is more vulnerable to predictability attacks.

In this work, we propose the first "block-by-block" solution with provable security. Note that, in the "epoch-by-epoch" proof-of-stake protocol, the randomness is extracted from the blocks that have been published an epoch, i.e., many blocks ago. Meanwhile, in the "block-by-block" proof-of-stake protocol, the randomness is

extracted from the previous block. Thus, comparing with "epoch-by-epoch" protocols, our protocol can also defend against nothing-at-stake attacks while being more resistant to predictability attacks.

6.2.2 Additional solutions to proof-of-stake

In addition to the Bitcoin-like *pure* proof-of-stake solutions, there are additional solutions to proof-of-stake blockchain systems. Several of those solutions have been implemented and deployed in the real world.

Multi-round protocols. Differently from Bitcoin-like *pure* proof-of-stake protocols, which have very low communication complexity, proof-of-stake protocols have been constructed using *multiple rounds* of communication. We below list several visible proposals along this line.

Algorand [20, 36] presents an interesting alternative solution. In Algorand, VRF has been used for selecting a committee of players; for each player, the opportunity to be selected is proportional to the number of coins in the player's account. Then, the committee members run a Byzantine Agreement (BA) sub-protocol to jointly generate a block. Algorand aims to scale to millions of players. According to the report from [36], the committee size can be multiple thousands; see Figure 3 in [36]. However, to ensure the security of Algorand, a bigger fraction of players must be honest, comparing with traditional BA protocols. For example, if the size of the committee is $2,000$, it requires 80% of honest players to guarantee the security of Algorand. We remark that, the idea of using VRF for proof-of-stake protocol, has later been adopted in Ouroboros Praos [23] but for Bitcoin-like proof-of-stake.

Ouroboros [45] presents the first provably secure proof-of-stake protocol. Ouroboros protocol consists of multiple epochs in which each epoch consists of multiple round. In each round, a player will be selected as the leader to generate a new block. The leader is selected based on the stake distribution, the current round number, and a random string. Note that, the random string is updated in every epoch via a coin tossing protocol that is executed by the players.

EOS [2] presents a delegated proof-of-stake protocol in which the token holders (those who hold the token on the blockchain) may select block producers through a continuous approval voting system. At the beginning of each round, 21 unique block producers are chosen by preference of votes cast by token holders. The selected block producers can create new blocks under the agreement of 15 or more block producers.

Dfinity [41] proposes a four-layer consensus protocol. The players are registered at the first layer. The second layer provides the randomness for all higher layer. The blocks are generated at the third layer. In each round, the protocol ranks the player based on the random beacon of that round. All players can generate new blocks. However, each block has a different weight. The weight of the block is assigned based of the rank of the block procedure at that round. The best chain is selected as the "heaviest" chain in term of accumulated block weight. The forth layer provides fast finality of the block by using threshold signature.

Hybrid proof-of-stake. Bitcoin-like *hybrid* proof-of-stake has previously been investigated. In [25], Duong et al studied hybrid consensus using both proof of stake and proof of work. In [3, 48], Andreina et al studied hybrid consensus using both proof of stake and trusted hardware (Trusted Execution Environment).

6.3 Earlier versions of this work

We remark that, our project is the first proof-of-stake protocol in the format of "block by block". Except our solutions here, existing provably secure Bitcoin-like proof-of-stake protocols are all in the format of "epoch by epoch". Compared with "epoch by epoch" style, proof-of-stake protocols, it is highly non-trivial to defend against nothing-at-stake attacks in the "block-by-block" ones.

This project was started in 2017, and the first version was online July 2017 [30]; This was concurrent and independent of Ouroboros Praos [23]. In an early version [31] that submitted to Eurocrypt 2018, we provided a strategy which allows new players to join the proof-of-stake system; allowing new players to securely join the proof of stake system has been independently investigated in Ouroboros Genesis [5].

An anonymous Eurocrypt 2018 reviewer identified an attack on the " fully greedy" strategy in [31]. Here, the honest players extend the set of chains that are slightly shorter than the best chain. The adversary makes attempts to extend on both chain in private, i.e., the adversary does not publish their block immediately. Note that, since the length of the two chains are equal, the honest players will extend both chains. However, as the chain extension process are random, at some given rounds, only one chain get "lucky" and can be extended by an honest player. When an honest player publishes a new block in a chain, the adversary will release a block from the other chain to "balance" the length of the two chains. Eventually, the two chains will be diverted and only share a common block in the ancient past. In a later version [29], this issue was fixed by introducing the "D-greedy strategy." We remark that, Bagaria et al in [7], independently identified this attack.

Bagaria et al. [7] also pointed out the analysis for the chain growth in [29] is not correct[5]. The adversary can slow down the chain growth of honest players by publishing a private chain such that (1) the length of the private chain equals the length of the public best chain, and (2) there are no public chains (expect the prefixes of this private chain) that are "close" to this private chain. In this case, since the private chain and the public best chain have the same length, the honest players will randomly select a chain as the new best chain. If the honest players select the private chain as the new best chain, the chain growth of the honest players will be slow down since the set of best chains only contains a single best chain.

Finally, we note that, in the current version, all issues have been addressed; especially we introduce several new analysis strategies. In addition, we consider a rational analysis framework by Brown-Cohen et al [14] and provide the investigation of the unpredictability of our proof-of-stake protocols.

Acknowledgement: We are very grateful to Thang Dinh, Andrew Miller, Pramod Viswanath, and an anonymous Euorcrypt 2018 reviewer, for pointing out technical issues in the early versions of the work, and for providing us their valuable feedbacks and suggestions. We thank Alex Chepurnoy, Yi Ding, Tuyet Duong, and Yanxue Jia for helpful discussions and for proofreading the early versions of the paper.

References

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

[2] EOS whitepaper. 2018. https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md.

[3] S. Andreina, J.-M. Bohli, G. Karame, W. Li, and G. A. Marson. Pots: A secure proof of tee-stake for permissionless blockchains. *IEEE Transactions on Services Computing*, 2020.

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

[5] C. Badertscher, P. Gazi, A. Kiayias, A. Russell, and V. Zikas. Ouroboros Genesis: Composable Proof-of-Stake Blockchains with Dynamic Availability. In *CCS*, 2018. https://eprint.iacr.org/2018/378.

[6] C. Badertscher, U. Maurer, D. Tschudi, and V. Zikas. Bitcoin as a transaction ledger: A composable treatment. In J. Katz and H. Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 324–356. Springer, Heidelberg, Aug. 2017.

[7] V. Bagaria, A. Dembo, S. Kannan, S. Oh, D. Tse, P. Viswanath, X. Wang, and O. Zeitouni. Proof-of-stake longest chain protocols: Security vs predictability. *arXiv preprint arXiv:1910.02218*, 2019.

[8] M. Bellare and S. K. Miner. A forward-secure digital signature scheme. In M. J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 431–448. Springer, Heidelberg, Aug. 1999.

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

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

[5] The authors also provide ideas to analyze the chain growth for the early version of our protocol. However, the analysis in this work is independent and difference with the one in [7].

[11] E. Blum, A. Kiayias, C. Moore, S. Quader, and A. Russell. The combinatorics of the longest-chain rule: Linear consistency for proof-of-stake blockchains. In S. Chawla, editor, *31st SODA*, pages 1135–1154. ACM-SIAM, Jan. 2020.

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

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

[14] J. Brown-Cohen, A. Narayanan, A. Psomas, and S. M. Weinberg. Formal barriers to longest-chain proof-of-stake protocols. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, pages 459–473, 2019.

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

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

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

[18] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, Oct. 2001.

[19] C. L. Canonne. A short note on poisson tail bounds. *Retrieved from the website: http://www. cs. columbia. edu/ ccanonne*, 2017.

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

[21] K.-M. Chung, H. Lam, Z. Liu, and M. Mitzenmacher. Chernoff-hoeffding bounds for markov chains: Generalized and simplified. *arXiv preprint arXiv:1201.0559*, 2012.

[22] P. Daian, R. Pass, and E. Shi. Snow white: Robustly reconfigurable consensus and applications to provably secure proofs of stake. In *FC*, 2019. http://eprint.iacr.org/2016/919.

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

[24] A. Dembo, S. Kannan, E. N. Tas, D. Tse, P. Viswanath, X. Wang, and O. Zeitouni. Everything is a race and nakamoto always wins. In J. Ligatti, X. Ou, J. Katz, and G. Vigna, editors, *ACM CCS 20*, pages 859–878. ACM Press, Nov. 2020.

[25] T. Duong, L. Fan, J. Katz, P. Thai, and H.-S. Zhou. 2-hop blockchain: Combining proof-of-work and proof-of-stake securely. In L. Chen, N. Li, K. Liang, and S. A. Schneider, editors, *ESORICS 2020, Part II*, volume 12309 of *LNCS*, pages 697–712. Springer, Heidelberg, Sept. 2020.

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

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

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

[29] L. Fan, J. Katz, and H.-S. Zhou. A large-scale proof-of-stake blockchain in the open setting, Feb 2019. A version submitted to and presented at Stanford Blockchain Conference 2019.

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

[31] L. Fan and H.-S. Zhou. A Scalable Proof-of-Stake Blockchain in the Open Setting (or, How to Mimic Nakamoto's Design via Proof-of-Stake). Sept 2017. A version submitted to Eurocrypt 2018; see https://cryptographylab.bitbucket.io/pubs/iChing.pdf.

[32] L. Fan and H.-S. Zhou. A Scalable Proof-of-Stake Blockchain in the Open Setting (or, How to Mimic Nakamoto's Design via Proof-of-Stake). Feb 2018. https://eprint.iacr.org/2017/656/20180218:233721.

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

[34] J. A. Garay, A. Kiayias, and N. Leonardos. The bitcoin backbone protocol with chains of variable difficulty. In J. Katz and H. Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 291–323. Springer, Heidelberg, Aug. 2017.

[35] P. Gaži, A. Kiayias, and A. Russell. Stake-bleeding attacks on proof-of-stake blockchains. In *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, pages 85–92. IEEE, 2018.

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

[37] M. Goemans. Chernoff bounds, and some applications. *https://math.mit.edu/~goemans/18310S15/chernoff-notes.pdf*, 2015.

[38] O. Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, Cambridge, UK, 2004.

[39] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.

[40] S. Goldwasser and R. Ostrovsky. Invariant signatures and non-interactive zero-knowledge proofs are equivalent (extended abstract). In E. F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 228–245. Springer, Heidelberg, Aug. 1993.

[41] T. Hanke, M. Movahedi, and D. Williams. Dfinity technology overview series, consensus system. *arXiv preprint arXiv:1805.04548*, 2018.

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

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

[44] A. Kiayias, S. Quader, and A. Russell. Consistency of proof-of-stake blockchains with concurrent honest slot leaders. *arXiv preprint arXiv:2001.06403*, 2020.

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

[46] L. Kiffer, R. Rajaraman, and a. shelat. A better method to analyze blockchain consistency. In D. Lie, M. Mannan, M. Backes, and X. Wang, editors, *ACM CCS 2018*, pages 729–744. ACM Press, Oct. 2018.

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

[48] W. Li, S. Andreina, J.-M. Bohli, and G. Karame. Securing proof-of-stake blockchain protocols. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, pages 297–315. Springer, 2017.

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

[50] S. Micali, M. O. Rabin, and S. P. Vadhan. Verifiable random functions. In *40th FOCS*, pages 120–130. IEEE Computer Society Press, Oct. 1999.

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

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

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

[54] R. Pass, L. Seeman, and a. shelat. Analysis of the blockchain protocol in asynchronous networks. In J.-S. Coron and J. B. Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 643–673. Springer, Heidelberg, Apr. / May 2017.

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

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

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

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

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

A Supplemental Material for Section 3

A.1 The probability of generating blocks in a round

We first show that by using the unique signature scheme, all players have the same probability to generate a new block from a given core-chain.

Lemma A.1. *Consider core-chain protocol* Π^{core} *in the presence of an arbitrary adversary, and assume* $(\mathsf{uKeyGen}, \mathsf{uKeyVer}, \mathsf{uSign}, \mathsf{uVerify})$ *is a unique digital signature scheme. Consider a chain* $\mathcal{C}$ *at round* r*. The probability that a player* $P_j \in \mathcal{P}$ *(where* P_j *can be honest or malicious) generates a new block from the chain* $\mathcal{C}$ *at round* r *is* $p = \frac{\mathtt{T}}{2^\kappa}$*, where* $\mathtt{T}$ *is the difficulty parameter.*

Proof sketch. Recall in protocol Π^{core} that the player P_j can generate a new core-block at round r, if the following hash inequality is satisfied:

$$\mathsf{H}(prev, r, \text{PK}, \sigma) < \mathtt{T},$$

where $prev$ is the hash value of the last block on the chain $\mathcal{C}$, and PK is a public key that has been registered in the initialization and has been recorded at the genesis block, and σ is a signature on the context $\langle prev, r\rangle$, i.e., $\mathsf{uVerify}(\text{PK}, \langle prev, \mathtt{r}\rangle, \sigma) = 1$.

Here, the hash function H returns a random value in $\{0,1\}^\kappa$. For each tuple of $(prev, r, \text{PK}, \sigma)$, the probability that the hash inequality is satisfied is $p = \frac{\mathtt{T}}{2^\kappa}$. Note that, for a given chain $\mathcal{C}$, the hash value $prev$ is fixed. In any round r, the player P_j will generate exactly one "qualified" signature σ.

Indeed, the unforgeability of the unique signature scheme ensures that the player P_j cannot forge any "qualified" signatures without knowing the secret key. Hence, the player P_j can generate the signature by using his own key pair $(\text{SK}_j, \text{PK}_j)$. Furthermore, the uniqueness of the unique signature scheme guarantees that for each key pair, there is exactly one "qualified" signature over a context. Therefore, for a given chain $\mathcal{C}$ at a round r, the player P_j can generate exactly one valid tuple $(prev, r, \text{PK}, \sigma)$. In other words, the probability that the player P_j generates a new block from the chain $\mathcal{C}$ at round r is p. □

From Lemma A.1, we have the following corollary.

Corollary A.2. *Let* n *be the number of players and* ρ *be the fraction of malicious players in the protocol execution. At a given round* r*, we have*

- *The probability that an honest player generates a new block in round* r *is* $\alpha_0 = 1 - (1-p)^{n(1-\rho)}$.
- *Consider a core-chain* $\mathcal{C}$*, the probability that a malicious player creates a new block from the chain* $\mathcal{C}$ *is* $\beta = 1 - (1-p)^{n\rho}$.

A.2 More materials for effective stake of the adversary

To maximize the adversarial amplification ratio the adversary extend from all valid chains. However, depend on the strategy, the adversary may or may not public its chains. We bound the adversarial amplification ratio by using a branching random walk, in which a set of all chain with the same randomness is considered as a branch. The adversarial amplification ratio will later be used to prove the chain quality and common prefix properties.

Let X' is a Poisson random variable with expectation β that represents the number of blocks that are generated by the adversary in a round, i.e.,

$$\Pr[X' = x] = p_x = e^{-\beta} \cdot \frac{\beta^x}{x!}$$

After each round, for a given branch, we add $X + 1$ new branches with the increasing length $0, 1, \cdots, X$. Formally, we describe the chain extension of the adversary as a branching process as follows. At the beginning, there is only one branch of length 0, i.e., $Z_0 = \{0\}$. Let Z_k be the set of all branches at time slot k, where $k \in \{0, 1, 2, \cdots\}$ and G_k be the number of branches in Z_K. Let $X_{k,i}$ be a random variable denoting the random process in i-th branch in Z_k. Here $X_{k,i}$ are independent and identically distributed random variables of X (recall that X is a Poisson random variable that has the expected value $\mathbb{E}[X] = \beta$) , over all $k \in \{0, 1, 2, \cdots\}$ and $i \in \{1, 2, \cdots, G_k\}$. Let $\ell_{k,i}$ be the length of the i-th branch in Z_k. We will add $X_{k,i} + 1$ branches with the length $\ell_{k,i}, \ell_{k,i} + 1, \cdots, \ell_{k,i} + X_{k,i}$ into Z_{k+1}. We denote L_k as the maximum length of all branch in Z_k, i.e.,$L_k = \max_{i \in \{1,2,\cdots,G_k\}} \ell_{k,i}$.

The length of a branch set is equivalent to the increasing length of the longest chain.

Before presenting the detail proofs, we introduce an useful inequality as follows.

