Best-Possible Unpredictable Proof-of-Stake: An Impossibility and a Practical Design

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

Jonathan Katz
Google
Washington DC, USA
jkatz2@gmail.com

Zhenghao Lu
Shanghai Jiao Tong University
Shanghai, China
zhenghao.lu.sh@gmail.com

Phuc Thai
Sky Mavis
Ho Chi Minh, Vietnam
phuc.thai@skymavis.com

Hong-Sheng Zhou
Virginia Commonwealth University
Richmond VA, USA
hszhou@vcu.edu

2025 IEEE 10th European Symposium on Security and Privacy (EuroS&P) | 979-8-3315-9493-0/25/$31.00 ©2025 IEEE | DOI: 10.1109/EuroSP63326.2025.00035

***Abstract*—The proof-of-stake (PoS) protocols aim to reduce the unnecessary computing power waste seen in Bitcoin. Various practical and provably secure designs have been proposed, like Ouroboros Praos (Eurocrypt 2018) and Snow White (FC 2019). However, the essential security property of unpredictability in these protocols remains insufficiently explored. This paper delves into this property in the cryptographic setting to achieve the "best possible" unpredictability for PoS protocols.**

We first present an impossibility result for *all* PoS protocols under the *single-extension* design framework, where each honest player extends one chain per round. The state-of-the-art permissionless PoS protocols (e.g., Praos, Snow White, and more), are all under this single-extension framework. Our impossibility result states that, if a single-extension PoS protocol achieves the best possible unpredictability, then this protocol cannot be proven secure unless more than 73% of stake is honest.

To overcome this impossibility, we introduce a new design framework called *multi-extension* PoS, allowing each honest player to extend *multiple* chains using greedy strategy in a round. This strategy allows us to construct a class of PoS protocols that achieve the best possible unpredictability. Additionally, we design a new tiebreak rule for the multi-extension protocol to choose the best chain that can be extended faster, ensuring that the adversary cannot slow-down the chain growth of honest players. It is noteworthy that these protocols can be proven secure, assuming a much smaller fraction (e.g., 57%) of stake to be honest.

For a comprehensive security analysis in the cryptographic setting, we develop several new techniques. Analyzing chain growth becomes highly non-trivial as players can extend multiple chains. We introduce a new analysis framework using the Markov chain to assess the chain growth of a multi-extension protocol. To prove the common prefix property, we introduce a concept called "virtual chains" and present a reduction from the regular version of the common prefix to "common prefix w.r.t. virtual chains."

***Index Terms*—blockchain, consensus, proof-of-stake, unpredictability**

1. Introduction

Cryptocurrencies like Bitcoin [1] have proven to be a phenomenal success. These protocols are executed by a **large-size** peer-to-peer network of nodes using the proof-of-work (PoW) mechanism [2], [3]. They provide a trustworthy, append-only, and always-available public ledger, facilitating the implementation of a global payment system (e.g., Bitcoin) or a global computer (e.g., Ethereum). In Bitcoin, consensus is achieved through a PoW mechanism. Specifically, the participant who discovers *a valid solution* (a random nonce) to *the hash-based PoW puzzle* becomes the block producer for generating the next block.

From proof-of-work to proof-of-stake. However, PoW mechanisms require significant computational resources. A promising alternative involves securing a blockchain using *coins (also known as stake)* rather than computing power. This approach could lead to a more environmentally friendly system, as it doesn't depend on high energy consumption for security. Several Proof-of-Stake (PoS) mechanisms have been proposed and are widely discussed within the cryptocurrency community (e.g., [4], [5], [6], [7]). In PoS-based blockchain protocols, participants must prove ownership of a certain amount of stake (coins) to take part in the consensus process. Compared to PoW, the computational cost of maintaining the blockchain in PoS systems is considerably lower.

From ad hoc to rigorous designs. Early PoS-based designs (e.g., [4], [5], [6], [7]) and PoW-based systems, such as Bitcoin, were initially developed in an *ad hoc* manner. Over time, however, the field has shifted towards a more rigorous approach, requiring formal security definitions and proofs. Notable works like [8] and [9] have analyzed Bitcoin's PoW-based blockchain within the *cryptographic setting*, where malicious players can deviate arbitrarily, while honest players follow the protocol specification. Their analysis demonstrated that Bitcoin can achieve essential security properties, including common prefix, chain quality, and chain growth. Similarly, research has explored PoS-based, Bitcoin-like consensus protocols, such as [10], [11], [12], though these systems remain vulnerable to *predictability* attacks.

DOI 10.1109/EuroSP63326.2025.00035

(Un)predictability. Intuitively, predictability in a protocol means that certain players know in advance that they will be selected to generate blocks. Reference [13] investigated the predictability of PoS in incentive-driven scenarios, where players may deviate from the protocol to maximize their profits. Attackers can exploit predictability to lower the cost or difficulty of incentive-driven attacks, such as selfish-mining [13] or bribery [12].

In a selfish-mining attack, an attacker gains an unfair advantage by selectively withholding or releasing blocks. Predictability allows attackers to craft more effective strategies for these attacks. In a bribery attack, the attacker attempts to influence players to work on specific chains, facilitating double-spending or censorship. By predicting which players are likely to mine upcoming blocks, attackers can target them for bribery. These attacks undermine fairness and disincentivize honest participation in the protocol.

To counter this, PoS protocols must minimize predictability to mitigate these risks. Ideally, a PoS protocol should strive for *best possible* unpredictability to neutralize predictability-based attacks, thereby preserving fairness and encouraging honest participation.

PoS via BFT techniques. Before presenting our results, it is important to note that conventional Byzantine Fault Tolerance (BFT) techniques have recently seen significant improvements and have been applied to construct PoS consensus protocols. Notable examples of BFT-based PoS include Algorand [14], [15] and Ethereum's Casper [16], among others.

The key distinction between Bitcoin-like PoS and BFT-based PoS lies in their interactivity: the former is *non-interactive*, while the latter requires *multiple rounds of interaction* among participants. Unlike Bitcoin-like PoS (e.g., [10], [11], [12], and the protocol in this paper), BFT-based PoS selects a small committee from a large pool of participants. This **small committee**[1] then determines the block generator through several rounds of interaction.

It is worth noting that in the Casper protocol [16], the selected committee is publicly known, making it vulnerable to DDoS attacks. In contrast, Algorand [14], [15] keeps the committee hidden, though this comes at the cost of additional communication rounds. In this paper, we focus exclusively on Bitcoin-like PoS protocols.

1.1. Our Results

Our first result is that we formally define (the best possible) unpredictability in the cryptographic setting. We assert that a protocol achieves the best possible unpredictability if it only allows players to predict whether they can generate the next block, and nothing more.

Our major results are summarized as follows. First, based on the definition of the best possible unpredictability, we identify an interesting impossibility for a class of PoS protocols following a *single-extension* design framework. Existing provably secure Bitcoin-like PoS protocols (e.g., [10], [11], [12]) are all within the single-extension framework. Secondly, to overcome the impossibility, we introduce a new design framework, called *multi-extension*. We develop a novel D-distance-greedy strategy in the multi-extension framework, which allows us to design a provably secure Bitcoin-like PoS protocol. Finally, we present new analysis techniques to analyze the chain growth and the common prefix properties for PoS protocols in the multi-extension framework. We next elaborate on our major results.

Impossibility result of single-extension protocols. We formally define a *single-extension* framework for constructing PoS protocols, which is followed by existing PoS protocols such as Ouroboros Praos [10], Snow White [11], and Bagaria et al. [12]. In a single-extension protocol, each honest player selects the best chain and then attempts to extend it. Then, we identify an interesting impossibility result for single-extension PoS protocols: *For any single-extension PoS protocol that achieves the best possible unpredictability, it cannot be secure if the honest players control less than 73% of the stake.* It is noteworthy that we are the **first** to present the impossibility result for single-extension protocols. Previous results (e.g., [12]) demonstrate that some single-extension protocols can be secure with 73% honest stake. However, these works never claim the impossibility result. To prove the impossibility result, we formally define the single-extension protocol and introduce several new concepts.

New design: Overcoming the impossibility via multi-extension protocols. To overcome the impossibility, we introduce a new design framework for PoS protocols called *multi-extension*. In a multi-extension protocol, each honest player is allowed to extend multiple chains in a round, as opposed to just one. It is worth noting that designing a secure and practical multi-extension PoS protocol is challenging. For example, Bagaria et al. [12] have shown that a protocol allowing honest players to extend slightly shorter chains than the longest chain is vulnerable to "balance attacks" which can break security. Fortunately, we achieve a specific design that follows a novel *D-distance-greedy* strategy, that can be proven to be secure.

Our *D-distance-greedy* strategy allows the honest player to extend *a set of chains*, where the parameter D is a positive integer. Specifically, in this strategy, the players extend a set of chains that are "close" to the longest chain. We say a chain is "close" to the longest chain if a common prefix can be obtained by removing the last D blocks from the longest chain. This ensures that the honest players extend a set of chains that share the same prefix, making it impossible for adversaries to launch balance attacks. By using this strategy, we can construct a protocol that achieves the best possible unpredictability. At the same time, the protocol can be proven to be secure with a smaller fraction (e.g., 57%) of honest stake. We have implemented the protocol proposed in this paper and made it available as an open-source project[2].

New security analysis techniques for multi-extension protocols. Analyzing the security of multi-extension protocols poses substantial technical challenges. To address these, we introduce several novel techniques to establish that our protocol satisfies key security properties, includ-

1. For instance, Casper recommends a minimal committee size of 111, while in practice, the size can be in the hundreds—still significantly smaller than the total number of PoS participants in Ethereum.

2. The open-source project can be accessed at: https://github.com/fractalplatform/fractal.

ing chain growth, common prefix, and (the best possible) unpredictability.

Chain growth. To analyze the chain growth property, we introduce a new Markov chain analysis framework designed for multi-extension protocols. Simply put, we consider the collection of chains "close" to the longest chain as a state of the Markov chain. By calculating the transition probabilities, we can determine the Markov chain's stationary distribution, which represents the probabilities that states appear in a random walk. For each state, if some longest chains are extended, then the total length of the chain will increase by 1. Based on the number of longest chains in each state, we can calculate the expected chain growth. Using the Chernoff bound, we transform this expectation into a lower bound for chain growth.

Common prefix. Previous analysis of Bitcoin's PoW consensus [8], [9] shows that the key factor for establishing the common prefix property is that honest parties can contribute only one block at a block height. Breaking this property requires the adversary to generate more blocks than the honest parties, which is infeasible due to the majority control of mining power by the honest parties. Our proposed protocol aims to defend against nothing-at-stake attacks by allowing players to extend multiple chains. Thus, we can no longer guarantee that honest parties contribute only one block per block height. To analyze the common prefix property for the multi-extension protocol, we introduce the notions of *virtual block-sets* and *virtual chains*, and then define the *common prefix property w.r.t. virtual chains*. We can prove the common prefix w.r.t. virtual chains by showing that the honest players only contribute at most one virtual block-set at a block height. Afterward, we show that the standard common prefix property can be reduced to common prefix w.r.t. virtual chains.

Extensions: Full-fledged blockchain and adaptive stake registration. Our protocol can be "upgraded" to a regular blockchain to include payload, such as transactions, in the blocks. The chain in our protocol serves as a randomness beacon to select a PoS player to generate a new block and extend the blockchain. Similar to [17], we also allow new players to join the system and participate in the process of extending chains, as long as they have registered their stake a specified number of rounds earlier. This ensures that the adversary cannot gain any extra advantage.

1.2. Organization

In Section 2, we introduce an analytical framework for PoS protocols. In Section 3, we show an impossibility result for the *single-extension* PoS protocols. In Section 4, we construct a new PoS protocol in the *multi-extension* design framework, bypassing the impossibility of the single-extension PoS protocols. Supplemental materials for Section 2, Section 3, and Section 4, are available in Appendices A, B, and C, respectively.

Next, we provide the security analysis of our new PoS protocol. Section 5 offers a concise overview of the security analysis, with detailed analysis provided in the Supplementary materials due to space constraints. Specifically, in Appendix D, we provide a new analytical framework to analyze the chain growth property of a PoS protocol in the multi-extension design framework. Then, Appendix E presents a new analytical framework to analyze the common prefix property for our new protocol. In Appendix F, we show the chain quality and the best possible unpredictability properties of the new protocol.

Additionally, in Section 6, we discuss how to extend our protocol into a full-fledged blockchain protocol and enable players to register their key pairs adaptively. Finally, in Section 7, we review related work in the field.