Lemma A.3 (Theorem 1 in [19]). *Consider a Poisson random variable X that has the expected value λ. We have the following inequalities.*

- *For any $\epsilon > 0$, we have*
$$\Pr[X > \lambda(1 + \epsilon)] \leq e^{-\frac{\lambda^2\epsilon^2}{2\lambda(1+\epsilon)}} = e^{-\lambda\frac{\epsilon^2}{2(1+\epsilon)}}$$
- *For any $0 < \epsilon < 1$, we have*
$$\Pr[X > \lambda(1 - \epsilon)] \leq e^{-\lambda\frac{\epsilon^2}{2(1+\epsilon)}}$$

Lemma A.4 (Theorem 3 in [37]). *Let $X_1, X_2, \cdots, X_k$ be identical independent random variables in range $[0, 1]$ with expected value λ. Then, for any $\epsilon > 0$, we have*

$$\Pr[\sum_{i=1}^{k} X_i < (1 + \epsilon)k\lambda] \leq e^{-\Omega(k)}.$$

To prove Lemma 3.2, first, we bound the number of different branch in the end of the process (See Lemma A.5). Finally, we use union bound the bound the maximum length of all branches.

Now, we ready to prove the adversarial amplification ratio.

Lemma A.5. *Consider the set of branch Z_k at time k. For any $\epsilon'' > 0$, we have*

$$\Pr[G_k > (\beta + 1)^{k(1+\epsilon'')}] < e^{-\Omega(k)}$$

Proof. The expected number of branches that the adversary can create from one branch in one round is $\beta+1$. Thus, for $j \in 0, 1, \cdots$, we have $\mathbb{E}[\frac{G_{j+1}}{G_j}] = \beta+1$. In other words, we have $\mathbb{E}[\log(G_{j+1}) - \log(G_j)] = \log(\beta+1)$.

Let $Q_1, Q_2, \cdots, Q_k$ be independent and identically distributed random variables with expectation $\log(\beta+1)$. We have,

$$\log G_k = \sum_{j=1}^{k} Q_j.$$

Therefore,

$$\begin{aligned}
&\Pr[G_k > (\beta+1)^{k(1+\epsilon'')}] \\
=&\Pr[\log(G_k) > k(1+\epsilon'')\log(\beta+1)] \\
=&\Pr[\sum_{j=1}^{k} Q_j > k(1+\epsilon'')\log(\beta+1)] \\
<&e^{-\Omega(k)} \text{ (Lemma A.5).}
\end{aligned}$$

□

Now, we ready to bound the maximum length of a branch at a time k.

Proof of Lemma 3.2. For any $\mathtt{A} > 1$, we have,

$$\begin{aligned}
&\Pr\left[L_k > (1+\epsilon)k\beta\mathtt{A}\right] \\
=&\Pr\left[\max_{i\in\{1,2,\cdots,G_k\}} \ell_{k,i} > (1+\epsilon)k\beta\mathtt{A}\right] \\
\leq&\sum_{i\in\{1,2,\cdots,G_k\}} \Pr\left[\ell_{k,i} > (1+\epsilon)k\beta\mathtt{A}\right] && \text{(Union bound)} \\
\leq&G_k \cdot \Pr[Y > (1+\epsilon)k\beta\mathtt{A}] \\
\leq&(\beta+1)^{k(1+\epsilon'')} \cdot \Pr[Y > (1+\epsilon)k\beta\mathtt{A}] && \text{(Claim A.5)}
\end{aligned}$$

We have,

$$\ell_{k,i} = \sum_{j=1}^{k} X'_j \leq \sum_{j=1}^{k} X_j$$

where X_j be i.i.d. Poisson random variable with expectation β.

Let $Y = \sum_{j=1}^{k} X_j$ be a Poisson random variable with expectation $k\beta$. From Lemma A.3, we have,

$$\Pr[Y > (1+\epsilon')k\beta] < e^{-\Omega(k)}.$$

Thus, we have,

$$\Pr[\ell_{k,i} > (1+\epsilon')k\beta] < e^{-\Omega(k)},$$

Since Y is a Poisson random variable with expectation $k\beta$, we have.

$$\Pr[Y = i] = e^{-k\beta}\frac{(k\beta)^i}{i!}. \tag{2}$$

Consider $\mathtt{A} > 1$ such that

$$\begin{aligned} \mathtt{A}\left(\frac{\mathtt{A}}{e}\right)^{\mathtt{A}-1} &= (\beta+1)^{1/\beta} \\ \Rightarrow \mathtt{A}^{(1+\epsilon')k\beta}\left(\frac{\mathtt{A}}{e}\right)^{(1+\epsilon')(\mathtt{A}-1)k\beta} &\leq (\beta+1)^{k(1+\epsilon')} \end{aligned}$$

Thus,

$$\begin{aligned} A &= \mathtt{A}^{(1+\epsilon')k\beta}\left(\frac{\mathtt{A}}{e}\right)^{(1+\epsilon')(\mathtt{A}-1)k\beta} \\ &\geq \frac{\sqrt{2\pi}\sqrt{(1+\epsilon')\mathtt{A}k\beta}((1+\epsilon')\mathtt{A}k\beta)^{(1+\epsilon')\mathtt{A}k\beta}}{\sqrt{2\pi}\sqrt{(1+\epsilon')k\beta}((1+\epsilon')k\beta)^{(1+\epsilon')k\beta}(k\beta)^{(1+\epsilon')(\mathtt{A}-1)k\beta}} \end{aligned}$$

Using Stirling's approximation, we have

$$A \geq \frac{((1+\epsilon')\mathtt{A}k\beta)!}{((1+\epsilon')k\beta)!(k\beta)^{(1+\epsilon')(\mathtt{A}-1)k\beta}} \tag{3}$$

Combine with Eq. 2, we have,

$$A \geq \frac{\Pr[Y=(1+\epsilon')k\beta]}{\Pr[Y=(1+\epsilon')\mathtt{A}k\beta]}$$

$$\Rightarrow \frac{\Pr[Y=(1+\epsilon')k\beta]}{\Pr[Y=(1+\epsilon')\mathtt{A}k\beta]} \leq (\beta+1)^{k(1+\epsilon')}$$

Furthermore, for any $i \geq 0$, we have,

$$\begin{aligned} &\frac{\Pr[Y=(1+\epsilon')k\beta+i]}{\Pr[Y=(1+\epsilon')k\beta+i+1]} > \frac{\Pr[Y=(1+\epsilon')\mathtt{A}k\beta+i]}{\Pr[Y=(1+\epsilon')\mathtt{A}k\beta+i+1]} \\ \Rightarrow &\frac{\Pr[Y=(1+\epsilon')k\beta+i]}{\Pr[Y=(1+\epsilon')\mathtt{A}k\beta+i]} > \frac{\Pr[Y=(1+\epsilon')k\beta+i+1]}{\Pr[Y=(1+\epsilon')\mathtt{A}k\beta+i+1]} \end{aligned}$$

Thus,

$$\begin{aligned} &\frac{\Pr[Y=(1+\epsilon')k\beta}{\Pr[Y=(1+\epsilon')\mathtt{A}k\beta]} > \frac{\sum_{i=1}^{\infty}\Pr[Y=(1+\epsilon')k\beta+i+1]}{\sum_{i=1}^{\infty}\Pr[Y=(1+\epsilon')\mathtt{A}k\beta+i+1]} \\ \Rightarrow &\frac{\Pr[Y=(1+\epsilon')k\beta}{\Pr[Y=(1+\epsilon')\mathtt{A}k\beta]} > \frac{\Pr[Y>(1+\epsilon')k\beta}{\Pr[Y>(1+\epsilon')\mathtt{A}k\beta]} \\ \Rightarrow &\frac{\Pr[Y>(1+\epsilon')k\beta]}{\Pr[Y>(1+\epsilon')\mathtt{A}k\beta]} \leq (\beta+1)^{k(1+\epsilon')} \\ \Rightarrow &(\beta+1)^{k(1+\epsilon')}\Pr[Y>(1+\epsilon')\mathtt{A}k\beta] \geq \Pr[Y>(1+\epsilon')k\beta] \end{aligned}$$

We have,

$$\begin{aligned} &\Pr\left[\max_{i\in\{1,2,\cdots,G_k\}}\ell_{k,i} > (1+\epsilon)k\beta\mathtt{A}\right] \\ \leq &(\beta+1)^{k(1+\epsilon')}\Pr[Y>(1+\epsilon')\mathtt{A}k\beta] + e^{-\Omega(k)} \\ \leq &\Pr[Y>(1+\epsilon')k\beta] + e^{-\Omega(k)} \\ = &e^{-\Omega(k)} \end{aligned}$$

□

A.3 Security analysis for the basic version of core-chain protocol

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 longest core-chain. Let $\beta_0 = \rho np$ be the expected number of malicious stakeholders that are qualified in a round to extend any chosen core-chain. Let α and β be the effective counterparts, respectively in the network delay setting. Here we assume $np \ll 1$. This means the expected number of stakeholders that are qualified to extend a core-chain in a round is much less than 1. Additionally, we assume that $\alpha_0 = \lambda\beta_0$ where $\lambda \in (1, \infty)$.

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

Theorem A.6 (Theorem 1.1, restated)**.** *Consider core-chain protocol Π^{core} where all players follow the simple strategy of extending the longest chain; in addition, all players have their stake registered before the protocol execution starts. Let α and β be the effective expected number of blocks generated by honest and malicious players in a round respectively. If $\alpha = \lambda\beta$, $\lambda > 1$, then the protocol Π^{core} can achieve chain growth, chain quality, common prefix and chain soundness properties.*

Basic terms Before giving the details of the security analysis, we define two terms, *public chain* and *honest successful round*, as follows.

Definition A.7 (Public chain)**.** *Consider a round r. We say a chain $\mathcal{C}$ is a* public chain in round r *if such chain $\mathcal{C}$ is known by all honest players in round r.*

Definition A.8 (Honest successful round)**.** *We say a round r is an* honest successful *round, if in the round r, at least one honest PoS-player is selected to extend the core-chain.*

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

A.3.1 Analysis with bounded delay

We assume that the malicious parties can delay messages up to Δ number of rounds. When an honest PoS-player is qualified to generate a new PoS block-core, he will broadcast it to the network and expect all parties to receive it. The honest player may not obtain the best PoS core-chain and thus work on a different PoS core-chain. If an honest player generates a new PoS block-core during the delay time and later receives a better PoS block-core from the network, then this generated PoS block-core will become useless and thus this honest player's effort during the time window is wasted. In this subsection, we provide a formal analysis for our core-chain protocol in the presence of the network delay.

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

- ω is the randomness in the execution,
- Messages of honest players may be delayed by at most Δ rounds.

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

- The randomness is fixed to ω as in $\mathsf{HYB}^r(\omega)$,
- The network delays all messages generated by *honest* PoS-players is exact Δ rounds,
- Remove all new messages sent by the adversary to honest players, and delay currently undelivered messages from corrupted parties to the maximum of Δ rounds,
- Whenever a chain is being delayed, no *honest* PoS-players make attempts to extend that chain

In $\mathsf{REAL}(\omega)$, the number of honest successful rounds is not less than that in the $\mathsf{HYB}^r(\omega)$. The following lemma shows that the chain growth rate in the real execution is not lower than that in the hybrid execution.

In order to distinguish core-chain in $\mathsf{HYB}^r(\omega)$ with in $\mathsf{REAL}(\omega)$ executions, we use $\mathcal{C}_{\text{hybrid}}$ to denote it.

Claim A.9. *Consider two executions* $\mathsf{REAL}(\omega)$ *and* $\mathsf{HYB}^r(\omega)$ *for all* ω, r. *For any honest PoS-player* P *at round* r', *where* $r' > r$, *let* $\mathcal{C}'$ *denote the PoS core-chain of* P *at round* r' *in the execution* $\mathsf{REAL}(\omega)$ *and* $\mathcal{C}'_{\text{hybrid}}$ *denote the PoS core-chain of* P *at round* r' *in the execution* $\mathsf{HYB}^r(\omega)$. *Then we have* $\mathsf{len}(\mathcal{C}') \geq \mathsf{len}(\mathcal{C}'_{\text{hybrid}})$.

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

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

Claim A.10. *Consider* $\mathsf{HYB}^r(\omega)$ *where the adversary is allowed to delay messages for at most* Δ *rounds. Let* $\alpha_0 > 0$ *be the expected number of honest stakeholders that are chosen in a round. Let* α *be the actual probability that a round* $s > r$ *is an honest successful round. Then we have that* $\alpha = \frac{\alpha_0}{1+\Delta\alpha_0}$.

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

Let $\mathtt{VIEW}^r$ denote the $\mathtt{VIEW}$ at round r in $\mathsf{REAL}(\omega)$ where $r > 0$. Let $\mathsf{len}(\mathtt{VIEW}^r)$ 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 $\mathsf{HYB}^r(\omega)$.

Claim A.11. *Consider* $\mathsf{HYB}^r(\omega)$. *For any honest successful round* s, *where* $s > r$, *it holds that* $\mathsf{len}(\mathtt{VIEW}^{s+\Delta}) - \mathsf{len}(\mathtt{VIEW}^s) \geq 1$.

Proof. By Definition A.8, there is at least one honest PoS-player producing a PoS block-core at round s. Let $\mathcal{C}^s_{\text{hybrid}}$ be the PoS core-chain that is extended by the PoS-player at round s. We have $\mathsf{len}(\mathcal{C}^s_{\text{hybrid}}) \geq \mathsf{len}(\mathtt{VIEW}^s)$. At the end of round s the honest player will broadcast the extended chain with length $\mathsf{len}(\mathcal{C}^s_{\text{hybrid}}) + 1$. At the end of round $s+\Delta$, all honest players will receive the extended core-chain, we have $\mathsf{len}(\mathtt{VIEW}^{s+\Delta}) \geq \mathsf{len}(\mathcal{C}^{s+\Delta}_{\text{hybrid}}) = \mathsf{len}(\mathcal{C}^s_{\text{hybrid}}) + 1$. Putting them together, we have $\mathsf{len}(\mathtt{VIEW}^{s+\Delta}) - \mathsf{len}(\mathtt{VIEW}^s) \geq 1$. □

Corollary A.12. *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* $\mathsf{len}(\mathtt{VIEW}^{r+t+\Delta}) - \mathsf{len}(\mathtt{VIEW}^r) \geq h$.

Proof. Let r_k be the kth honest successful round where $r < \mathtt{r}_k < r+t$ and $1 \leq k \leq h$. From Claim A.11, we have $\mathsf{len}(\mathtt{VIEW}^{\mathtt{r}_k+\Delta}) - \mathsf{len}(\mathtt{VIEW}^{\mathtt{r}_k}) \geq 1$. Then we have $\mathsf{len}(\mathtt{VIEW}^{r+t}) - \mathsf{len}(\mathtt{VIEW}^r) \geq \sum_{i=1}^{h} (\mathsf{len}(\mathtt{VIEW}^{\mathtt{r}_k+\Delta}) - \mathsf{len}(\mathtt{VIEW}^{\mathtt{r}_k})) \geq h$. □

A.3.2 Achieving chain growth property

We here demonstrate that our core-chain protocol satisfies the growth property (Definition 2.1). The concrete statement to be proved can be found in Lemma 3.3.

Claim A.13. *Consider* $\mathsf{HYB}^r(\omega)$, *and* $\delta > 0$. *Let* X *be the number of honest successful rounds from round* r *to round* $r+t$, *where* $t>0$. *Then we have* $\Pr[X > (1-\delta)\alpha t] > 1 - e^{-\Omega(t)}$.

Proof. Based on Claim A.10, we have that, on average, there are αt number of honest successful rounds in any t consecutive rounds. By Chernoff bound, we have $\Pr[X \leq (1-\delta)\alpha t] \leq e^{-\delta^2\alpha t/2}$. Thus, we have $\Pr[X > (1-\delta)\alpha t] > 1 - e^{-\delta^2\alpha t/2} = 1 - e^{-\Omega(t)}$. □

Claim A.14. *Consider* $\mathsf{HYB}^r(\omega)$ *and* $\delta > 0$. *Consider an honest PoS-player* P *with the best PoS core-chain* $\mathcal{C}_{\text{hybrid}}$ *in round* r, *and an honest PoS-player* P' *with the best PoS core-chain* $\mathcal{C}'_{\text{hybrid}}$ *in round* r', *respectively, where* $r'-r \gg \Delta$. *Then we have*

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

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

Proof. First, we note that $\mathcal{C}_{\text{hybrid}}$ will be received by all honest players no later than round $r+\Delta$ because player P is honest. We have $\mathsf{len}(\mathcal{C}_{\text{hybrid}}) \leq \mathsf{len}(\mathtt{VIEW}^{r+\Delta})$. 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 Claim A.13, in any t consecutive rounds the number of honest successful round is more than $(1-\delta)\alpha t$ with the probability at least $1 - e^{-\Omega(t)}$. Together with Claim A.11 and Corollary A.12, we have $\mathsf{len}(\mathtt{VIEW}^{r'}) - \mathsf{len}(\mathtt{VIEW}^{r+\Delta}) \geq (1-\delta)\alpha t$. Chain $\mathcal{C}'_{\text{hybrid}}$ is an valid PoS core-chain accepted by an honest PoS-player P' at round r'. We have $\mathsf{len}(\mathcal{C}'_{\text{hybrid}}) \geq \mathsf{len}(\mathtt{VIEW}^{r'})$. Putting these together, we get $\mathsf{len}(\mathcal{C}'_{\text{hybrid}}) - \mathsf{len}(\mathcal{C}_{\text{hybrid}}) \geq \mathsf{len}(\mathtt{VIEW}^{r'}) - \mathsf{len}(\mathtt{VIEW}^{r+\Delta}) \geq (1-\delta)\alpha t$ with probability at least $1 - e^{-\Omega(t)}$. The corresponding growth rate is $g = (1-\delta)\alpha$. □

Reminder of Lemma 3.3. *Consider an execution of core-chain protocol* Π^{core}, *where an honest PoS-player* P_1 *is with best local core-chain* $\mathcal{C}_1$ *in round* r_1, *and an honest PoS-player* P_2 *with best local core-chain* $\mathcal{C}_2$ *in round* r_2, *and* $r_2 > r_1$. *Then we have* $\Pr\left[\mathsf{len}(\mathcal{C}_2) - \mathsf{len}(\mathcal{C}_1) \geq g \cdot t\right] \geq 1 - e^{-\Omega(t)}$ *where* $t = r_2 - r_1$, $g = (1-\delta)\alpha$, *and* $\delta > 0$.