2. Security Model

2.1. Blockchain Protocol Executions

The security of Bitcoin-like PoW-based protocols has been rigorously investigated by [8] and then by [9] in the cryptographic setting. Below we define a framework for analyzing Bitcoin-like PoS-based blockchain protocols. We note many formulation ideas are taken from the previous frameworks [8], [9], and our analysis will be in the semi-synchronous network setting as in [9].

The execution of a PoS blockchain protocol. Following Canetti's formulation[18], we present an abstract model for a PoS blockchain protocol Π in the hybrid world of the semi-synchronous network communication functionality, the random oracles, and certain initialization functionality, similarly drawn from [9].

We consider the execution of blockchain protocol Π that is directed by an environment $\mathcal{Z}(1^\kappa)$, where κ is a security parameter. A necessary condition in all common blockchain systems is that all players agree on the first, i.e., the *genesis block*, which consists of the identities (e.g., public keys) and the stake distribution of the players. During execution, the stake distribution can be changed, the player can join or leave the system. For simplicity, we focus on the idealized "flat" model where all PoS-players have the same number of stakes.

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

In any round r, each PoS-player $P \in \mathcal{P}$, with a local state st, receives a message from the environment $\mathcal{Z}$, and potentially receives messages from other players. Then, it executes the protocol, broadcasts a message to other players, and updates its local state. Note that the network is under the control of the adversary $\mathcal{A}$, meaning that $\mathcal{A}$ is responsible for delivering all messages sent by players. At any round r of the execution, the environment $\mathcal{Z}$ can send message $(\textsc{Corrupt}, P)$, where $P \in \mathcal{P}$, to $\mathcal{A}$. Then, the adversary $\mathcal{A}$ will have access to the player's local state and control P. Let $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$ be a random variable denoting the joint $\texttt{VIEW}$ of all players (i.e., all their inputs,

random coins, and messages received, including those from the random oracle) in the above protocol execution; note that this joint view fully determines the execution. Protocol players are allowed to join the protocol execution $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$. We assume that when (honest) PoS-players leave the protocol execution, they will *erase* their own local internal information. We remark that players may sell their own secret keys, and this is out of scope of this paper.

Block and 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 genesis block. We use $\mathsf{len}(\mathcal{C})$ to denote *blockchain length*, i.e., the number of blocks in blockchain $\mathcal{C}$; and here $\mathsf{len}(\mathcal{C}) = \ell$. (Note that since all chains must consist of the genesis block, we do not count it as part of the chain's length.) We use sub blockchain (or subchain) for referring to a segment of a chain; here for example, $\mathcal{C}[j,m]$, 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 $\mathcal{C}[\neg\kappa]$.

2.2. Chain Growth, Common Prefix, Chain Quality

Previously, several fundamental security properties for Bitcoin-like PoW-based blockchain protocols have been defined: *chain growth* [19], *common prefix* [8], [9], and *chain quality* [8]. 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, characterized by the parameter $\mu \in (0,1)$, aims to indicate the ratio of contributions from honest players that are contained in a sufficiently long and continuous part of an honest chain, is at least μ. We follow the same path to define the security properties for Bitcoin-like PoS-based blockchain protocols, as below.

Definition 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* $r_2 - r_1 = \Omega(\kappa)$*, 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 (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 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 any* $\ell = \Omega(\kappa)$ *consecutive blocks of* $\mathcal{C}$*, the ratio of honest blocks is at least* μ*.*

2.3. Unpredictability

Unpredictability has been investigated by [13] in incentive-driven settings. At a high level, predictability means that (certain) protocol players are aware that they will be selected to generate blocks of the blockchain, *before* they actually generate the blocks. (Please see several predictability-based attacks in Supplemental material A.1.)

We investigate the unpredictability property in the cryptographic setting. Consider a malicious player $P \in \mathcal{P}$ at round r. Let $\mathtt{VIEW}^r$ be the view of all players at round r, and $\mathcal{C}^r$ be the best (valid) chain of all players in $\mathtt{VIEW}^r$. At round r, the adversary $\mathcal{A}$ attempts to predict if the (malicious) player P can extend the best chain at a future round r', where $r' > r$. Let $z_P^{r'} \in \{0,1\}$ be a prediction: $z_P^{r'} = 1$ means that $\mathcal{A}$ predicts that player P can extend the best chain at round r'. Now we need to introduce another random variable $\bar{z}_P^{r'}$ to indicate if the malicious player P indeed is able to extend the best chain at round r' (as the adversary predicated at an early round r, where $r < r'$) or not. Let $\mathtt{VIEW}^{r'}$ be the view of all players at round r', and $\mathcal{C}^{r'}$ be the best valid chain of all players in $\mathtt{VIEW}^{r'}$. We set $\bar{z}_P^{r'} = 1$ if there exists a chain $\mathcal{C} = \mathcal{C}^{r'}\|B$ in $\mathtt{VIEW}$ with a block B generated by player P at round r', otherwise we set $\bar{z}_P^{r'} = 0$.

Consider a view $\mathtt{VIEW}$, protocol round r, and a malicious player P. For $L \in \mathbb{N}$ and a prediction $z_P^{r'}$ where $r' > r$, we define the predicate $\mathsf{predictable}$ to be true if the prediction $z_P^{r'}$ accurately predicts whether or not player P can generate a new chain at round r' that is L blocks longer than the longest chain at round r.

More concretely, we define $\mathsf{predictable}(\mathtt{VIEW}, P, L, r, r', z_P^{r'}) = 1$ if and only if the following three conditions hold: *(i)* $r' > r$; *(ii)* $\mathsf{len}(\mathcal{C}^{r'}) + 1 - \mathsf{len}(\mathcal{C}^r) = L$; and *(iii)* $z_P^{r'} = \bar{z}_P^{r'}$. We say a player is L-unpredictable at round r if the adversary cannot predict whether or not the player can generate the next L blocks. Here, we consider $L = 2$, since the best possible unpredictability for any PoS protocol is 2-unpredictability. In any PoS protocol, all players can *always* predict whether or not they can generate the next block. In other words, 1-unpredictability cannot be achieved, and 2-unpredictability is the best. (A generalized definition can be found in Supplemental material A.2.)

Definition 4 (The best possible unpredictability)**.** *Consider a blockchain protocol* Π*. We say protocol* Π *achieves the best possible unpredictability if for all* PPT $\mathcal{Z}, \mathcal{A}$*, for any malicious player* P *at any round* r*, we have,*

$$\Pr\left[\begin{array}{l}\mathtt{VIEW} \leftarrow \mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}} \\ (r', z_P^{r'}) \leftarrow \mathcal{A}(P, r, \mathtt{VIEW}^r)\end{array} \;\middle|\; \mathsf{predictable}(\mathtt{VIEW}, P, 2, r, r', z_P^{r'}) = 0\right] > 1 - \mathsf{negl}(\kappa),$$

where $\mathsf{negl}(\cdot)$ *is a negligible function.*

3. An Impossibility Result

In this section, we present an impossibility result for a class of PoS protocols, namely, *single-extension PoS*

protocols. We first define the single-extension PoS framework in Section 3.1. Then, in Section 3.2, we state the impossibility result for the single-extension PoS protocols.

3.1. Single-Extension Proof-of-Stake Protocols

Intuitively, for a Bitcoin-like PoS protocol in the *single-extension* framework, in each round, each honest player identifies only a single "best chain," and then extends this chain.

Definition 5 (Single-extension framework for PoS protocols)**.** *A single-extension PoS protocol* Π *is executed by a set of player* $\mathcal{P}$*. Initially, each player* $P \in \mathcal{P}$ *holds a key pair* $(\mathrm{SK}, \mathrm{PK})$*. The protocol* Π *is parameterized by deterministic algorithms* $(\mathsf{Context}, \mathsf{Extend}, \mathsf{Validate}, \mathsf{BestChain})$ *as follows:*

- *The validation algorithm* $\mathsf{Validate}$ *takes a chain* $\mathcal{C}$ *and a round number* r *as inputs and returns* 1 *if the chain* $\mathcal{C}$ *is valid at round* r*, and returns* 0 *otherwise.*
- *The context extraction algorithm* $\mathsf{Context}$ *takes a valid chain* $\mathcal{C}$ *as input and returns a context* η*. The context* η *is the hash value of some blocks on the chain* $\mathcal{C}$*, based on some hash function* $\mathsf{hash} : \{0,1\}^* \to \{0,1\}^*$*, which is treated as a random oracle.*
- *The extension algorithm* Extend *(parameterized by* $p \in (0,1)$*), takes a context* η*, a round number* r*, and a secret key* SK *and returns a new block* B *with probability* p *or* $\perp$ *(if no new block is generated).*
- *The best chain algorithm* $\mathsf{BestChain}$ *takes a set of valid chains* $\mathbb{C}$ *and returns the longest chain* $\mathcal{C}_{\mathrm{best}}$ *as the best chain. Here, the honest player will only* ***extend a single chain****, i.e., the longest chain* $\mathcal{C}_{\mathrm{best}}$*.*

The execution of a single-extension protocol consists of two phases as follows.

Blockchain initialization phase. *In this phase, the genesis block will be created; the genesis block consists of a randomness, the public information, and the stake distribution of the players. Consider an (initial) group of PoS-players* $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ *and a security parameter* κ*. Each player* $P_j \in \mathcal{P}$ *generates a pair of public key* PK_j *and private key* SK_j*. The public keys of all players are stored in the genesis block, denoted by* B_0*, of the blockchain system.*

Blockchain extension phase. *In each round* r*, a player* P *with the secret key* SK *proceeds as follows. First, the player* P *computes* $\mathcal{C}_{\mathrm{best}} := \mathsf{BestChain}(\mathbb{C})$*. Here the local set of chains* $\mathbb{C}$ *consists of all verified valid chains that are received (or generated) by* P*. Then, the player* P *uses the function* $\mathsf{Context}$ *to compute the context* η *in the best chain* $\mathcal{C}_{\mathrm{best}}$*, i.e.,* $\eta := \mathsf{Context}(\mathcal{C}_{\mathrm{best}})$*. Finally, based on the context* η*, the current round number* r*, and the secret key* SK*, the player* P *uses the function* Extend *to determine whether or not it can generate a new block. If the player* P *can generate a new block* B*, it creates a new chain* $\mathcal{C} := \mathcal{C}_{\mathrm{best}} \| B$*, adds* $\mathcal{C}$ *to the set of chains* $\mathbb{C}$*, and broadcasts* $\mathcal{C}$ *to all other players. The pseudocode can be found in Algorithm 1.*

We remark that the state-of-the-art PoS protocols (e.g., [11], [10], [12]) can be categorized as single-extension PoS protocols (see Supplemental material B.1 for details). However, we only focus on Bitcoin-like PoS protocols, where players can generate new blocks in a *non-interactive* fashion by solving PoS puzzles. This framework *cannot* be applied for BFT-like protocols (e.g., Algorand [14], [15]), where players must interact with each other to generate new blocks.

Algorithm 1: A single-extension PoS protocol Π.

State : Initially, the set of chains $\mathbb{C}$ only consists of the genesis block. At round r, the PoS-player $P \in \mathcal{P}$, with $(\mathrm{SK}, \mathrm{PK})$ and local chain set $\mathbb{C}$, proceeds as follows.

Upon receiving a chain $\mathcal{C}'$, set $\mathbb{C} := \mathbb{C} \cup \{\mathcal{C}'\}$ after verifying $\mathsf{Validate}(\mathcal{C}', r) = 1$;

Set $\mathcal{C}_{\mathrm{best}} := \mathsf{BestChain}(\mathbb{C})$;

Set $\eta := \mathsf{Context}(\mathcal{C}_{\mathrm{best}})$; Set $B := \mathsf{Extend}(\eta, r, \mathrm{SK})$;

if $B \neq \perp$ **then**

  Set $\mathcal{C} := \mathcal{C}_{\mathrm{best}} \| B$; Add $\mathcal{C}$ to the set $\mathbb{C}$; Broadcast $\mathcal{C}$;

3.2. Impossibility Result for Single-Extension Proof-of-Stake Protocols

We present an impossibility result for single-extension PoS protocols. Concretely, consider a PoS protocol in the single-extension framework, we can show that, if the PoS protocol achieves the best possible unpredictability, then the protocol cannot maintain security properties, such as the common prefix, when honest players control less than 73% of the stake.