Proof. In order to distinguish the notation clearly, we use $\mathcal{C}'_{\text{hybrid}}$ and $\mathcal{C}''_{\text{hybrid}}$ to denote the PoS core-chains of the best core-chains of P at round r' and r'' in the execution of $\mathsf{HYB}^r(\omega)$. From Claim A.14, we have $\Pr[\mathsf{len}(\mathcal{C}''_{\text{hybrid}}) \geq \mathsf{len}(\mathcal{C}'_{\text{hybrid}}) + g \cdot t] \geq 1 - e^{-\Omega(t)}$ where $t = r'' - r'$, in $\mathsf{HYB}^r(\omega)$. We now turn to the core-chain growth property in $\mathsf{EXEC}_{\Pi^{\text{core}},\mathcal{A},\mathcal{Z}}$. From the definition of hybrid execution, we know that all honest players have same initial status at round r'. We have $\mathsf{len}(\mathcal{C}') = \mathsf{len}(\mathcal{C}'_{\text{hybrid}})$. By Claim A.9, we have $\mathsf{len}(\mathcal{C}'') \geq \mathsf{len}(\mathcal{C}''_{\text{hybrid}})$. It follows that,

$$\begin{aligned} &\Pr[\mathsf{len}(\mathcal{C}'') \geq \mathsf{len}(\mathcal{C}') + g \cdot t] \\ \geq\ &\Pr[\mathsf{len}(\mathcal{C}''_{\text{hybrid}}) \geq \mathsf{len}(\mathcal{C}'_{\text{hybrid}}) + g \cdot t] \\ \geq\ &1 - e^{-\Omega(t)} \end{aligned} \tag{4}$$

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

A.3.3 Achieving common prefix property

We now turn to proving the common prefix property (Definition 2.2) for the core-chain protocol Π^{core}. The concrete statement can be found in Lemma 3.4.

Intuitively, from the assumption that honest players control more resource than the malicious ones, we can see that 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. From the assumption $\alpha + \beta^\circ \ll 1$, we see that in most rounds no new block will be generated. This means all honest players will have the same view in most rounds. Since all honest players will extend the same public chain, the public chain will be the longest one.

Actually, we do not need to assume the strategy of malicious players. The common prefix property holds for all adversary model. We only give the proof idea in this section. The formal proof can be found in the section 4.3.

Reminder of Lemma 3.4. *Assume that $\alpha = \lambda\beta^\circ$, and $\lambda > 1$. Consider an execution of core-chain protocol Π^{core}, where two honest PoS-players, P in round r and P_1 in round r_1, with the local best core-chains $\mathcal{C}$ and $\mathcal{C}_1$, respectively, where $r_1 \geq r$. Then we have $\Pr[\mathcal{C}[\neg\kappa] \preceq \mathcal{C}_1] \geq 1 - e^{-\Omega(\kappa)}$.*

The main proof ideas are as follows: Considering from round r to r_1, there are $r_1 - r$ rounds totally. Let $\mathcal{C}_1$ be the best chain in round r_1. We will argue that the malicious players can not fork a chain $\mathcal{C}$ from round r with almost same length of $\mathcal{C}_1$. For the network delay Δ is small, in most rounds all of the honest players will take the same best chain. This means that in most rounds the honest players will try to extend the same chain. That is only one chain will be extended in a round by honest players. Putting these together, at least one chain of $\mathcal{C}_1$ and $\mathcal{C}$ will not be extended in more than $\frac{r_1 - r}{2}$ rounds. WLOG, we assume it is $\mathcal{C}$. In these rounds, the best chain will be extended by honest layers and $\mathcal{C}$ will be extended only by malicious players.

We assume $\alpha = \lambda\beta$, where $\lambda > 1$. If $r_1 - r$ is long, the malicious player can not keep $\mathcal{C}$ be extended with same growing rate with best chain. First, we will build the relationship between length of a core-chain and the number of rounds.

Claim A.15. *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 $c = \frac{1}{\alpha + \beta^\circ}$.*

Proof. All players can extend $\alpha + \beta^\circ$ number of PoS block-cores in a round on average. In order to generate ℓ block-cores, it will consume $\frac{\ell}{\alpha+\beta^\circ}$ rounds on average. Let $c = \frac{1}{\alpha+\beta^\circ}$, 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. □

Definition A.16 (Divergent length)**.** *Given two different core-chain $\mathcal{C}_1$ and $\mathcal{C}_2$. Let B be the last common block on $\mathcal{C}_1$ and $\mathcal{C}_2$. Let ℓ_1 be the length from B to the end of $\mathcal{C}_1$ and ℓ_2 be the length from B to the end of $\mathcal{C}_2$. The divergent length of $\mathcal{C}_1$ and $\mathcal{C}_2$ is $\ell = max\{\ell_1, \ell_2\}$.*

Claim A.17. *Let $\alpha = \lambda\beta^\circ$, $\lambda > 1$ and $(\alpha + \beta^\circ)\Delta \ll 1$, exists $\delta > 0$. Consider $\mathsf{REAL}(\omega)$. Let $\mathcal{C}$ be the best public core-chain in round r. Let $\mathcal{C}_1$ be another valid core-chain which is different with $\mathcal{C}$. Let ℓ be the divergent length of $\mathcal{C}$ and $\mathcal{C}_1$. We have $\Pr[\mathsf{len}(\mathcal{C}) - \mathsf{len}(\mathcal{C}_1) > (1-\delta)\ell] > 1 - e^{-\Omega(\ell)}$.*

Proof. Suppose the last common block-core of $\mathcal{C}$ and $\mathcal{C}_1$ is generated in round $s = r - t$. From Claim A.15, we have $t > (1-\delta)\frac{\ell}{\alpha+\beta^\circ}$ with probability no less than $1 - e^{-\Omega(\ell)}$. Let $X = \mathsf{len}(\mathcal{C}) - \mathsf{len}(\mathcal{C}^s)$ be the length growth of best public core-chain in the t rounds, with Lemma 3.3, 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 $(\alpha + \beta^\circ)t$ block-cores which are longer than core-chain $\mathcal{C}^s$ on average. With the network delay, this will confuse the honest players $(\alpha + \beta^\circ)\Delta t$ 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 $Y = (\alpha + \beta^\circ)\Delta t\alpha$ on average. For $(\alpha + \beta^\circ)\Delta \ll 1$, 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 $Z = \beta^\circ t$ on average. With Chernoff bounds, we have $Z < (1+\delta)\beta^\circ t$ with probability no less than $1 - e^{-\Omega(t)}$. Putting these together, we have $\Pr[X - (Y+Z) > (1-\delta)\frac{\lambda-1}{\lambda+1}\ell] > 1 - e^{-\Omega(t)} = 1 - e^{-\Omega(\ell)}$. For $\lambda > 1$, we have $\Pr[\mathsf{len}(\mathcal{C}) - \mathsf{len}(\mathcal{C}_1) > (1-\delta)\ell] = \Pr[X - (Y+Z) > (1-\delta)\ell] > 1 - e^{-\Omega(\ell)}$. This completes the proof. □

A.3.4 Achieving chain quality property

The chain-quality property (Definition 2.3) ensures that the ratio of blocks from honest players in a continuous part of longest core-chain has a lower bound. We show that the number of block-cores produced by the adversarial miners is bounded by the number of their stakes. 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.

We note that the discussion of chain-quality property is only for the best chain. From the chain growth property, we know that network delay will discount the effective honest stakes. In the following proof, when we calculate the length of chain we use the α in stead of α_0.

Now we consider the contribution from honest players in any consecutive block-cores. If the adversarial players want to contribute more PoS block-cores on the core-chain, they will try to generate more PoS block-cores and beat the PoS block-cores from honest players in the competition. Thus, the worst case is the adversarial players make use of all the stakes to generate PoS block-cores and win all of the competition. First, we will prove the core-chain quality property in any t consecutive rounds.

Claim A.18. *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* $\alpha = \lambda\beta^\circ$ *where* $\lambda > 1$*. Then we have* $\Pr[\mu \geq 1 - (1+\delta)\frac{1}{\lambda}] > 1 - e^{-\Omega(t)}$ *for any* $\delta > 0$*, where* μ *is the ratio of honest block-cores of the PoS core-chain* $\mathcal{C}$*.*

Proof. Consider the ℓ consecutive PoS block-cores of $\mathcal{C}$ that are generated from round r to round $r+t$. From Lemma 3.3, 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

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

We then have

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

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

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

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

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

Reminder of Lemma 3.5. *Assume* $\alpha = \lambda\beta^\circ$*, and* $\lambda > 1$*. Consider an execution of core-chain protocol* Π^{core}*, where an honest PoS-player is with core-chain* $\mathcal{C}$*. If among* ℓ *consecutive block-cores in* $\mathcal{C}$*, there are* ℓ_{good} *block-cores that are generated by honest PoS-players, then we have* $\Pr\left[\frac{\ell_{\mathrm{good}}}{\ell} \geq \mu\right] \geq 1 - e^{-\Omega(\ell)}$ *where* $\mu = 1 - (1+\delta)\frac{1}{\lambda}$*, and* $\delta > 0$*.*

Proof. Let t be the number of rounds that the ℓ block-cores are generated. From Claim A.15, we have $\Pr[t > (1-\delta)c\ell] > 1 - e^{-\Omega(\ell)}$. From Claim A.18, 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. □

B Supplemental Material for Section 4

Here, we present the full security analyze of our protocol $\Pi^{\text{core}\circ}$.

B.1 Achieving chain growth with $\Delta = 1$

We first analyze the chain growth in the execution where all messages from honest players can be broadcasted to all other honest nodes in $\Delta = 1$. Here, we use the Markov chain model that we described in Section 4.3.1. As we mentioned, the chain growth rate in each round is at least equal to the stationary probability of state ⓪ of the Markov chain. Thus, to analyze the chain growth property, we first compute the stationary probability q_0 of state ⓪. Then, we use a concentration bound in [21] to prove the chain growth property of protocol $\Pi^{\text{core}\circ}$.

B.1.1 Expected chain growth

Before we compute stationary probability of state ⓪, we compute the expect number of longest chains $f_{i,\mathtt{D}}$ for all $i \geq 0$. Recall that at round r, a new longest chain of length ℓ is generated. Let $f_{i,j}$ (i, j is integers such that $i \geq 0, 0 \leq j \leq \mathtt{D}$) be the expected number of chains with length $\ell - \mathtt{D} + j$ at round $r + i$. Here, we consider the worst case for the honest players in which when a new longest chain is generated at round r, the best chain set only consists of the longest chain as its prefix. Thus, we have

$$\begin{cases} f_{0,j} \geq 1, \\ f_{i,0} \geq 1, \\ f_{i,j} = f_{i-1,j} + f_{i-1,j-1} \cdot \alpha, \forall i > 0, j > 0. \end{cases} \tag{5}$$

Hence,

$$f_{i,j} \geq 1 + \sum_{k=1}^{\min(i,j)} \binom{i}{j} \alpha^k \tag{6}$$

Now, we are ready to compute compute stationary probability of state ⓪. From Eq. 1, we have,

$$\begin{cases} \sum_{i=0}^{m} \mathsf{q}_i = 1, \\ \mathsf{q}_0 = 1 - \sum_{i=1}^{m} \frac{\mathsf{q}_i}{\mathsf{q}_0} \cdot \alpha \cdot (f_{i,\mathtt{D}} - 1), \\ \mathsf{q}_i = \mathsf{q}_0 \prod_{j=0}^{i-1} (1 - \alpha \cdot f_{j,\mathtt{D}}), \forall 1 \leq i < m, \\ \mathsf{q}_m = \mathsf{q}_{m-1} \cdot (1 - (1 - \alpha \cdot f_{m-1,\mathtt{D}})) \\ \qquad\quad + \mathsf{q}_m \cdot (1 - (1 - \alpha \cdot f_{m,\mathtt{D}})) \end{cases} \tag{7}$$

We now compute the stationary probability q_0 when $\mathtt{D} = 0, 1, 2$ as follows.

The case: $\mathtt{D} = 0$. We have, $f_{i,0} = 1, \forall i \geq 0$. Replace this to the third equation in Eq. 7, we have

$$\frac{\alpha}{\mathsf{q}_0} = 1.$$

Thus, $\hat{\mathtt{A}}_0^{\circ} = \frac{\mathsf{q}_0}{\alpha} = 1$

The case: $\mathtt{D} = 1$. We have, $f_{i,1} = 1 + i\alpha, \forall i \geq 0$. Replace this to the second equation in Eq. 7, we have

$$\frac{\alpha}{\mathsf{q}_0} = 1 - \sum_{i=1}^{m} \frac{\mathsf{q}_i}{\mathsf{q}_0} \cdot \alpha \cdot (i\alpha)$$

Let $A_1 = \sum_{i=1}^{m} \left(\frac{\mathsf{q}_i}{\mathsf{q}_0} \cdot i\alpha^2 \right) \geq \sum_{i=1}^{m_1} \left(\frac{\mathsf{q}_i}{\mathsf{q}_0} \cdot i\alpha^2 \right)$, where $m > m_1 = a_1/\alpha$ and $a_1 \approx 1.2$.
From the second equation of Eq. 7, we have

$$\frac{\mathsf{q}_i}{\mathsf{q}_0} \approx (1 - i\alpha + \frac{i^3}{3}\alpha^3 - \frac{i^4}{12}\alpha^4 - \frac{i^5}{12}\alpha^5)$$

We have

$$\begin{aligned}
A_1 &\geq \sum_{i=1}^{m_1} \left(\frac{\mathsf{q}_i}{\mathsf{q}_0} \cdot i\alpha^2 \right) \\
&\approx \sum_{i=1}^{m_1} \left((1 - i\alpha + \frac{i^3}{3}\alpha^3 - \frac{i^4}{12}\alpha^4 - \frac{i^5}{12}\alpha^5) \cdot i\alpha^2 \right) \\
&= \sum_{i=1}^{m_1} \left(i\alpha^2 - i^2\alpha^3 + \frac{i^4}{3}\alpha^5 - \frac{i^5}{12}\alpha^6 - \frac{i^6}{12}\alpha^7 \right) \\
&\approx \frac{m_1^2}{2}\alpha^2 - \frac{m_1^3}{3}\alpha^3 + \frac{m_1^5}{15}\alpha^5 - \frac{m_1^6}{72}\alpha^6 - \frac{m_1^5}{84}\alpha^7.
\end{aligned}$$

Replace $m_1 = a_1/\alpha$, we have,

$$A_1 \geq \frac{a_1^2}{2} - \frac{a_1^3}{3} + \frac{a_1^5}{15} - \frac{a_1^6}{72} - \frac{a_1^7}{84} \approx 0.23.$$

Thus, the amplification ratio

$$\hat{\mathtt{A}}_1^{\circ} = \frac{\mathsf{q}_0}{\alpha} = \frac{1}{1 - A_1} \geq \frac{1}{1 - 0.23} \approx 1.3.$$

Here, we require 67.7% honest stake for the protocol to be secure.