Let N be the number of players and ρ be the fraction of malicious players in the protocol execution. Let p be the probability that a player can extend a chain in a round. The probability that honest players extend a chain in a round is $\alpha = 1-(1-p)^{N\cdot(1-\rho)}$. Similarly, the probability that the adversary extends a given chain is $\beta = 1-(1-p)^{N\cdot\rho} \approx \frac{\rho}{1-\rho}\cdot\alpha$, if p is sufficiently small. We are now ready to state the impossibility theorem for protocol Π.

Theorem 1. *Consider a single-extension PoS protocol* Π *achieves the best possible unpredictability. If* $\alpha < e \cdot \beta$*, where* $e = 2.72$*, then* Π *cannot achieve common prefix property.*

Proof Sketch. The roadmap for the proof is depicted in Fig. 1.

We prove this impossibility through a new notion called *distinct-context-extension* property. Essentially, a protocol is said to satisfy this property when all different chains in the protocol's execution have distinct contexts. Then, we prove this theorem in the following two steps.

First, we show that if the single-extension PoS protocol Π achieves the best possible unpredictability, it must achieve distinct-context-extension property. Intuitively, if two chains share some common context, players can extend both chains simultaneously. Therefore, when a player receives one chain, he can predict whether or not he can extend the other chain in the future. This contradicts the best possible unpredictability. Secondly, we show that if the single-extension PoS protocol Π achieves distinct-context-extension property, then the adversary can amplify its stake by a factor $e = 2.72$ by extending all valid chains. Concretely, we model the chain extension of the adversary as a random tree, where each branch of this tree represents

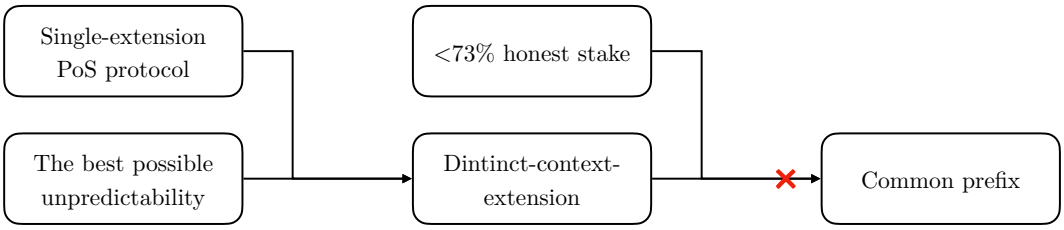

Figure 1. The roadmap for the proof of our impossibility result (Theorem 1).

a chain in the blockchain. To bound the growth rate of the chain, we first bound the number of branches in the random tree, and then, based on the number of branches and the growth rate of each branch, we can determine the maximum length of all branches in the random tree using Chernoff bound, thereby deriving the amplification rate of the stake.

Therefore, if $\alpha < 2.72\beta$, the adversary can extend the chain faster than the honest player with only $27\%(=\frac{1}{1+2.72})$ of the stake. More specifically, the adversary can break the common prefix property by keeping its chain hidden for a sufficiently long period and then publishing the hidden chain. As the chain of the adversary is longer, it will become the new longest chain. Since the new best chain does not share a common prefix with the old best chain of the honest players, the common prefix property does not hold. □

Due to the lack of space, we provide the formal proof of Theorem 1 in Supplemental material B.2. Specifically, we present the formal definition of distinct-context-extension property in Supplemental material B.2.1, and prove the impossibility result in two steps, which can be found in Supplemental materials B.2.2 and B.2.3, respectively.

We remark that we are the first to show the impossibility result for single-extension PoS protocols. To show the impossibility, we need to formally define the single-extension PoS protocol and introduce a new concept of a distinct-context-extension property. Although similar proof of the second step has been shown in [12], the impossibility result has not been presented in those works.

In addition, this impossibility does not hold for single-extension PoW protocols. In these protocols, the property of distinct-context-extension is also necessary for the best possible unpredictability. However, the cost of computing power to generate a new block in a PoW protocol is prohibitively high, preventing players from extending multiple chains to improve their chances of generating new blocks. On the other hand, the computing power required to extend a chain in a PoS protocol is very cheap, making it possible for an adversary to extend multiple chains and increase their chances of generating new blocks. Therefore, a PoW protocol can achieve the best possible unpredictability and maintain security when 51% of the mining power is honest.

4. Greedy Strategies: How to overcome the impossibility

In Section 3, we have obtained the impossibility of single-extension PoS protocols. In this section, we introduce a *multi-extension* framework that allows honest players to extend multiple chains (see Supplemental Material C.1 for the definition of a multi-extension protocol). We then present greedy strategies that follow this framework. In these strategies, honest players are allowed to extend multiple chains that are "close" to each other. Additionally, we design a new tiebreak rule for the multi-extension protocol to maximize the chain growth of honest players. Our protocol can achieve the best possible unpredictability while requiring a much smaller fraction (e.g., 57%) of honest stake to achieve security properties.

For simplicity, we consider the idealized "flat" model where all PoS-players have the same number of stakes and register exactly one public key in the genesis block. (In Section 6.2, we extend it to the non-flat model where the PoS-players may have different numbers of stakes.) Plus, we assume all protocol players have their stake registered at the beginning of the protocol execution. (In Section 6.3, we will consider a dynamic stake distribution, allowing new players to join during the protocol execution.)

4.1. Greedy Strategies

We allow the players to take a *greedy* strategy to extend the chains in a protocol execution: instead of extending a single best chain (i.e., the longest chain), the players are allowed to extend *a set of best chains*, expecting to extend the best chain faster. This is possible because extending chains in a PoS protocol is "very cheap." We remark that the set of best chains should be carefully chosen; otherwise, the protocol may not be secure. In our greedy strategy, the honest player extends 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.

Distance-greedy strategies. Consider a protocol execution. In each player's local view, there are multiple chains, which can be viewed as a tree. Concretely, the genesis block is the root of the tree, and each path from the root to 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. First, we define the "distance" between two chains in a tree. Intuitively, we say the distance from a "branch" chain to a "reference" chain is d if we can obtain a prefix of the reference chain by removing the last d blocks of the branch chain.