The case: $\mathtt{D} = 2$. We have, $f_{i,2} = 1 + i\alpha + \frac{i \cdot (i-1)}{2}\alpha^2, \forall i \geq 0$. Replace this to the second equation in Eq. 7, we have

$$\begin{aligned}
\frac{\alpha}{\mathsf{q}_0} &= 1 - \sum_{i=1}^{m} \frac{\mathsf{q}_i}{\mathsf{q}_0} \cdot \alpha \cdot (i\alpha + \frac{i \cdot (i-1)}{2}\alpha^2) \\
&= 1 - \sum_{i=1}^{m} \prod_{j=0}^{i-1} (1 - \alpha - j\alpha^2 - \frac{i \cdot (i-1)}{2}\alpha^3) \\
&\qquad \cdot \alpha \cdot (i\alpha + \frac{i \cdot (i-1)}{2}\alpha^2) \\
&\leq 1 - \sum_{i=1}^{m} \prod_{j=0}^{i-1} (1 - \alpha - j\alpha^2 - \frac{i \cdot (i-1)}{2}\alpha^3) \frac{i \cdot (i-1)}{2}\alpha^3 \\
&\qquad - A_1
\end{aligned}$$

Let

$$A_2 = \sum_{i=1}^{m} \prod_{j=0}^{i-1} (1-\alpha-j\alpha^2-\frac{i\cdot(i-1)}{2}\alpha^3)\frac{i\cdot(i-1)}{2}\alpha^3$$
$$\geq \sum_{i=1}^{m_2} \prod_{j=0}^{i-1} (1-\alpha-j\alpha^2-\frac{i\cdot(i-1)}{2}\alpha^3)\frac{i\cdot(i-1)}{2}\alpha^3.$$

where $m > m_2 = a_2/\alpha$ and $a_2 \approx 1.1$.

We have,

$$\prod_{j=0}^{i-1} (1-\alpha-j\alpha^2-\frac{i\cdot(i-1)}{2}\alpha^3)$$
$$\approx (1-i\alpha+\frac{i^3}{6}\alpha^3-\frac{i^4}{24}\alpha^4-\frac{i^5}{24}\alpha^5)$$

Thus,

$$A_2 \geq \sum_{i=1}^{m_1} \left(\prod_{j=0}^{i-1} (1-\alpha-j\alpha^2-\frac{i\cdot(i-1)}{2}\alpha^3) \cdot \frac{i(i-1)}{2}\alpha^3 \right)$$
$$\approx \sum_{i=1}^{m_1} \left((1-i\alpha+\frac{i^3}{6}\alpha^3-\frac{i^4}{24}\alpha^4-\frac{i^5}{24}\alpha^5) \cdot \frac{i(i-1)}{2}\alpha^3 \right)$$
$$\approx \frac{1}{2} \sum_{i=1}^{m_1} \left(i^2\alpha^3 - i^3\alpha^4 + \frac{i^5}{6}\alpha^6 - \frac{i^6}{24}\alpha^7 - \frac{i^7}{24}\alpha^8 \right)$$
$$\approx \frac{1}{2} \left(\frac{m_2^2}{3}\alpha^2 - \frac{m_2^3}{4}\alpha^3 + \frac{m_2^5}{36}\alpha^5 - \frac{m_2^6}{168}\alpha^6 - \frac{m_2^5}{192}\alpha^7 \right).$$

Replace $m_2 = a_2/\alpha$, we have,

$$A_2 \geq \frac{1}{2} \left(\frac{a_2^2}{3} - \frac{a_2^3}{4} + \frac{a_2^5}{36} - \frac{a_2^6}{168} - \frac{a_2^7}{192} \right) \approx 0.05.$$

Thus, the amplification ratio

$$\hat{\mathtt{A}}_2^{\circ} = \frac{\mathtt{q}_0}{\alpha} = \frac{1}{1-A_1-A_2} \geq \frac{1}{1-0.23-0.05} \approx 1.39. \tag{8}$$

Here, we require 66.2% honest stake for the protocol to be secure.

B.1.2 Useful definitions for the random walk on a Markov chain

Here, we introduce some useful definitions in a Markov chain that we will use to prove the chain growth property. We recall that a Markov chain has a finite set of states V. For each pair of states $\textcircled{i}$ and $\textcircled{j}$, there is a transition probability $f_{i,j}$ of going from state $\textcircled{i}$ to state $\textcircled{j}$, where for each i, $\sum_{j\in V} f_{i,j} = 1$. A random walk in the Markov chain starts at some state s_0. At a given time step, if it is in state s_i, the next state s_{i+1} is selected randomly with probability $f_{s_i,s_{i+1}}$.

Normalized conductance. Let S be a subset of states, the normalized conductance $\Phi(S)$ of S ratio between the total conductance of all edges from S to $\bar{S} = V \setminus S$ (the complement set of S) over the total of conductance of all nodes in S, i.e.,

$$\Phi(S) = \frac{\sum_{i \in S, j \in \bar{S}} \mathsf{q}_i f_{i,j}}{\sum_{i \in S} \mathsf{q}_i}$$

The normalized conductance of the Markov chain, denoted as Φ, is defined by

$$\Phi = \min_S \Phi(S)$$

Mixing time. For a given $\varepsilon > 0$, the ε-mixing time of a Markov chain is the minimum integer T such that for any starting distribution, the 1-norm distance between the t-step running average probability distribution and the stationary distribution is at most ε. The ε-mixing time of a Markov chain is bounded as follows.

Lemma B.1. *The ε-mixing time of a random walk on a Markov chain is*

$$O\left(\frac{\ln(1/\mathsf{q}_{\min})}{\Phi^2 \varepsilon^3}\right),$$

where $\mathsf{q}_{\min}$ is the minimum minimum stationary probability of any state.

Lemma B.2 (Theorem 3 in [21])**.** *Let M be an Markov chain with state space $[m]$ and stationary distribution Q. Let $T = T(\varepsilon)$ be its ε-mixing time for $\varepsilon \geq 1/8$. Let $(s_1, \cdots, s_t)$ denote a t-step random walk on M starting from an initial distribution φ on $[m]$, i.e., $s_1 \leftarrow \varphi$. Let $g : [m] \to [0, 1]$ be a weight function such that the expected weight $\mathbb{E}_{s \leftarrow \varphi}[g(s)] = \mu$ for all i. Define the total weight of the walk $(s_1, \cdots, s_t)$ by $X \triangleq \sum_{i=1}^{t} f_i(s_i)$. There exists some constant c (which is independent of μ, δ and ε) such that*

$$\Pr[X < (1-\delta)\mu t] < c \cdot \|\varphi\|_{\mathsf{Q}} \cdot e^{-\delta^2 \mu t/(72T)}, \text{for} 0 < \delta < 1.$$

where, $\|\varphi\|_{\mathsf{Q}} = \sum_{i \in [m]} \varphi_i^2 / \mathsf{q}_i$.

B.1.3 Proof of chain growth property

Now, we ready to proof the chain growth property for our protocol $\Pi^{\text{core}\circ}$.

Lemma B.3 (Chain growth)**.** *Consider core-chain protocol $\Pi^{\text{core}\circ}$ in the presence of an arbitrary adversary in which the network delay $\Delta = 1$. Consider an honest PoS-player P_1 with best local PoS core-chain $\mathcal{C}_1$ in round r_1, and an honest PoS-player P_2 with best local core-chain $\mathcal{C}_2$ in round r_2, where $r_2 > r_1$. Then we have $\Pr\left[\mathsf{len}(\mathcal{C}_2) - \mathsf{len}(\mathcal{C}_1) \geq g \cdot t\right] \geq 1 - e^{-\Omega(t\alpha)}$ where $t = r_2 - r_1$, $g = (1-\delta)\alpha_0^\circ$, $\alpha_0^\circ = \hat{\mathtt{A}}_{\mathtt{D}}^\circ \alpha$ ($\hat{\mathtt{A}}_2^\circ = 1.39$), and $\delta > 0$.*

Proof. At round $r_1 + 1$ the random walk starts at a state $s_1 \leftarrow \mathsf{Q}$ where Q is the stationary distribution. Note that, the random walk goes to state ⓪ when a new longest chain is generated. In other words, the increasing length of the longest chain from round r_1 to round r_2 equal the number of appearances of state ⓪ in the random walk. Thus, we define the weight function $g : [0, m] \to [0, 1]$ for the random walk to represent the chain growth as follows,

$$g(s) = \begin{cases} 1 \text{ if } s = 0, \\ 0 \text{ if } s \neq 0. \end{cases}$$

We have

$$\mathbb{E}_{s \leftarrow \mathsf{Q}}[g(s)] = \mathsf{q}_0.$$

Let $X \triangleq \sum_{i=1}^{t} f_i(s_i)$ be the increasing length of the longest chain. From Lemma B.2, we have,

$$\Pr[X < (1-\delta)\mathsf{q}_0 t] < c \cdot \|\mathsf{Q}\|_{\mathsf{Q}} \cdot e^{-\delta^2 \mathsf{q}_0 t/(72T)},$$

where c is a constant, T is ε-mixing time, and $\varepsilon > 1/8$.

Here, we have

$$\|\mathsf{Q}\|_{\mathsf{Q}} = O(1), \text{ and } T = O(1),$$

Thus, we have,

$$\Pr[X < (1-\delta)\mathsf{q}_0 t] < e^{-\Omega(t\alpha)},$$

□

B.2 Achieving chain growth with arbitrary Δ

We now modify the Markov chain to deal with the network delay. Intuitively, we add $\Delta-1$ new states to deal with the network delay. Jumping ahead, the network delay with reduce the chain growth by a fraction of $\frac{1}{1+(\Delta-1)\hat{\mathsf{A}}_{\mathsf{D}}^{\circ}\alpha}$.

We remark that, here, we analyze the chain growth in the worst delay setting in the hybrid experiment in Appendix A.3.1. Here, the network delay of all messages generated by honest player is exact Δ. Then, from Claim A.9, we can show that the chain growth of our protocol is lower bounded by the chain growth in the worst delay setting.

Representing chain extension in the presence of network delay, via Markov chain. Similar to the Markov chain in Fig 3, the Markov chain that represents the chain extension process (see Fig. 6) consists of $m+1+\Delta$ states ⓪,①,$\cdots$,ⓜ and ①,$\cdots$,Δ, where $\Delta' = \Delta - 1$, $m = a_{\mathsf{D}}/\alpha$ is an integer and $a_{\mathsf{D}} \geq 1$ is a constant.

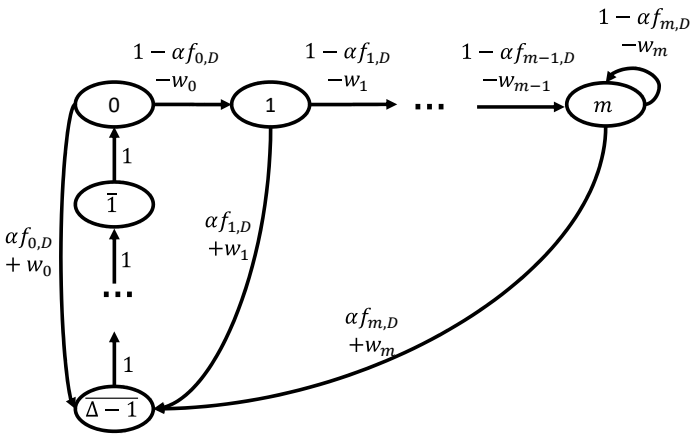

Figure 6: The Markov chain that represents the chain extension process. State i represents the i-th round after a new longest chain is generated. At state i, with probability $\alpha f_{i,\mathsf{D}}$, the honest players can extend the longest chain and we move to state Δ (where $\Delta' = \Delta - 1$). With some probability $w_i \geq 0$, the adversary publishes a new longest chain and we move to state Δ.

For any state i, the transition probability of going from state i to state ⓪ is $\alpha f_{i,\mathsf{D}} + w_i$, where $w_i \geq 0$. Here, the adversary publish a new longest chain at state i with probability $w_i \geq 0$. Plus, the probability that an honest player generates new longest chain is $\alpha f_{i,\mathsf{D}}$. As we mentioned above, when a new longest chain is generated, we move to the state Δ (where $\Delta' = \Delta - 1$) in the next round. In each delayed stated i, with probability 1 we move to the state j (where $j = i - 1$) after each round.

With probability $1 - \alpha f_{i,\mathsf{D}} - w_i$, there is no new longest chain is generated. Thus, for any state i (where $0 \leq i < m$), the transition probability of going from state i to state j (where $j = i + 1$) is $1 - \alpha f_{i,\mathsf{D}} - w_i$. For state m, the transition probability of going from state m to state m is $1 - \alpha f_{m,\mathsf{D}} - w_m$.

Chain growth as a random walk on the Markov chain. We now model the honest chain extend process from round r_1+1 to round r_2 ($r_2-r_1=t>0$) as a random walk $s_1,\cdots,s_t$. We start at round r_1+1 at state s_1. At round r_1+i, state s_i is randomly selected transition probabilities of going from state s_{i-1} to other states.

Let $\mathsf{q'}_i$ be the stationary probability of the state i and $\mathsf{Q'}=[\mathsf{q'}_0,\cdots\mathsf{q'}_m]$ be the stationary distribution We obtain the following equations for the stationary distribution.

$$\begin{cases}\sum_{i=0}^{\infty}\mathsf{q'}_i+\sum_{i=1}^{\Delta-1}\mathsf{q'}_{\bar{i}}=1,\\ \mathsf{q'}_0=\sum_{i=0}^{\infty}\mathsf{q'}_i\cdot(w_i+\alpha\cdot f_{i,\mathtt{D}}),\\ \mathsf{q'}_{\bar{i}}=\mathsf{q'}_0,\forall 1\le i\le\Delta-1\\ \mathsf{q'}_i=\mathsf{q'}_{i-1}\cdot(1-(1-w_{i-1}-\alpha\cdot f_{i-1,\mathtt{D}})),\\ \qquad\qquad\forall 1\le i<m,\\ \mathsf{q'}_m=\mathsf{q'}_{m-1}\cdot(1-(1-w_{m-1}\alpha\cdot f_{m-1,\mathtt{D}}))\\ \qquad+\mathsf{q'}_m\cdot(1-(1-wm-\alpha\cdot f_{m,\mathtt{D}}))\end{cases}\tag{9}$$

Similar to the analysis in Appendix B.1, the chain growth rate in each round is at least equal to the stationary probability $\mathsf{q'}_0$ of state ⓪ of the Markov chain. From Eq. 7 and Eq. 9, we have, $\mathsf{q'}_0=(\frac{1}{1+(\Delta-1)\mathsf{q}_0})\mathsf{q}_0$.

Lemma B.4 (Chain growth)**.** *Consider core-chain protocol $\Pi^{\mathrm{core}\circ}$ in the presence of an arbitrary adversary. Consider an honest PoS-player P_1 with best local PoS core-chain $\mathcal{C}_1$ in round r_1, and an honest PoS-player P_2 with best local core-chain $\mathcal{C}_2$ in round r_2, where $r_2>r_1$. Then we have $\Pr\left[\mathsf{len}(\mathcal{C}_2)-\mathsf{len}(\mathcal{C}_1)\ge g\cdot t\right]\ge 1-e^{-\Omega(t\alpha)}$ where $t=r_2-r_1$, $g=(1-\delta)\alpha^\circ$, $\alpha^\circ=\frac{\hat{\mathtt{A}}_{\mathtt{D}}^{\circ}\alpha}{1+\Delta\hat{\mathtt{A}}_{\mathtt{D}}^{\circ}\alpha}$ ($\hat{\mathtt{A}}_2^{\circ}=1.39$), and $\delta>0$.*

Proof. At round r_1+1 the random walk starts at a state $s_1\leftarrow\mathsf{q'}$ where $\mathsf{q'}$ is the stationary distribution of the Markov chain. Note that, the random walk goes to state ⓪ went a new longest chain is generated. In other words, the increasing length of the longest chain from round r_1 to round r_2 equal the number of appearances of state ⓪ in the random walk. Thus, we define the weight function $g:[0,m]\to[0,1]$ for the random walk to represent the chain growth as follows,

$$g(s)=\begin{cases}1\text{ if }s=0,\\ 0\text{ if }s\neq 0.\end{cases}$$

We have

$$\mathbb{E}_{s\leftarrow\mathsf{Q'}}[g(s)]=\mathsf{q'}_0.$$

Let $X\triangleq\sum_{i=1}^{t}f_i(s_i)$ be the increasing length of the longest chain. From Lemma B.2, we have,

$$\Pr[X<(1-\delta)\mathsf{q'}_0t]<c\|\mathsf{Q'}\|_{\mathsf{Q'}}e^{-\delta^2\mathsf{q'}_0t/(72T)},$$

where c is a constant, T is ε-mixing time, and $\varepsilon>1/8$.

Here, we have

$$\|\mathsf{Q'}\|_{\mathsf{Q'}}=O(1),\quad\text{and}\quad T=O(1),$$

Thus, we have,

$$\Pr[X<(1-\delta)\mathsf{q'}_0t]<e^{-\Omega(t\alpha)},$$

□

B.3 Achieving common prefix

Common prefix with respect to virtual chains We first analyze the common prefix property wrt virtual chains. We note that the virtual block sets will be updated through time. Thus, we override the equal operator of virtual block sets as follows.

Definition B.5 (Equal operator for virtual block sets)**.** *Consider two virtual block sets $\mathcal{V}_i$ and $\mathcal{V}'_i$ at the same block height i. We say $\mathcal{V}_i$ equals $\mathcal{V}'_i$ (i.e., $\mathcal{V}_i = \mathcal{V}'_i$) if the following constraint is satisfied.*

$$\forall B_i \in \mathcal{V}_i, \forall B'_i \in \mathcal{V}'_i, \mathsf{distance}(\mathcal{C}^{(B_i)} \rightarrow \mathcal{C}^{(B'_i)}) \leq \mathtt{D}.$$

Note that, by the definition of virtual block sets, the above constraint is equivalent to the following constraint

$$\forall B_i \in \mathcal{V}_i, \forall B'_i \in \mathcal{V}'_i, \mathsf{distance}(\mathcal{C}^{(B'_i)} \rightarrow \mathcal{C}^{(B_i)}) \leq \mathtt{D}.$$

Thus, the equal operator for virtual block sets is symmetric. We also write $\mathcal{V}[i] \neq \mathcal{V}'[i]$ if the above constraint is not satisfied.

Definition B.6 (Common prefix wrt virtual chains)**.** *Consider a blockchain protocol Π with a set $\mathcal{P}$ of players. The common prefix with respect to virtual chains, states the following: for any honest player P' adopting a local best virtual chain $\mathcal{V}'$ at round r', and honest player P adopting a local best virtual chain $\mathcal{V}$ at round r, in the execution $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$, where $P', P \in \mathcal{P}$ and $r \leq r'$, it holds that $\mathcal{V}[\neg\kappa] \preceq \mathcal{V}'$, where $\mathcal{V}[\neg\kappa]$ is the virtual chain resulting from removing the last κ blocks.*

Intuitively, the analysis of common prefix wrt virtual chain is similar to the analysis of common prefix in Bitcoin [54] and in protocol Π^{core} in Appendix A.3.3. Indeed, the key component to analyze common prefix is that for most of the time, the honest players only generate at most *one* block or virtual block set at a block height. This is quite straightforward in Bitcoin. Recall that the honest players always extend on the longest chain. Thus, the honest players only generate two blocks at the same block height if the player, who generates the later block, has not received the earlier block yet. As the network delay is relatively small (comparing with the time interval between two blocks), this event rarely occurs. Source text This is quite straightforward in Bitcoin. Recall that the honest players always extend on the longest chain. Thus, the honest players only generate two blocks at the same block height if the player, who generates the later block, has not received the earlier block yet. As the network delay is relatively small (comparing with the time interval between two blocks), this event rarely occurs. On the other hand, in our protocol, the honest players may extend from multiple chains. Hence, it is likely that the honest players may generate more than one block in a block height. Fortunately, by the definition of virtual block set, we can show that most of the time the honest players only generate one virtual block set at a block height (please see Lemma B.7 for the detail proof).

Similar to the argument in Appendix A.3.3, considering from round r to r_1, there are $r_1 - r$ rounds totally. Let $\mathcal{V}_1$ be the best chain in round r_1. We will argue that the malicious players can not fork a chain $\mathcal{V}$ from round r with almost same length of $\mathcal{V}_1$. We remark that, in protocol $\Pi^{\mathrm{core}\circ}$, the honest players extend a set of best chains instead of the best chain. Thus, when an honest player generates a new virtual chain, in the next Δ the adversary may minimize the network delay of some chains to increase the chance that an honest player generates a new different virtual chain at the same length. However, as the network delay Δ is small, in most rounds, all of the honest players still take the same best virtual chain. In other words, in most rounds the honest players will try to extend the same virtual chain. That is only one virtual chain will be extended in a round by honest players. Putting these together, at least one virtual chain of $\mathcal{V}_1$ and $\mathcal{V}$ will not be extended in more than $\frac{r_1 - r}{2}$ rounds. Without loss of generality, we assume it is $\mathcal{V}$. In these rounds, the best virtual chain will be extended by honest layers and $\mathcal{V}$ will be extended only by malicious players.

We assume $\alpha^\circ = \lambda\beta^\circ$, $\lambda > 1$, and $\delta > 0$. If $r_1 - r$ is long, the malicious player can not keep $\mathcal{V}$ be extended with same growing rate with best chain. Assume, $\Delta\alpha^\circ \ll 1$, most of the time, there is at most one honest virtual block set at a block height. Now, we are ready to prove the common prefix property on virtual chains.

Lemma B.7. *Consider an honest player P. Let $\mathcal{V}_{\text{best}} = V_0 \| V_1 \| \cdots \| V_\ell$ (ℓ is the length of the best chain) be the best virtual chain in the local state of player P at the beginning of round r. If player P generates a new block B at round r, with block height ℓ', i.e., $\mathsf{len}(\mathcal{C}^{(B)}) = \ell'$. Then, either $\ell' = \ell + 1$ (a new longest chain is generated) or $B \in V_{\ell'}$ (the new block belongs to an existing virtual block set).*

Proof. Let $\mathcal{C}_{\text{best}}$ be the best chain in the local state of player P at the beginning of round r. From definition 4.8, we have $\mathcal{C}_{\text{best}} \in \mathbb{C}^{(\mathcal{V}_{\text{best}})}$. Recall from procedure $\mathtt{D}$-BestCore$^\circ$ in Algorithm 4, the set of best chains is consists of all that chains $\mathcal{C}'$ in which the distance from the best chain $\mathbb{C}_{\text{best}}$ to $\mathcal{C}'$ is smaller than $\mathtt{D}$, i.e.,

$$\mathbb{C}_{\text{best}} = \{\mathcal{C}' | \mathsf{distance}(\mathcal{C}_{\text{best}} \to \mathcal{C}') \leq \mathtt{D}\}.$$

Let $\mathcal{C}$ be the chain that player P extend at round r by adding the block B. Since the honest players only extend the chains in the set of best chains, we have, $\mathcal{C} \in \mathbb{C}_{\text{best}}$. We consider two case of $\mathcal{C}$ as follows.

- $\mathsf{len}(\mathcal{C}) = \ell$. In this case, since the block B is extended from the chain $\mathcal{C}$, we have,

$$\ell' = \mathsf{len}(\mathcal{C}^{(B)}) = \mathsf{len}(\mathcal{C} \| B) = \ell + 1$$

- $\mathsf{len}(\mathcal{C}) < \ell$. Since $\mathsf{distance}(\mathcal{C}_{\text{best}} \to \mathcal{C}) \leq \mathtt{D}$, from Definition 4.1, we have,

$$\begin{aligned}
&\mathcal{C}_{\text{best}}[0, \ell - \mathtt{D}] \preceq \mathcal{C} \\
\Rightarrow &\mathcal{C}_{\text{best}}[0, \mathsf{len}(\mathcal{C}) + 1 - \mathtt{D}] \preceq \mathcal{C} \| B \\
\Rightarrow &\mathsf{distance}(\mathcal{C}_{\text{best}}[0, \mathsf{len}(\mathcal{C}) + 1] \to \mathcal{C} \| B) \leq \mathtt{D} \\
\Rightarrow &\mathsf{distance}(\mathcal{C}^{(\mathcal{C}_{\text{best}}[\ell'])} \to \mathcal{C}^{(B)}) \leq \mathtt{D}
\end{aligned}$$

  Plus, since $\mathcal{C}_{\text{best}}$ belongs to $\mathcal{V}_{\text{best}}$, we have $\mathcal{C}_{\text{best}}[\ell'] \in V_{\ell'}$. Thus, we have $B \in V_{\ell'}$.

□

Lemma B.8 (Common prefix wrt virtual chains)**.** *Assume $\alpha^\circ = \lambda\beta^\circ$, $\lambda > 1$, and $\delta > 0$. Consider an execution of core-chain protocol $\Pi^{\text{core}\circ}$ with an arbitrary adversary. Consider two honest players, P in round r with the local best virtual-chain $\mathcal{V}$, and P' in round r' with the local best virtual-chain $\mathcal{V}'$, respectively, where $r' \geq r$. Then we have $\Pr[\mathcal{V}[\neg\kappa] \preceq \mathcal{V}'] \geq 1 - e^{-\Omega(\kappa)}$.*

Proof. Assume, towards a contradiction, $\mathcal{V}[\neg\kappa] \not\preceq \mathcal{V}'$. Let r_0 be the round that the last common virtual block set of $\mathcal{V}$ and $\mathcal{V}'$ is generated. Since the length of a best virtual chain equals the length of the corresponding best chain, based on the chain growth property in Lemma B.4, from round r_0 to round r, the length of the virtual chain $\mathcal{V}$ increase by at least $\alpha^\circ t$, where $t = r - r_0$. We recall from Lemma B.7 that, there is at most one honest virtual block set at a block height. Thus, the adversary needs to create at least $\alpha^\circ t$ virtual block sets from round r_0 to round r. This happens with probability less than $e^{-\Omega(\kappa)}$. □

From common prefix w.r.t. virtual chain, to the standard common prefix property. Next, we prove common prefix property from common prefix wrt virtual chain. Consider the set of chains where the latest blocks in those chain belong to a virtual block set. By the definition of the virtual block set, all of those chains share the same common prefix after pruning the last $\mathtt{D}$ blocks. Thus, if a protocol achieves common prefix wrt virtual chain, it also achieves common prefix property by pruning extra $\mathtt{D}$ blocks.

Lemma B.9 (Common prefix)**.** *Assume $\alpha^\circ = \lambda\beta^\circ$, $\lambda > 1$, and $\delta > 0$. Consider an execution of core-chain protocol $\Pi^{\text{core}\circ}$ with an arbitrary adversary. Consider two honest players, P in round r with the local best core-chain $\mathcal{C}$, and P' in round r' with the local best core-chain $\mathcal{C}'$, respectively, where $r' \geq r$. Then we have $\Pr[\mathcal{C}[\neg(\kappa + \mathtt{D})] \preceq \mathcal{C}'] \geq 1 - e^{-\Omega(\kappa)}$.*

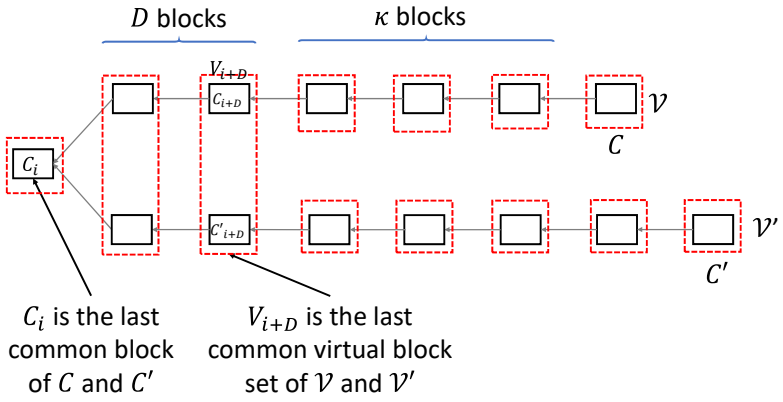

Figure 7: If common prefix property does not hold, i.e., $\mathcal{C}[\neg(\kappa+\mathtt{D})] \preceq \mathcal{C}'$, then common prefix wrt virtual chain property does not hold, i.e., $\mathcal{V}[\neg\kappa] \preceq \mathcal{V}'$. Here, $\mathcal{C}$ belongs to $\mathcal{V}$ and $\mathcal{C}'$ belongs to $\mathcal{V}'$.

Proof. Let $\mathcal{V}$ be the virtual chain of $\mathcal{C}$ and $\mathcal{V}'$ be the virtual chain of $\mathcal{C}'$. From Lemma B.8, we have $\Pr[\mathcal{V}[\neg\kappa] \preceq \mathcal{V}'] \geq 1 - e^{-\Omega(\kappa)}$. From this, we will prove that $\Pr[\mathcal{C}[\neg(\kappa+\mathtt{D})] \preceq \mathcal{C}'] \geq 1 - e^{-\Omega(\kappa)}$.

We will prove by contradiction. More concretely, we will prove that, if $\mathcal{C}[\neg(\kappa+\mathtt{D})] \not\preceq \mathcal{C}'$, then $\mathcal{V}[\neg\kappa] \not\preceq \mathcal{V}'$. Let $\mathcal{C}[i]$ be the last block after pruning $\kappa+\mathtt{D}$ blocks from $\mathcal{C}$ (please also see Figure 7). Since $\mathcal{C}[\neg(\kappa+\mathtt{D})] \not\preceq \mathcal{C}'$, we have

$$
\begin{aligned}
& \mathsf{distance}(\mathcal{C}[i] \rightarrow \mathcal{C}') > 0 \\
\Rightarrow\ & \mathsf{distance}(\mathcal{C}[i+\mathtt{D}] \rightarrow \mathcal{C}') > \mathtt{D} \\
\Rightarrow\ & V_{i+\mathtt{D}} \neq V'_{i+\mathtt{D}} \\
\Rightarrow\ & \mathcal{V}[\neg\kappa] \not\preceq \hat{\mathcal{V}}
\end{aligned}
$$

□

B.4 Achieving chain quality

A D-distance-greedy adversary can extend a chain faster than basic adversary, when $\mathtt{D} > 0$. Intuitively, this will reduce the chain quality. However, the number of blocks from malicious players on any chain is bounded. If we assume the honest players extend chains faster than the malicious players, the chain quality property will still hold as in Lemma 3.5.

Lemma B.10 (Chain quality). *Assume $\alpha^\circ = \lambda\beta^\circ$, $\lambda > 1$, and $\delta > 0$. Consider an execution of core-chain protocol $\Pi^{\text{core}\circ}$ with an arbitrary adversary. Consider an honest PoS-player with PoS core-chain $\mathcal{C}$. Consider that ℓ consecutive block-cores of $\mathcal{C}$, where ℓ_{good} block-cores are generated by honest PoS-players. Then we have $\Pr\left[\frac{\ell_{good}}{\ell} \geq \mu\right] \geq 1 - e^{-\Omega(\ell)}$ where $\mu = 1 - (1+\delta)\frac{1}{\lambda}$.*

Proof. We prove by contradiction. Assume all block from round r' to round r'' are generated by malicious players. From Lemma B.3, we have, the length of the longest chain from round r' to round round r'' increase by at least $(1-\delta)\alpha^\circ t$, where $t = r'' - r'$. As all blocks from round r' to round r'' are generated by malicious players, the adversary can grow the chain with the rate $(1-\delta)\alpha^\circ$. However, from Lemma 3.2, the adversary can grow the chain with the rate at most β° (a contradiction since $\beta^\circ < (1-\delta)\alpha^\circ$).

□

B.5 Empirical amplification ratio

We also run a simulation to compute the amplification ratio. Here, we simulate multiple greedy strategies. For each strategy, we run in 10000 rounds and the probability to create a new block in a round is 0.01, i.e., if the

players follow 0-distance-greedy strategy, on average, for every 100 rounds, a new block is generated. We run the simulation 1000 times and take the average results. Table 1 shows the length of the longest chain, the number of blocks, and the amplification ratio of different strategies. Here ℓ is the length of the longest chain, i.e., the player, who follows ℓ-distance-greedy strategy, extends all chains.

Table 1: Simulation of greedy strategies

Strategies	Longest chain	Empirical amplification ratio	Theoretical amplification ratio
0-distance-greedy	100.6	1	1
2-distance-greedy	162.1	1.62	1.39 (lower bound)
ℓ-distance-greedy	257.5	2.57	2.72 (upper bound)

C From Core-chain to Blockchain

In this section, we start to extend the core-chain protocol $\Pi^{\text{core}\circ}$ in Section 4 to a blockchain protocol that realizes a ledger. Here, the payloads (lists of transactions) will be included in the blocks, We want to emphasize that the payload cannot be included into the core block directly. If the payload is in the core block, the malicious players may try to brute-force different payloads to obtain the solution that satisfies the hash inequality. Furthermore, the scheme must guarantee that a malicious player cannot change the payload he signed before.