Definition 6 (Distance between two chains)**.** *Let $\mathcal{C}$ be a chain of length ℓ, and $\mathcal{C}_1$ be a chain of length ℓ_1. We view $\mathcal{C}$ as the "reference" chain, and $\mathcal{C}_1$ to be the "branch" chain. Next, we define the distance between $\mathcal{C}$ and $\mathcal{C}_1$, and we use* $\mathsf{distance}(\textit{branch chain} \rightarrow \textit{reference chain})$*,*

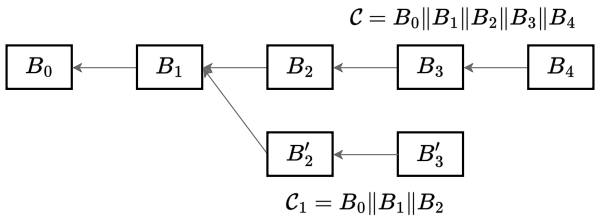

Figure 2. A toy example for illustrating the distance between two chains $\mathcal{C} = B_0\|B_1\|B_2\|B_3\|B_4$ and $\mathcal{C}_1 = B_0\|B_1\|B'_2\|B'_3$. Here, $\mathsf{distance}(\mathcal{C}_1 \to \mathcal{C}) = 2$, i.e., the distance from $\mathcal{C}_1$ to $\mathcal{C}$ is 2. Similarly, $\mathsf{distance}(\mathcal{C} \to \mathcal{C}_1) = 3$, i.e., the distance from $\mathcal{C}$ to $\mathcal{C}_1$ is 3.

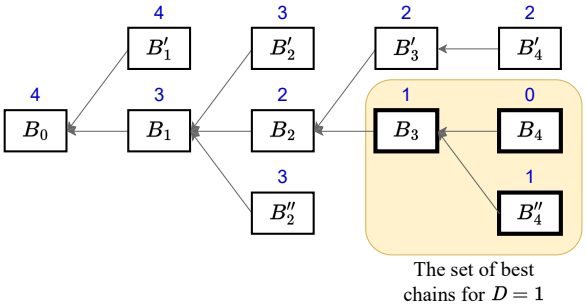

Figure 3. A toy example of 1-distance-greedy strategy. Here, the best chain is $\mathcal{C}_{\text{best}} = B_0\|B_1\|B_2\|B_3\|B_4$. The number in blue on top of each block denotes the distance from the *best chain* $\mathcal{C}_{\text{best}}$ (i.e., the branch chain) to the *reference chain* that consists of a sequence of blocks from the genesis block B_0 to that block. The bold blocks in the yellow area are the last blocks of the chains in the set of best chains. The honest players will extend the bold blocks (B_3, B_4, B''_4).

i.e., $\mathsf{distance}(\mathcal{C}_1 \to \mathcal{C})$ *to denote the distance. 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 between the reference chain* $\mathcal{C}$ *and the branch* $\mathcal{C}_1$ *is* d*, and we write* $\mathsf{distance}(\mathcal{C}_1 \to \mathcal{C}) = d$.

Remark 1. *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 possible that* $\mathsf{distance}(\mathcal{C} \to \mathcal{C}_1) \neq \mathsf{distance}(\mathcal{C}_1 \to \mathcal{C})$*. For example, in Fig. 2, the distance of* $\mathcal{C}$ *from* $\mathcal{C}_1$ *is* 3*, i.e.* $\mathsf{distance}(\mathcal{C} \to \mathcal{C}_1) = 3$*, 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.,* $\mathsf{distance}(\mathcal{C} \to \mathcal{C}) = 0$.

Now we are ready to define the distance-greedy strategies. Intuitively, a player following a distance-greedy strategy will try to extend a *set of best chains*, where the distance between the best chain and the chains in this set is quite small. Here, we consider the best chain as the branch chain and all other chains in the set of best chains as the reference chains. By the definition of the distance, we can obtain a common prefix of all reference chains by removing the last few blocks of the branch chain. Jumping ahead, we will use this observation to prove the common prefix property of our protocol. Formally, we have the following definition.

Definition 7 (D-distance-greedy strategy)**.** *Consider a blockchain protocol execution. Let* P *be a player of the protocol execution, and let* $\mathbb{C}$ *be the set of chains in player* P*'s local view. Let* $\mathcal{C}_{\text{best}}$ *be the longest chain at round* r*, where* $\ell = \mathsf{len}(\mathcal{C}_{\text{best}})$*. Let* D *be non-negative integers. Define a set of chains* $\mathbb{C}_{\text{best}}$ *as*

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

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

A toy example of the 1-distance-greedy strategy can be found in Fig. 3.

4.2. The Multi-Extension Protocol $\Pi^{\bullet}$

We present a new protocol $\Pi^{\bullet}$ to achieve the best possible unpredictability while only requiring a much smaller fraction (e.g., 57%) of honest stake to achieve the security properties. For simplicity, we construct a protocol in which the payloads in all blocks are empty. (We will extend it to include payloads in Section 6.1.) Protocol $\Pi^{\bullet}$ uses a unique digital signature scheme and a hash function as building blocks. For completeness, we present the definition of the unique digital signature scheme in Supplemental material C.2.

Blockchain initialization phase. In this phase, the genesis block will be created. Here, the genesis block consists of a randomness, the public keys of the players. Given a 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 $(\mathrm{SK}_j, \mathrm{PK}_j) \leftarrow \mathsf{uKeyGen}(1^{\kappa})$, 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 $B_0 = \langle(\mathrm{PK}_1, \mathrm{PK}_2, \cdots, \mathrm{PK}_n), \mathsf{rand}\rangle$[3].

Blockchain extension phase. Following the design of a multi-extension framework, our protocol is parameterized by four algorithms: $\mathsf{Context}^{\bullet}$, $\mathsf{Mining}^{\bullet}$, $\mathsf{Validate}^{\bullet}$, and D-$\mathsf{BestChainSet}^{\bullet}$. In our protocol, 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 D, i.e., $\mathsf{distance}(\mathcal{C}_{\text{best}} \to \mathcal{C}) \leq D$.

The procedure D-$\mathsf{BestChainSet}^{\bullet}$ 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 D-$\mathsf{BestChainSet}^{\bullet}$ iterates through all chains in the local state and uses algorithm $\mathsf{Validate}^{\bullet}$ to remove the invalid chains. Here, the algorithm $\mathsf{Validate}^{\bullet}$ takes a chain $\mathcal{C}$ and the current round r as input and evaluates every block of the chain $\mathcal{C}$ sequentially. Let ℓ be the length of $\mathcal{C}$. Starting from the head of $\mathcal{C}$, for every block $\mathcal{C}[i]$, for all $i \in [\ell]$, in the chain $\mathcal{C}$, the procedure D-$\mathsf{BestChainSet}^{\bullet}$ verifies that 1) $\mathcal{C}[i]$ is linked to the previous block $\mathcal{C}[i-1]$ correctly, 2) the hash inequality is correct, and 3) the signature is correctly generated by the player. Then, the procedure D-$\mathsf{BestChainSet}^{\bullet}$ selects the best chain $\mathcal{C}_{\text{best}}$ as the longest chain and iterates through the set of chains in the local state of the player to find all

3. For simplicity, we omit the stake distribution in the genesis block since all players have the same number of stake in the flat model.