Intuitively, the core-chain can be viewed as a (biased) random beacon to select PoS-players to generate new blocks with payloads. The blocks with payloads will also be linked together as a hash chain which is called *main-chain* More concretely, consider a best core-chain $\mathcal{C} = B_0 \| B_1 \| \cdots \| B_\ell$ with the corresponding main-chain $\tilde{\mathcal{C}} = \tilde{B}_0 \| \tilde{B}_1, \cdots \| \tilde{B}_\ell$ [6]. Once a new block-core $B_{\ell+1}$ is generated by a PoS-player, then the same PoS-player is selected to generate the new block $\tilde{B}_{\ell+1}$, in the following format $\tilde{B}_{\ell+1} = \langle \tilde{h}_\ell, B_{\ell+1}, X_{\ell+1}, \tilde{\text{PK}}, \tilde{\sigma} \rangle$ where $\tilde{\sigma} \leftarrow \mathsf{Sign}_{\tilde{\text{SK}}}(\tilde{h}_\ell, B_{\ell+1}, X_i)$, $\tilde{h}_\ell := \mathsf{hash}(\tilde{B}_\ell)$, and X_{i+1} is payload. Here we note that in our blockchain protocol design, the PoS-player holds two combined pairs of keys, (SK, PK) of the strengthened unique signature scheme $(\mathsf{uKeyGen}, \mathsf{uSign}, \mathsf{uVerify})$, and $(\tilde{\text{SK}}, \tilde{\text{PK}})$ of a regular [7] digital signature scheme $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$. Now the blocks in the main blockchain are "glued" with the block-cores in the blockchain, and we can reduce the security of the blockchain protocol to the security of the blockchain protocol.

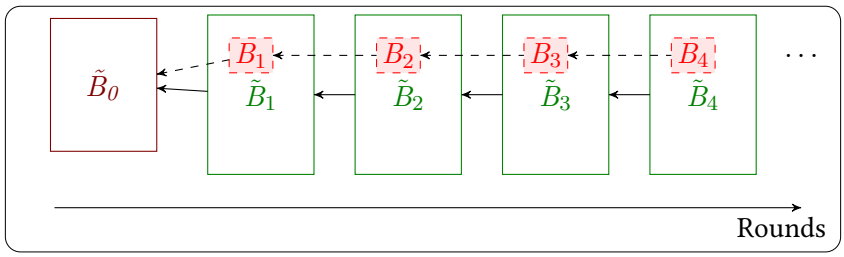

Figure 8: Blockchain structure

Blockchain $\tilde{\mathcal{C}}$ consists of initial setup information (i.e., genesis block) $\tilde{B}_0$, 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 $\tilde{B}_0$ and the ordered sequence of block-cores $B_1, B_2, B_3, \ldots$.

[6] The genesis core-block B_0 and the genesis block $\tilde{B}_0$ are the same

[7] We note that, to achieve adaptive security, this regular signature scheme will be replaced by a forward-secure digital signature scheme [8].

C.1 Ledger and transactions

A ledger consists of multiple accounts in which each account has a number of stakes. To enable the ledger and transactions, we introduction the following functions.

- GenAccount: It returns an account with a public identifier id and a corresponding secret information id_s.
- ExtractLedger: It takes input and a main-chain $\tilde{\mathcal{C}}$ and returns a ledger L that is extracted from $\tilde{\mathcal{C}}$.
- ReadLedger: It takes input as a ledger L, a public identifier id, and returns the current number of stakes that the account with the identifier id has in the ledger L.
- IssueTran: It takes input as a ledger L, a public identifier id and a corresponding secret information id_s. The function IssueTran returns a transaction tx that transfer some stake from the account with the public identifier id to other accounts.
- VerifyTran: It takes input as a ledger L, a transaction tx, and returns 1 if the transaction tx is valid. Otherwise, it returns 0

C.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 4.

Algorithm 5: Protocol Π^{main}

State : At round $\mathtt{r}$, the PoS-player $P \in \mathcal{P}$, with key pairs $(\mathrm{SK}, \mathrm{PK})$, $(\tilde{\mathrm{SK}}, \tilde{\mathrm{PK}})$ and local state $state$, proceeds as follows.

Let $\tilde{\mathbb{C}}$ be the set of local chains in $state$
Let X be the set of transactions in $state$
Compute $\tilde{\mathbb{C}}_{\mathrm{best}} \leftarrow \mathtt{D\text{-}BestMain}^{\bullet}(\tilde{\mathbb{C}}, \mathtt{r})$
for $\tilde{\mathcal{C}} \in \mathbb{C}_{\mathrm{best}}$ **do**
  $\ell := \mathsf{len}(\tilde{\mathcal{C}})$
  Parse $\tilde{\mathcal{C}}$ into $\tilde{B}_0 \| \tilde{B}_1 \| \cdots \| \tilde{B}_\ell$
  for i *from* 1 *to* ℓ **do**
    Parse $\tilde{B}_i$ into $\langle \langle \tilde{h}_i, B_i, X_i \rangle, \tilde{\mathrm{PK}}_i, \tilde{\sigma}_i \rangle$
  Obtain the core-chain $\mathcal{C} := B_0 \| B_1 \| \cdots \| B_\ell$
  $prev \leftarrow h(B_\ell)$
  $\sigma := \mathsf{uSign}(\mathrm{SK}, \langle prev, \mathtt{r} \rangle)$
  if $\mathsf{H}(prev, \mathtt{r}, PK, \sigma) < \mathtt{T}$ **then**
    Create new block $B := \langle prev, \mathtt{r}, \mathrm{PK}, \sigma \rangle$
    $\mathsf{L}_{\tilde{\mathcal{C}}} := \mathsf{ExtractLedger}(\tilde{\mathcal{C}})$
    Set the payload $X_{\tilde{\mathcal{C}}} := \emptyset$
    for $\mathrm{tx} \in X$ **do**
      if $\mathsf{VerifyTran}(\mathsf{L}_{\tilde{\mathcal{C}}}, \mathrm{tx}) = 1$ **then**
        Add tx to $X_{\tilde{\mathcal{C}}}$
    $\tilde{h} = \mathsf{hash}(\tilde{B}_\ell)$
    $\tilde{\sigma} \leftarrow \mathsf{Sign}(\tilde{\mathrm{SK}}, \langle \tilde{h}, B, X_{\tilde{\mathcal{C}}} \rangle)$
    $\tilde{B} := \langle \langle \tilde{h}, B, X_{\tilde{\mathcal{C}}} \rangle, \tilde{\mathrm{PK}}, \tilde{\sigma} \rangle$
    $\tilde{\mathcal{C}}_1 = \tilde{\mathcal{C}} \| \tilde{B}$
    Broadcast $\tilde{\mathcal{C}}_1$

Initialization. Similar to the core-chain protocol, the public keys of the unique digital signature scheme are stored in the genesis block. However, since another regular digital signature scheme is used, the public keys of this digital signature scheme are also stored in the genesis block. To be precise, given an (initial) group

of PoS-players $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$, a security parameter κ, a unique digital signature scheme (uKeyGen, uKeyVer, uSign, uVerify), and a regular digital signature scheme (KeyGen, Sign, Verify), the initialization is as follows: each PoS-player $P_j \in \mathcal{P}$, with an identifier id_j, generates two key pairs $(\mathrm{sk}_j, \mathrm{pk}_j) \leftarrow \mathsf{uKeyGen}(1^\kappa)$, $(\tilde{\mathrm{sk}}_j, \tilde{\mathrm{pk}}_j) \leftarrow \mathsf{KeyGen}(1^\kappa)$ and keeps sk_j, $\tilde{\mathrm{sk}}_j$ secret. The public keys are stored in the genesis block $\tilde{B}_0$, alongside with a randomness rand and an initial ledger $\mathsf{L}_{\mathsf{init}}$, i.e., $\tilde{B}_0 = \langle((\mathrm{pk}_1, \tilde{\mathrm{pk}}_1), (\mathrm{pk}_2, \tilde{\mathrm{pk}}_2), \cdots, (\mathrm{pk}_n, \tilde{\mathrm{pk}}_n)), \mathsf{rand}, \mathsf{L}_{\mathsf{init}}\rangle$. We remark that, here, the players are independent with the ledger.

Blockchain extension. Similar to the core-chain protocol, for each PoS-player P, once activated by the environment at a round, the party P finds the best valid blockchain $\tilde{\mathcal{C}}_{\mathrm{best}}$ by running the procedure D-BestMain$^\bullet$, and then updates its local blockchain $\tilde{\mathcal{C}} := \tilde{\mathcal{C}}_{\mathrm{best}}$ (see Algorithm 5 for the pseudocode). Note that, the i-th block in blockchain $\tilde{\mathcal{C}}$, is in the following format $\tilde{B}_i := \langle\langle \tilde{h}_{i-1}, B_i, X_i\rangle, P_i, \tilde{\sigma}_i\rangle$. That means, from $\tilde{B}_i$, we can obtain the i-th block-core B_i. We thus can derive the core-chain $\mathcal{C}$ from the blockchain $\tilde{\mathcal{C}}$. If the PoS-player P is selected, i.e., she/he generates a signature σ for $context := \langle h_\ell, \mathtt{r}_{\ell+1}\rangle$, and pass the hash inequality. Then she/he defines a new block-core $B_{\ell+1} := \langle\langle h_\ell, \mathtt{r}_\ell\rangle, P, \sigma\rangle$, updates his local core-chain $\mathcal{C}$. Once the new block-core $B_{\ell+1}$ is generated, the PoS-player P select a payload $X_{\ell+1}$ that consists of all valid transactions on the chain $\tilde{\mathcal{C}}$ in her/his local state. generates a signature $\tilde{\sigma}$ for $\langle \tilde{h}_\ell, B_{\ell+1}, X_{\ell+1}\rangle$. Then he can define a new block $\tilde{B}_{\ell+1} := \langle\langle \tilde{h}_\ell, B_{\ell+1}, X_{\ell+1}\rangle, P, \tilde{\sigma}\rangle$, and update his local blockchain $\tilde{\mathcal{C}}$. He then broadcasts the local blockchain to the network. Please refer to Algorithm 5 for more details of our blockchain protocol.

Algorithm 6: PROCEDURE D-BestMain$^\bullet$

Input : A chain set $\tilde{\mathbb{C}}$ at round $\mathtt{r}$.
Output: The best chain set $\tilde{\mathbb{C}}_{\mathrm{best}}$.

for $\tilde{\mathcal{C}} \in \tilde{\mathbb{C}}$ **do**
  Parse $\tilde{\mathcal{C}}$ into $\tilde{B}_0 \| \tilde{B}_1 \| \cdots \| \tilde{B}_\ell$
  for i *from* 1 *to* ℓ **do**
    Parse $\tilde{B}_i$ into $\langle\langle \tilde{h}_i, B_i, X_i\rangle, \tilde{\mathrm{pk}}_i, \tilde{\sigma}_i\rangle$
    if $\mathsf{hash}(\tilde{B}_{i-1}) \neq \tilde{h}_i$ *or* $\mathsf{Verify}(\tilde{pk}_i, \langle \tilde{h}, B_i, X_i\rangle, \tilde{\sigma}_i) = 0$ **then**
      Remove $\tilde{\mathcal{C}}$ from $\tilde{\mathbb{C}}$
    $\mathsf{L}_{i-1} := \mathsf{ExtractLedger}(\tilde{\mathcal{C}}[0, i-1])$
    for $\mathrm{tx} \in X_i$ **do**
      if $\mathsf{VerifyTran}(\mathsf{L}_{i-1}, \mathrm{tx}) = 0$ **then**
        Remove $\tilde{\mathcal{C}}$ from $\tilde{\mathbb{C}}$
    Parse B_i into $\langle prev_i, \mathtt{r}_i, \mathrm{pk}_i, \sigma_i\rangle$
    if $h(B_{i-1}) \neq prev_i$ *or* $\mathsf{H}(prev_i, \mathtt{r}_i, pk_i, \sigma_i) \geq \mathtt{T}$ *or* $\mathsf{uVerify}(pk_i, \langle prev_i, \mathtt{r}_i\rangle, \sigma_i) = 0$ *or* $\mathtt{r}_i > \mathtt{r}$ **then**
      Remove $\tilde{\mathcal{C}}$ from $\tilde{\mathbb{C}}$
Set $\tilde{\mathcal{C}}_{\mathrm{best}}$ be the longest core-chain in $\tilde{\mathbb{C}}$
for $\tilde{\mathcal{C}} \in \tilde{\mathbb{C}}$ **do**
  if $\mathsf{distance}(\tilde{\mathcal{C}}_{\mathrm{best}} \to \tilde{\mathcal{C}}) \leq \mathtt{D}$ **then**
    $\tilde{\mathbb{C}}_{\mathrm{best}} := \tilde{\mathbb{C}}_{\mathrm{best}} \cup \{\tilde{\mathcal{C}}\}$
Return $\tilde{\mathbb{C}}_{\mathrm{best}}$

The best set of main-chains procedure. The procedure D-BestMain$^\bullet$ will output *a set of best main-chains* including the longest chain, and several chains that are very close to, and slightly (i.e., D blocks) shorter than the longest chain (see Algorithm 6 for the pseudocode). First, the procedure D-BestMain$^\bullet$ iterates through the set of main-chains $\tilde{\mathbb{C}}$ to identify the valid main-chains. To be precise, for each main-chain $\tilde{\mathcal{C}} \in \tilde{\mathbb{C}}$, the procedure D-BestMain$^\bullet$ evaluates every blocks on the main-chain $\tilde{\mathcal{C}} = \tilde{B}_0 \| \tilde{B}_1 \| \cdots \| \tilde{B}_\ell$. For each main-block $\tilde{B}_i$, the procedure D-BestMain$^\bullet$ (1) verifies the signature on the payload is correct, (2) verifies the transactions in the payload is valid and (3) verifies the corresponding core-block B_i is correct. Then, it sets the longest valid main-chain as the best main-chain. Finally, the procedure D-BestMain$^\bullet$ iterates through the set of main-chains in the local state

of the player to find all the chains in which the distance from the best chain to those chains does not exceed D.

C.3 Analysis of blockchain protocol

As mentioned before, our blockchain protocol Π^{main} can be viewed as an augmented version of the core-chain protocol $\Pi^{\text{core}\circ}$ in Section 4; each security property of our blockchain protocol can be reduced to the corresponding property of the core-chain protocol. We note that, as in the core-chain protocol $\Pi^{\text{core}\circ}$, the security properties hold under the assumption of honest majority of effective stakes based on α° and β°.

Theorem C.1. *Consider blockchain protocol* Π^{main} *where honest players follow the* 2*-distance-greedy strategy. Assume that* $(\mathsf{uKeyGen}, \mathsf{uKeyVer}, \mathsf{uSign}, \mathsf{uVerify})$ *is a secure unique digital signature scheme, and* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *is secure digital signature scheme. If* $\alpha^\circ = \lambda\beta^\circ$, $\lambda > 1$, *then the protocol* Π^{main} *can achieve chain growth, chain quality, and common prefix properties.*

From the unforgeability property of the digital signature scheme, we can show that the protocol Π^{main} can achieve the same security as the protocol $\Pi^{\text{core}\circ}$. Intuitively, by using the digital signature scheme, we can ensure that (1) the honest block producers (the players can can generate new core-blocks) can always generate a valid corresponding blocks and (2) the blocks that are not generated by the block producers of the corresponding core-blocks are not valid. To be precise, the correctness of signature generation property guarantees that if an honest player can generate a core-block, she/he can generate a valid block. Furthermore, the unforgeability of signature generation guarantees that the malicious players cannot generate a block on a core-block that is generated by an honest players. Note that, from a core-block, the adversary can generate multiple blocks at *the same block height* on the main-chain. However, we will show that the adversary cannot take this advantage to break the security of the protocol Π^{main}.

First, we introduce the notion of the *corresponding core-chain* of a main chain. Intuitively, the corresponding core-chain of a main chain is the core-chain in which each block on the main-chain is generated from a core-block on the core-chain. In detail, consider a main-chain $\tilde{\mathcal{C}} = \tilde{B}_0 \| \tilde{B}_1 \| \cdots \| \tilde{B}_\ell$. For all $i \in [\ell]$, we parse $\tilde{B}_i$ into $\langle \langle \tilde{h}_i, B_i, X_i \rangle, \tilde{\text{PK}}_i, \tilde{\sigma}_i \rangle$. Then the corresponding core-chain of $\tilde{\mathcal{C}}$ is $\mathcal{C} = B_0 \| B_1 \| \cdots \| B_\ell$ (where $B_0 = \tilde{B}_0$).

Note that, the adversary could generate multiple blocks from the same core-block, and thus generate and extend multiple main-chains with respect to the same corresponding core-chain. However, an honest player will extend exactly *one* main chain; if there are multiple main-chains, the honest player will always choose the best one to extend. At any moment, if there is an honest player who generates a core-block, the player will extend the core-chain, and at the same time, generates a single main-chain in the system. We have the following lemmas.