Algorithm 2: PROTOCOL $\Pi^\bullet$

State : Initially, the set of chains $\mathbb{C}$ only consists of the genesis block. At round r, the PoS-player $P \in \mathcal{P}$, with (SK, PK) and local set of chains $\mathbb{C}$, proceeds as follows.

Upon receiving a chain $\mathcal{C}'$, set $\mathbb{C} := \mathbb{C} \cup \{\mathcal{C}'\}$ after verifying $\mathsf{Validate}^\bullet(\mathcal{C}', r) = 1$;

Compute $\mathbb{C}_{\text{best}} := D\text{-}\mathsf{BestChainSet}^\bullet(\mathbb{C})$;

for $\mathcal{C} \in \mathbb{C}_{\text{best}}$ **do**

  $\eta := \mathsf{Context}^\bullet(\mathcal{C})$; $B := \mathsf{Mining}^\bullet(\eta, r, \text{SK}, \text{PK})$;

  if $B \neq \perp$ **then**

    $\mathcal{C}_1 := \mathcal{C} \| B$; Broadcast $\mathcal{C}_1$;

// Algorithms $\mathsf{Context}^\bullet$, $\mathsf{Mining}^\bullet$, $\mathsf{Validate}^\bullet$, and D-$\mathsf{BestChainSet}^\bullet$.

$\mathsf{Context}^\bullet(\mathcal{C})$:

  $\ell := \mathsf{len}(\mathcal{C})$; $\eta := h(\mathcal{C}[\ell])$; Return η;

$\mathsf{Mining}^\bullet(\eta, r, \text{SK}, \text{PK})$:

  $\sigma := \mathsf{uSign}(\text{SK}, \langle \eta, r \rangle)$

  if $\mathsf{H}(\eta, r, \text{PK}, \sigma) < \mathtt{T}$ **then** Create new block $B := \langle \eta, r, \text{PK}, \sigma \rangle$; Return B;

  else Return $\perp$

$\mathsf{Validate}^\bullet(\mathcal{C}, r)$:

  Parse $\mathcal{C}$ into $B_0 \| B_1 \| \cdots \| B_\ell$;

  for $i \in [1, \ell]$ **do**

    Parse B_i into $\langle \eta_i, r_i, \text{PK}_i, \sigma_i \rangle$;

    if $h(B_{i-1}) \neq \eta_i$ or $\mathsf{H}(\eta_i, r_i, \text{PK}_i, \sigma_i) \geq \mathtt{T}$ or $\mathsf{uVerify}(\text{PK}_i, \langle \eta_i, r_i \rangle, \sigma_i) = 0$ or $r_i > r$ **then** Return 0;

  Return 1;

D-$\mathsf{BestChainSet}^\bullet(\mathbb{C})$:

  Set $\mathcal{C}_{\text{best}}$ as the longest chain in $\mathbb{C}$ and $\mathbb{C}_{\text{best}} = \{\mathcal{C}_{\text{best}}\}$;

  for $\mathcal{C} \in \mathbb{C}$ **do**

    if $\mathsf{distance}(\mathcal{C}_{\text{best}} \to \mathcal{C}) \leq D$ **then** $\mathbb{C}_{\text{best}} := \mathbb{C}_{\text{best}} \cup \{\mathcal{C}\}$;

the chains in which the distances from the best chain to those chains do not exceed D.

For a chain $\mathcal{C} = B_0 \| B_1 \| B_2 \| \ldots \| B_i$ in the set of best chains $\mathbb{C}_{\text{best}}$, the honest players P, with key pair (SK, PK), make attempts to extend the chain $\mathcal{C}$ as follows. Let r denote the current time (or round number). The player P first computes the context $\eta := \mathsf{Context}^\bullet(\mathcal{C})$. Here, algorithm $\mathsf{Context}^\bullet$ return the hash value of the last block on $\mathcal{C}$, i.e., $\mathsf{Context}^\bullet(\mathcal{C}) = h(B_i)$. A PoS-player P can successfully generate a new block if the following hash inequality holds: $\mathsf{H}(\eta, r, \text{PK}, \sigma) < \mathtt{T}$, where $\sigma := \mathsf{uSign}(\text{SK}, \langle \eta, r \rangle)$. The new block B_{i+1} is defined as $B_{i+1} := \langle \eta, r, \text{PK}, \sigma \rangle$. The pseudocode of our protocol $\Pi^\bullet$ can be found in Algorithm 2.

4.3. A New Tiebreak Rule for $\Pi^\bullet$

We design a new tiebreak rule for our D-greedy strategy. In a multi-extension protocol, the honest players extend all chains in the set (of best chains). The probability of generating a new best chain can vary depending on the number of chains in the set of best chains. This opens up opportunities for an adversary to slow down the chain growth by publishing a chain with the same length as the best chain but with fewer chains in the set of best chains. To defend against this attack, it is crucial to establish a tiebreak rule that maximizes the growth of the chain. In contrast, the probability of generating a new best chain in a single-extension protocol is constant; thus such a tiebreak rule (that maximizes the growth) is not needed in single-extension PoS protocols.

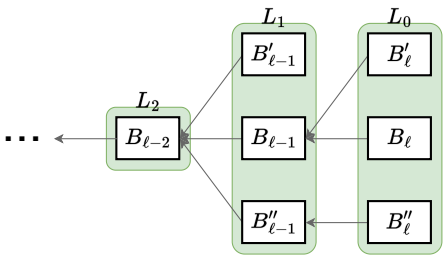

Figure 4. Partitioning a set of best chains $\mathbb{C}_{\text{best}}$ into multiple disjoint depth-based subsets for $D = 2$. Here, the set of best chains $\mathbb{C}_{\text{best}}$ is partitioned into 3 subsets L_0, L_1, L_2. Let ℓ be the length of the best chain. The 0-depth subset L_0 consists of 3 chains of length ℓ, i.e., the chains that have the last blocks are $B_\ell, B'_\ell, B''_\ell$. The 1-depth subset L_1 consists of 3 chains of length $\ell - 1$, i.e., the chains that have the last blocks are $B_{\ell-1}, B'_{\ell-1} B''_{\ell-1}$. The 2-depth subset L_2 consists of 1 chain of length $\ell - 2$, i.e., the chain that has the last block is $B_{\ell-2}$.