Lemma C.2. *Consider core-chain protocol* Π^{main} *in the presence of an arbitrary adversary. Assume that* $(\mathsf{uKeyGen}, \mathsf{uKeyVer}, \mathsf{uSign}, \mathsf{uVerify})$ *is a secure unique signature scheme and* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *is a secure signature scheme. Consider a main-chain* $\tilde{\mathcal{C}}$. *Let* n *be the number of players and* ρ *be the fraction of malicious players in the protocol execution. At a given round* r, *the probability that an honest player generates a new block on the main-chain* $\tilde{\mathcal{C}}$ *is* $\alpha_0 = 1 - (1-p)^{n(1-\rho)}$.

Proof sketch. Let $\mathcal{C}$ be the corresponding core-chain of the main-chain $\tilde{\mathcal{C}}$. Consider an honest player P_j that hold the key pairs $(\text{SK}_j, \text{PK}_j)$ and $(\tilde{\text{SK}}_j, \tilde{\text{PK}}_j)$. Recall in protocol Π^{main} that the player P_j can generate a new block at round r on the main chain $\tilde{\mathcal{C}}$ if

1. P_j can *generate* a valid core-block $B := \langle prev, r, \text{PK}_j, \sigma \rangle$ where $prev$ is the hash value of the last block on $\mathcal{C}$, and σ is a signature on the context $\langle prev, r \rangle$.
2. Then, P_j can *generate* a "qualified" signature $\tilde{\sigma}$ on $\langle \tilde{h}, B, X \rangle$, i.e., $\mathsf{Verify}(\tilde{\text{PK}}_j, \langle \tilde{h}, B, X \rangle, \tilde{\sigma}_i) = 1$, where $\tilde{h}$ is the hash value of the last block on the main-chain $\tilde{\mathcal{C}}$ and X is the payload.

From Lemma A.1, the probability that P_j can generate a new core-block on the core-chain $\mathcal{C}$ is p. Plus, the correctness of the signature scheme ensure that P_j can always generate a qualified signature. Thus, the probability that the player P_j generates a new block on the main-chain $\tilde{\mathcal{C}}$. Here, the number of honest players is $n(1-\rho)$. Thus, the probability that an honest player generates a new block on the main-chain $\tilde{\mathcal{C}}$ is $\alpha_0 = 1-(1-p)^{n(1-\rho)}$.

□

Lemma C.3. *Consider core-chain protocol* Π^{main} *in the presence of an arbitrary adversary. Assume that* $(\mathsf{uKeyGen}, \mathsf{uKeyVer}, \mathsf{uSign}, \mathsf{uVerify})$ *is a secure unique signature scheme and* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *is a secure signature scheme. Consider a set of main-chains* $\tilde{\mathbb{C}} = \{\tilde{\mathcal{C}}_1, \tilde{\mathcal{C}}_2, \cdots\}$ *in which all main-chains* $\tilde{\mathcal{C}}_i \in \tilde{\mathbb{C}}$ *have the same corresponding core-chain. Let* n *be the number of players and* ρ *be the fraction of malicious players in the protocol execution. At a given round* r*, the probability that a malicious player generates a new block on any main-chain* $\tilde{\mathcal{C}}_i \in \tilde{\mathbb{C}}$ *is* $\beta = 1-(1-p)^{n\rho}$.

Proof sketch. Let $\mathcal{C}$ be the corresponding core-chain of all the main chains in $\tilde{\mathbb{C}}$. Recall in protocol Π^{main} that a malicious player $P_j \in \mathcal{P}$ can generate a new block at round r on a main chain $\tilde{\mathcal{C}}_i \in \tilde{\mathbb{C}}$ if

1. P_j can *provide* a valid core-block $B := \langle prev, r, \text{PK}, \sigma\rangle$ where $prev$ is the hash value of the last block on $\mathcal{C}$, PK is a public key of the unique digital signature scheme, and σ is a signature on the context $\langle prev, r\rangle$. Note that, contrary to the first condition in Lemma C.2, here, the player P_j can use the core-block that is generated by another player.

2. Similar to the second condition in Lemma C.2, P_j can *generate* a "qualified" signature $\tilde{\sigma}$ on $\langle \tilde{h}, B, X\rangle$, i.e., $\mathsf{Verify}(\tilde{\text{PK}}, \langle \tilde{h}, B, X\rangle, \tilde{\sigma}_i) = 1$, where $\tilde{h}$ is the hash value of the last block on the main-chain $\tilde{\mathcal{C}}_i$ and X is the payload. Here, the pair of public keys $(\text{PK}, \tilde{\text{PK}})$ is registered in the genesis block $\tilde{B}_0$.

As we mentioned before, the player P_j can use the core-block that is generated by another player and try to generate her/his own block on that core-block. Fortunately, in this case, they player P_j will not able to generate a "qualified" signature. Indeed, if the player P_j use the core-block $B := \langle prev, r, \text{PK}, \sigma\rangle$ that is generated by another player, she/he will not know the corresponding secret keys of $(\text{PK}, \tilde{\text{PK}})$. The unforgeability of the signature scheme ensures that the player P_j cannot forge any "qualified" signatures without knowing the secret key.

Therefore, the player P_j can generate a new block at round r on a main chain $\tilde{\mathcal{C}}_i \in \tilde{\mathbb{C}}$ if and only if P_j can generate a new core-block on the corresponding core-chain $\tilde{\mathbb{C}}$. From Lemma A.1, this event happens with probability p. Here, the number of malicious players is $n\rho$. Thus, the probability that a malicious player generates a new block on any main-chain $\tilde{\mathcal{C}}_i \in \tilde{\mathbb{C}}$ is $\beta = 1-(1-p)^{n\rho}$.

□

Now, similar to the analysis for protocol $\Pi^{\text{core}\circ}$ in Appendix B, we can prove that protocol Π^{main} can achieve chain growth, common prefix and chain quality properties.

D Defending against Adaptive Registration

In previous sections (Sections C), we assume that all players have their stakes registered, before they join the protocol execution. Let's provide a brief explanation below. Note that the process of extending the chains is based on the hash inequality $\mathsf{H}(context, solution) < \mathsf{T}$, where $solution$ is in the form of (PK, σ). Since the adversary now knows the $context$, he can play a "rejection re-sampling" strategy to generate their keys adaptively. More concretely, the adversary first runs key generation algorithm to obtain a key-pair (PK, SK), and then check if the corresponding (PK, σ) is a valid solution to the hash inequality; if not, the adversary will repeat the process. By adopting this strategy, malicious players can significantly increase the probability that they are chosen to extend the chains. In this section, we propose new ideas to address the rejection re-sampling attack; as a result, we

can allow the players to have their stakes registered *during the protocol execution.* We note that, to maintain the security, at any round, we require majority of honest stakes (i.e., at least 66.2% of total stake is honest).

D.1 The modified blockchain protocol $\Pi^{\text{main}\bullet}$

To securely enable adaptive registration, the protocol has been modified as follows.

Initialization. Contrary to the protocol Π^{main}, the players in protocol $\Pi^{\text{main}\bullet}$ are bound with the account in the ledger. At the beginning, the public keys and the identifiers of an (initial) group of PoS-players $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ are stored in the genesis block B_0, alongside with a randomness rand and an initial ledger $\mathsf{L}_{\mathsf{init}}$, i.e., $\tilde{B}_0 = \langle((\text{PK}_1, \tilde{\text{PK}}_1, id_1), (\text{PK}_2, \tilde{\text{PK}}_2, id_2), \cdots, (\text{PK}_n, \tilde{\text{PK}}_n, id_n)), \mathsf{rand}, \mathsf{L}_{\mathsf{init}}\rangle$. Note that, the players needs to have at least st number of stakes in their account to participate in the mining, i.e., $\mathsf{ReadLedger}(\mathsf{L}_{\mathsf{init}}, id_i) \geq \mathsf{st}$, for all $i \in [n]$.

Blockchain extension. Here, the players extend the chain in the same way as in protocol Π^{main}. However, new player is allowed to join the protocol during the protocol execution.

Algorithm 7: PROTOCOL $\Pi^{\text{main}\bullet}$

State : At round $\mathtt{r}$, the PoS-player $P \in \mathcal{P}$, with key pairs (SK, PK), $(\tilde{\text{SK}}, \tilde{\text{PK}})$ and local state $state$, proceeds as follows.

Registration *(for a new player):*
- Player P generates two combined pairs pair $(\text{SK}, \text{PK}) \leftarrow \mathsf{uKeyGen}(1^\kappa)$ and $(\tilde{\text{SK}}, \tilde{\text{PK}}) \leftarrow \mathsf{KeyGen}(1^\kappa)$
- P broadcasts a registration transaction $\langle \mathsf{Registration}, \text{PK}, \tilde{\text{PK}}\rangle$

Deregistration *(for a player who no longer wish to participate in the protocol):*
- P broadcasts a registration transaction $\langle \mathsf{De\text{-}registration}, \text{PK}, \tilde{\text{PK}}\rangle$

Extending the set of best chains:
- Let $\tilde{\mathbb{C}}$ be the set of local chains in $state$
- Let X be the set of transactions in $state$
- Compute $\tilde{\mathbb{C}}_{\text{best}} \leftarrow \mathtt{D}\text{-}\mathsf{BestMain}^\bullet(\tilde{\mathbb{C}}, \mathtt{r})$
- **for** $\tilde{\mathcal{C}} \in \mathbb{C}_{\text{best}}$ **do**
  - $\ell := \mathsf{len}(\tilde{\mathcal{C}})$
  - Parse $\tilde{\mathcal{C}}$ into $\tilde{B}_0 \| \tilde{B}_1 \| \cdots \| \tilde{B}_\ell$
  - **for** i *from* 1 *to* ℓ **do**
    - Parse $\tilde{B}_i$ into $\langle\langle \tilde{h}_i, B_i, X_i\rangle, \tilde{\text{PK}}_i, \tilde{\sigma}_i\rangle$
  - Obtain the core-chain $\mathcal{C} := B_0 \| B_1 \| \cdots \| B_\ell$
  - $prev \leftarrow h(B_\ell)$
  - $\sigma := \mathsf{uSign}(\text{SK}, \langle prev, \mathtt{r}\rangle)$
  - **if** $\mathsf{H}(prev, \mathtt{r}, \text{PK}, \sigma) < \mathtt{T}$ **then**
    - Create new block $B := \langle prev, \mathtt{r}, \text{PK}, \sigma\rangle$
    - $\mathsf{L}_{\tilde{\mathcal{C}}} := \mathsf{ExtractLedger}(\tilde{\mathcal{C}})$
    - Set the payload $X_{\tilde{\mathcal{C}}} := \emptyset$
    - **for** $\text{tx} \in X$ **do**
      - **if** $\mathsf{VerifyTran}(\mathsf{L}_{\tilde{\mathcal{C}}}, \text{tx}) = 1$ **then**
        - Add tx to $X_{\tilde{\mathcal{C}}}$
    - $\tilde{h} = \mathsf{hash}(\tilde{B}_\ell)$
    - $\tilde{\sigma} \leftarrow \mathsf{Sign}(\tilde{\text{SK}}, \langle \tilde{h}, B, X_{\tilde{\mathcal{C}}}\rangle)$
    - $\tilde{B} := \langle\langle \tilde{h}, B, X_{\tilde{\mathcal{C}}}\rangle, \tilde{\text{PK}}, \tilde{\sigma}\rangle$
    - $\tilde{\mathcal{C}}_1 = \tilde{\mathcal{C}} \| \tilde{B}$
    - Broadcast $\tilde{\mathcal{C}}_1$

Adaptive registration. The players now are allowed to register during the protocol execution. To join the protocol,

player P, with an identifier id generates two combined pairs pair $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{uKeyGen}(1^\kappa)$ and $(\tilde{\mathsf{sk}}, \tilde{\mathsf{pk}}) \leftarrow \mathsf{KeyGen}(1^\kappa)$. P keeps sk, $\tilde{\mathsf{sk}}$ secret, and broadcasts a registration transaction $\langle \mathsf{Registration}, \mathsf{pk}, \tilde{\mathsf{pk}}, id \rangle$. Upon the registration transaction is included in a payload of a block, the player is considered as registered. Note that, to join the protocol, the player P needs to have more than st number of stakes. We remark that, after registration, the players need to reserve st stakes. The reserved stakes cannot be used for other purposes, e.g., generating transactions. After a new player register on the blockchain, she/he is allowed to extend the chain after η blocks. We refer to the players that are allowed to extend the chain as *qualified players*. We note that, the players who register before the protocol start, i.e., in the genesis block, are qualified to extend the chain with out waiting for η blocks.

Deregistration. A qualified player can deregister if she/he no longer wish to participate in the protocol. To deregister, the player will broadcast a deregistration transaction $\langle \mathsf{De\text{-}registration}, \mathsf{pk}, \tilde{\mathsf{pk}}, id \rangle$. Upon the deregistration transaction is included in a payload of a block, the player is considered as deregistered. The player now can spend the stakes in her/his account. [8]

Registration transaction verification. A registration transaction $\langle \mathsf{Registration}, \mathsf{pk}, \tilde{\mathsf{pk}}, id \rangle$ is consider to be valid on a chain $\mathcal{C}$ if the player with identifier id has more than st stakes. In detail, let $\mathsf{L} := \mathsf{ExtractLedger}(\mathcal{C})$ be the current ledger. We say the registration transaction $\langle \mathsf{Registration}, \mathsf{pk}, \tilde{\mathsf{pk}}, id \rangle$ is valid if $\mathsf{ReadLedger}(\mathsf{L}, id) \geq \mathsf{st}$.

New best chain strategy Instead of selecting the longest chain as the best chain, we compare the chains based on the creation time of the first few blocks after the last common block. That is, for two divergent chains, $\mathcal{C}$ and $\mathcal{C}'$, if $\max(\mathsf{distance}(\mathcal{C} \rightarrow \mathcal{C}'), \mathsf{distance}(\mathcal{C}' \rightarrow \mathcal{C})) > \eta$, then whichever chain that generates the η blocks first is the better one. (Please see Algorithm 8 for the pseudocode.)

In detail, the procedure $(\mathtt{D}, \eta)$-$\mathsf{BestMain}^\bullet$ iterates through the set of main-chains $\tilde{\mathbb{C}}$ to identify the valid main-chains. Note that, only qualified players are allowed to extend the chains, procedure $(\mathtt{D}, \eta)$-$\mathsf{BestMain}^\bullet$ needs to verify the block is generated by a qualified player. To be precise, for each main-chain $\tilde{\mathcal{C}} \in \tilde{\mathbb{C}}$, the procedure $(\mathtt{D}, \eta)$-$\mathsf{BestMain}^\bullet$ evaluates every block of the main-chain $\tilde{\mathcal{C}} = \tilde{B}_0 \| \tilde{B}_1 \| \cdots \| \tilde{B}_\ell$. For each block $\tilde{B}_i$, the procedure $(\mathtt{D}, \eta)$-$\mathsf{BestMain}^\bullet$ (1) verifies the signature on the payload is correct, (2) verifies the transactions in the payload is valid, (3) verifies the corresponding core-block B_i is correct, and (4) verifies the key pair in the block belongs to a qualified player, i.e., the key pair is in the genesis block or it is register η blocks ago. Here, to verify if a player is qualified, the procedure $(\mathtt{D}, \eta)$-$\mathsf{BestMain}^\bullet$ first checks if there exists an deregistration transaction of the player included on the main-chain. If there exists such transaction, the player is not qualified. Otherwise, the procedure $(\mathtt{D}, \eta)$-$\mathsf{BestMain}^\bullet$ finds the block in which the registration transaction of the player is included. If the block is genesis block or it was added on the main chain η blocks ago, the player is considered as a qualified player. To select the best main-chain, procedure $(\mathtt{D}, \eta)$-$\mathsf{BestMain}^\bullet$ first sets the best main-chain $\tilde{\mathcal{C}}_{\mathrm{best}}$ as the longest chain. Then, it iterates through the set of main-chains, for each main-chain $\tilde{\mathcal{C}} \in \tilde{\mathbb{C}}$, if $\mathsf{distance}(\tilde{\mathcal{C}}_{\mathrm{best}} \rightarrow \tilde{\mathcal{C}}) \leq \mathtt{D}$ and $\tilde{\mathcal{C}}$ generates the first η blocks after the last common block faster than $\tilde{\mathcal{C}}_{\mathrm{best}}$, it sets $\tilde{\mathcal{C}}_{\mathrm{best}} = \tilde{\mathcal{C}}$. Finally, the procedure $(\mathtt{D}, \eta)$-$\mathsf{BestMain}^\bullet$ iterates through the set of main-chains in the local state of the player to find all the chains in which the distance from the best chain to those chains does not exceed $\mathtt{D}$.

Now, malicious players cannot register a biased key pair for the following η blocks to increase the probability he will be elected. However, there is still an issue that the malicious players may register a biased key pair for a round, η blocks later. We will show this issue can be resolved if we further improve the best chain strategy. The intuition is that if the malicious players prepare a biased key pair for a public chain, then the honest player will win some blocks among the η blocks with high probability. The malicious players cannot predict the signature of honest players, so he cannot predict the input of the blocks η blocks after. This means that the malicious players cannot get advantage for η blocks after if the chain is public. If the malicious players decide to extend

[8] The adversary may corrupt the players who left the protocol can control the majority of the stake at a moment in the past. In this case, the adversary can perform a long-range attack can generating a chain from the past. As the adversary controls the majority of the stake at the moment, it can generate a better chain than the current public best chain. Here, we assume the players erase their local state when they leave the protocol.

Algorithm 8: PROCEDURE $(\mathtt{D}, \eta)$-BestMain$^{\bullet}$

Input : A chain set $\tilde{\mathbb{C}}$, round $\mathtt{r}$.
Output: The best chain set $\tilde{\mathbb{C}}_{\mathrm{best}}$.

for $\tilde{\mathcal{C}} \in \tilde{\mathbb{C}}$ **do**
  Parse $\tilde{\mathcal{C}}$ into $\tilde{B}_0 \| \tilde{B}_1 \| \cdots \| \tilde{B}_\ell$;
  for i *from* 1 *to* ℓ **do**
    Parse $\tilde{\mathcal{C}}$ into $\tilde{B}_0 \| \tilde{B}_1 \| \cdots \| \tilde{B}_\ell$
    for i *from* 1 *to* ℓ **do**
      Parse $\tilde{B}_i$ into $\langle \langle \tilde{h}_i, B_i, X_i \rangle, \tilde{\mathrm{PK}}_i, \tilde{\sigma}_i \rangle$
      if $\mathsf{hash}(\tilde{B}_{i-1}) \neq \tilde{h}_i$ *or* $\mathsf{Verify}(\tilde{PK}_i, \langle \tilde{h}, B_i, X_i \rangle, \tilde{\sigma}_i) = 0$ **then**
        Remove $\tilde{\mathcal{C}}$ from $\tilde{\mathbb{C}}$
      $\mathsf{L}_{i-1} := \mathsf{ExtractLedger}(\tilde{\mathcal{C}}[0, i-1])$
      for $\mathrm{tx} \in X_i$ **do**
        if $\mathsf{VerifyTran}(\mathsf{L}_{i-1}, \mathrm{tx}) = 0$ **then**
          Remove $\tilde{\mathcal{C}}$ from $\tilde{\mathbb{C}}$
      Parse B_i into $\langle prev_i, \mathtt{r}_i, \mathrm{PK}_i, \sigma_i \rangle$
      if $h(B_{i-1}) \neq prev_i$ *or* $\mathsf{H}(prev_i, \mathtt{r}_i, PK_i, \sigma_i) \geq \mathtt{T}$ *or* $\mathsf{uVerify}(PK_i, \langle prev_i, \mathtt{r}_i \rangle, \sigma_i) = 0$ *or* $\mathtt{r}_i > \mathtt{r}$ **then**
        Remove $\tilde{\mathcal{C}}$ from $\tilde{\mathbb{C}}$
    $\mathsf{reg} := -1$
    for j *from* $i-1$ *to* 0 **do**
      if $\tilde{B}_j$ *contains the deregistration transaction* $\langle \mathsf{De\text{-}registration}, PK_i, \tilde{PK}_i, id_i \rangle$ **then**
        Break
      if $\tilde{B}_j$ *contains the registration transaction* $\langle \mathsf{Registration}, PK_i, \tilde{PK}_i, id_i \rangle$ **then**
        if $j > 0$ **then**
          $\mathsf{L}' := \mathsf{ExtractLedger}(\tilde{\mathcal{C}}[0, j-1])$
        else
          $\mathsf{L}' := \mathsf{L}_{\mathsf{init}}$
        if $\mathsf{ReadLedger}(\mathsf{L}', id_i) \geq \mathsf{st}$ **then**
          $\mathsf{reg} := j$
        Break
    if $\mathsf{reg} = -1$ *or (*$\mathsf{reg} \neq 0$ *and* $i - \mathsf{reg} < \eta$*)* **then**
      Remove $\tilde{\mathcal{C}}$ from $\tilde{\mathbb{C}}$
Set $\tilde{\mathcal{C}}_{\mathrm{best}}$ be the longest core-chain in $\tilde{\mathbb{C}}'$;
for $\mathcal{C} \in \mathbb{C}$ **do**
  $d := \mathsf{distance}(\tilde{\mathcal{C}} \to \tilde{\mathcal{C}}_{\mathrm{best}})$;
  if $\mathsf{distance}(\tilde{\mathcal{C}} \to \tilde{\mathcal{C}}_{\mathrm{best}}) > \eta$ **then**
    Obtain i is the index of last common block of $\tilde{\mathcal{C}}$ and $\tilde{\mathcal{C}}_{\mathrm{best}}$;
    Parse $\tilde{\mathcal{C}}[i+\eta]$ into $\langle \langle \tilde{h}, \langle prev, \mathtt{r}, \mathrm{PK}, \sigma \rangle, X \rangle, \tilde{\mathrm{PK}}, \tilde{\sigma} \rangle$;
    Parse $\tilde{\mathcal{C}}_{\mathrm{best}}[i+\eta]$ into $\langle \langle \tilde{h}', \langle prev', \mathtt{r}', \mathrm{PK}', \sigma' \rangle, X' \rangle, \tilde{\mathrm{PK}}', \tilde{\sigma}' \rangle$;
    if $r < r'$ **then**
      $\tilde{\mathcal{C}}_{\mathrm{best}} := \tilde{\mathcal{C}}$;
for $\mathcal{C} \in \mathbb{C}$ **do**
  if $\mathsf{distance}(\tilde{\mathcal{C}}_{\mathrm{best}} \to \tilde{\mathcal{C}}) \leq \mathtt{D}$ **then**
    $\tilde{\mathbb{C}}_{\mathrm{best}} := \tilde{\mathbb{C}}_{\mathrm{best}} \cup \{\tilde{\mathcal{C}}\}$;
Return $\mathbb{C}_{\mathrm{best}}$;

a hidden blockchain, he can prepare a biased player for a block η blocks after. However, he will lose the chain growth competition for the first η blocks. Hence, by using the modified best chain strategy, they adversary cannot register a biased key pair to generate a better chain.

D.2 Security analysis

In this section, we present the security analysis of protocol $\Pi^{\text{main}\bullet}$. We recall that, the security properties of protocol $\Pi^{\text{core}\circ}$ have be proven under the assumption of honest majority of *effective stakes* based on α° and β°. Now, under the same assumption, we will show that our modified core-chain protocol $\Pi^{\text{main}\bullet}$ can also achieve the same properties. Please note that our new adversary is stronger since there is no restriction on how players are registered with respect to the protocol execution. We have the following theorem:

Theorem D.1. *Consider core-chain protocol $\Pi^{\text{main}\bullet}$ where $\eta = \Omega(\kappa + \mathtt{D})$, the honest players follow the 2-greedy strategy while adversarial players could follow any arbitrary strategy. Assume that* $(\mathsf{uKeyGen}, \mathsf{uKeyVer}, \mathsf{uSign}, \mathsf{uVerify})$ *is a secure unique digital signature scheme, and* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *is secure digital signature scheme. If $\alpha^\circ = \lambda\beta^\circ$, $\lambda > 1$, then the protocol $\Pi^{\text{main}\bullet}$ can achieve chain growth, chain quality, and common prefix properties.*

In our modified protocol $\Pi^{\text{main}\bullet}$, malicious players cannot register key pairs so that they can extend the chains immediately. What the malicious players can do, however, is to register biased key pairs now, and then try to extend the chains many rounds later. We will prove that malicious players cannot obtain additional advantage by playing this strategy. First, we will show that the probability that malicious players are able to predict the latest block of the best public chain is negligible.

Lemma D.2. *Let chain $\mathcal{C}$ be the best valid public chain with length ℓ in round r. Suppose the length of best valid public chain $\mathcal{C}'$ will be $\ell + \eta$ in round $r + t$. The probability that the malicious players predict the last block on chain $\mathcal{C}'$ in round r is $e^{-\Omega(\eta)}$ at most in round r.*

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

If malicious players cannot predict the last block of the best chain, then he cannot perform reject-resampling to select a biased key pair so that the corresponding stakeholder can be chosen in a future round with much higher probability. From Lemma D.2, we conclude that a malicious player, by playing adaptive key registration strategy, cannot improve the probability that he is chosen for extending the public chain. Next, we will show that the malicious players cannot gain advantage, by playing this adaptive strategy, for extending a private (hidden) chain.

From the modified protocol, we know that the adaptive key generation will not affect the first η blocks. That means, it is not helpful for the adversary to extend the hidden chain, via the adaptive key registration strategy.

Chain growth Honest players in protocol $\Pi^{\text{main}\bullet}$ will extend the chains in the same way as that in protocol $\Pi^{\text{core}\circ}$. We note that, the probability of the adversary to create a hidden chain with length η, that is longer than the public chain, is negligible. Thus, the adversary cannot create a private best chain , that are divert more than η blocks from the current public best chain. Hence, the best chain (selected from both public and private chains) has bigger or equal length with the public best chain. From Corollary B.3 we have:

Corollary D.3 (Chain growth)**.** *Consider core-chain protocol $\Pi^{\text{main}\bullet}$ that allows new players to register to the system adaptively. Consider an honest PoS-player P' with best local PoS core-chain $\mathcal{C}'$ in round r', and an honest PoS-player P'' with best local core-chain $\mathcal{C}''$ in round r'', where $r'' > r'$. Then we have $\Pr\left[\mathsf{len}(\mathcal{C}'') - \mathsf{len}(\mathcal{C}') \geq g \cdot t\right] \geq 1 - e^{-\Omega(t)}$, where $t = r'' - r'$, $g = (1-\delta)\alpha^\circ$, and $\delta > 0$.*

Common prefix Again, from Lemma D.2, the adversary cannot obtain extra benefit by playing the adaptive strategy. They cannot produce more blocks by adaptively selecting key pairs. From Corollary B.9 we have:

Corollary D.4 (Common prefix). *Consider $\alpha^\circ = \lambda\beta^\circ$, $\lambda > 1$, and $\delta > 0$. Consider core-chain protocol $\Pi^{\text{main}\bullet}$ that allows new players to register to the system adaptively. Consider two honest PoS-players, P in round r and P' in round r', with the local best PoS core-chains $\mathcal{C}, \mathcal{C}'$, respectively, where $r' \geq r$. Then we have $\Pr[\mathcal{C}[\neg\kappa] \preceq \mathcal{C}'] \geq 1 - e^{-\Omega(\kappa)}$.*

Chain quality From Lemma D.2, the adversary cannot obtain additional advantage by playing the adaptive strategy. That is, they cannot produce more blocks by adaptively selecting key-pairs and having their stakes registrated. From Corollary B.10 we have:

Corollary D.5 (Chain quality). *Consider $\alpha^\circ = \lambda\beta^\circ$, $\lambda > 1$, and $\delta > 0$. Consider core-chain protocol $\Pi^{\text{main}\bullet}$ that allows new players to register to the system adaptively. Consider an honest PoS-player with PoS core-chain $\mathcal{C}$. Consider that ℓ consecutive block-cores of $\mathcal{C}$, where ℓ_{good} block-cores are generated by honest PoS-players. Then we have $\Pr\left[\frac{\ell_{good}}{\ell} \geq \mu\right] \geq 1 - e^{-\Omega(\ell)}$, where $\mu = 1 - (1+\delta)\frac{1}{\lambda}$.*

E Extensions

Our design is a natural mimic of Nakamoto's but via proof-of-stake. We can easily "borrow" many ideas in Nakamoto's white paper (and in follow-up papers) to our design.

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

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

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

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

It is easy to argue that the winning probability of a PoS-player for generating a new block-core is proportional to the amount of stake he controls: 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 $1 - (1-p)^v \approx vp$.

F Additional Attacks

Posterior corruption (key-selling) attacks In a posterior corruption attack, an adversary can attempt to corrupt the secret keys of the block producers, i.e., the players that generated blocks, in the history of the system. Now, the adversary can use those secret keys to rewrite the history of the blockchain.

We remark that, in "epoch-by-epoch" proof-of-stake protocols, the adversary can rewrite the history in the last epoch by buying the *majority* secret keys of the players that generated blocks in that epoch. Indeed, since the public randomness is fixed for each epoch, the adversary can use the key of the block producers in the best chain to generate blocks in a different chain. If the adversary only extends on its chain and does not participate in extending the best chain, the chain of the adversary can be better than the best chain.

On the other hand, in our "block-by-block" proof-of-stake protocol, to rewrite the history, the adversary have to buy *all* secret keys of the block producer in the past. As the public randomness of our protocol updates in every block, even when the adversary control the secret keys of the block procedures in the past, it cannot change the structure of the core-chain but only the payloads of those blocks. Furthermore, if a player refuses to sell the secret key, the adversary cannot change the payload of the blocks that are generated before that blocks.

A typical solution to depend against posterior corruption attacks is using key-evolving cryptography [23]. The concept behind key-evolving cryptography and is that the lifetime of the key is divided into period for which a different private key is used, yet the public one remains the same. In this regard, there is a key update algorithm to derive the new private key from the previous one. Therefore, the period that the signature was issued becomes an integral part of the whole signature. As a result, even if a key is leaked it cannot be used to re-sign older messages.

Stake-bleeding attacks Stake-bleeding attacks [35] is a type of long-range attacks that cause by the change of the stake distribution among the players. In detail, adversary does not participate on extending the longest public chain and makes attempts to generate a hidden chain. All transactions on the longest public chain will be included on the hidden chain of the adversary. The number of stakes that are controlled by the adversary slowly increases in the hidden chain. In other words, the growth rate of the hidden chain slowly increases while the growth rate of the best public chain remains the same. Eventually, the hidden chain will be longer than the longest public chain. We remark that, this attack is rather slow. According to [35], if the adversary with 30% stake need 5.5 year to successfully perform stake-bleeding attacks.

Our protocol in Section D can defend against stake-bleeding attacks. Indeed, instead of selecting the longest chain as the best chain, the players choose the best chain by comparing the creation time of the first η blocks after the last common block. The chain that generates the first η blocks faster is considered as the better chain. In the stake-bleeding attack we describe above, the adversary cannot claim enough reward to control majority of stakes within η blocks. Thus, the adversary cannot generate the first η blocks faster than the honest players. Hence, the hidden chain of the adversary cannot be the best chain.

G Additional Preliminaries

Digital signature scheme [39] A digital signature scheme consists of three probabilistic polynomial-time algorithms $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ as follows.

- The key generation KeyGen takes as input a security parameter 1^κ and outputs a pair of signing-verification keys $(\mathrm{SK}, \mathrm{PK})$.
- The signing algorithm Sign takes as input a signing key SK and a message m and output a signature σ. We write this as $\sigma \leftarrow \mathsf{Sign}_{\mathrm{SK}}(m)$.
- The verification algorithm Verify takes a verification key PK, a message m, and a signature σ. It outputs 1 if the signature is correct, and 0 if otherwise. We write this as $\mathsf{Verify}_{\mathrm{PK}}(m, \sigma)$.

Definition G.1. *We say* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *is a digital signature scheme, if it satisfies:*

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

$$\Pr\left[\ (pk, sk) \leftarrow \mathsf{KeyGen}(1^\kappa); \sigma := \mathsf{Sign}_{sk}(x)\ :\ \mathsf{Verify}_{pk}(x, \sigma) = 1\ \right] = 1$$

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

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

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

Unique signature scheme In a unique signature scheme, for every possible verification key, every message to be signed, there is a unique signature. Please see Section 6.5.1 of Goldreich's textbook [38] for details. Here we include a version of the definition for syntax and properties: A unique signature scheme consists of four algorithms, a randomized key generation algorithm uKeyGen, a deterministic key verification algorithm uKeyVer, a deterministic signing algorithm uSign, and a deterministic verification algorithm uVerify; we expect for each verification key there exists only one signing key; we also expect for each pair of message and verification key, there exists only one signature. We have the following definition.

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

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

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

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*

$$\Pr\left[\ (pk, sk_1, sk_2) \leftarrow \mathcal{A}(1^\kappa)\ :\ \mathsf{uKeyVer}(pk, sk_1) = 1 \wedge\ \mathsf{uKeyVer}(pk, sk_2) = 1 \wedge\ sk_1 \neq sk_2\ \right] \leq \mathsf{negl}(\kappa)$$

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

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

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

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

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* $\mathsf{uSign}(sk, \cdot)$*.*

Unique signature schemes and related notions have been investigated in literatures (e.g., [40, 50, 49]). Please see Section 6.5.1 of Goldreich's textbook [38] for detailed discussions about the constructions. Several efficient constructions can be found in literature. For example, the well-known BLS signature [12] can be a good candidate.