Intuitively, when there are two equally longest chains in a round, the best chain is selected based on the expected time to extend the chain and generate a new best chain. Honest players will choose the chain that is expected to be extended more quickly.

Recall that, the probability of generating a new best chain can vary depending on the number of chains in the set. As the number of chains in the set of best chains increases, the chance for honest nodes to generate a new longest chain also increases. To take advantage of this, our tiebreak rule prioritizes the chain with more chains in the set of best chains. This guarantees that the adversary cannot slow down the chain growth of the honest players.

Depth-based subsets. Before presenting the tiebreak rule, we introduce the definition of the *depth-based subsets*. Consider a protocol execution at a certain round, let $\mathcal{C}_{\text{best}}$ denote the best chain and $\mathbb{C}_{\text{best}}$ be the set of best chains. In our protocol execution, honest players follow the D-greedy strategy and make attempts to extend the set of best chains. As shown in Fig. 4, we partition the set $\mathbb{C}_{\text{best}}$ into $D + 1$ number of *disjoint subsets* based on the length of those chains. Let $\ell = \mathsf{len}(\mathcal{C}_{\text{best}})$ be the length of the best chain and for all $i \in [0..D]$, the i-depth-based subset L_i is the subset of chains with the length of $\ell - i$ in the set $\mathbb{C}_{\text{best}}$. That is, $L_i = \{\mathcal{C} \in \mathbb{C}_{\text{best}} : \mathsf{len}(\mathcal{C}) = \ell - i\}$.

Our new tiebreak rule. The tiebreak rule states that when there are two chains of the same length, the one with a faster expected time for further extension is chosen. Recall that, in our protocol, honest players extend all chains in the set of best chain. The tiebreak rule is based on the number of chains in the set. By utilizing this rule, we can ensure that the adversary is unable to slow down the growth of the chain for honest players.

In our protocol, honest players use a D-greedy strategy to extend the set of best chains. The length of the best chain increases by 1 when a chain in the 0-depth subset (i.e., a chain with the same length as the best chain) is extended. Therefore, having more chains in the 0-depth subset will allow honest players to extend the best chain faster. Additionally, when a chain in the 1-depth subset is extended, the number of chains in the 0-depth subset also increases. As a result, if the number of chains in the

Algorithm 3: Tiebreak rule

Input : Two chains $\mathcal{C}_{\text{best}}$, $\mathcal{C}'_{\text{best}}$ of length ℓ, the local set of chains $\mathbb{C}$
Output: Return the better chain between $\mathcal{C}_{\text{best}}$ and $\mathcal{C}'_{\text{best}}$

$\mathbb{C}_{\text{best}} := \emptyset$; $\mathbb{C}'_{\text{best}} := \emptyset$
for $\mathcal{C} \in \mathbb{C}$ **do**
  if $\mathsf{distance}(\mathcal{C}_{\text{best}} \to \mathcal{C}) \leq D$ **then**
    $\mathbb{C}_{\text{best}} := \mathbb{C}_{\text{best}} \cup \{\mathcal{C}\}$
  if $\mathsf{distance}(\mathcal{C}'_{\text{best}} \to \mathcal{C}) \leq D$ **then**
    $\mathbb{C}'_{\text{best}} := \mathbb{C}'_{\text{best}} \cup \{\mathcal{C}\}$
for $i \in [0..D]$ **do**
  $L_i := \{\mathcal{C} \in \mathbb{C}_{\text{best}} : \mathsf{len}(\mathcal{C}) = \ell - i\}$
  $L'_i := \{\mathcal{C} \in \mathbb{C}'_{\text{best}} : \mathsf{len}(\mathcal{C}) = \ell - i\}$
$i := 0$
while $i \leq D$ **do**
  if $|L_i| > |L'_i|$ **then**
    Return $\mathcal{C}_{\text{best}}$
  if $|L_i| < |L'_i|$ **then**
    Return $\mathcal{C}'_{\text{best}}$
  if $|L_i| = |L'_i|$ **then**
    $i := i + 1$
Return $\mathcal{C}_{\text{best}}$

0-depth subset is the same, having more chains in the 1-depth subset will also allow honest players to extend the best chain faster.

To break the tie of two equally longest chains, the players compare the number of chains in subsets at 0-depth, 1-depth, 2-depth, and so on, to determine the best chain (see Algorithm 3 for the pseudocode). If the number of chains in subsets is the same, we break the tie by comparing the number of chains in the next depth subset, and so on. If the tie still cannot be broken, we just choose the first chain.

More concretely, suppose we have two equally longest chains $\mathcal{C}$ and $\mathcal{C}'$. Let $\mathbb{C}_{\text{best}}$ and $\mathbb{C}'_{\text{best}}$ be the corresponding sets of best chains for $\mathcal{C}$ and $\mathcal{C}'$, respectively. For $i \in [0..D]$, let L_i denote the set of chains in $\mathbb{C}_{\text{best}}$ with length $\ell - i$, and let L'_i denote the set of chains in $\mathbb{C}'_{\text{best}}$ with length $\ell - i$. In other words, L_i and L'_i are i-depth subsets in $\mathbb{C}_{\text{best}}$ and $\mathbb{C}'_{\text{best}}$, respectively. To break the tie between $\mathcal{C}$ and $\mathcal{C}'$, the players follow this procedure: If the number of chains in L_0 is bigger than the number of chains in L'_0, i.e., $|L_0| > |L'_0|$, we say the chain $\mathcal{C}$ is better than the chain $\mathcal{C}'$. Similarly, if $|L'_0| > |L_0|$, we say the chain $\mathcal{C}'$ is better than the chain $\mathcal{C}$. If $|L_0| = |L'_0|$, we compare the number of chains in L_1, L'_1. If $|L_1| > |L'_1|$, we say we say the chain $\mathcal{C}$ is better than the chain $\mathcal{C}'$. Similarly, if $|L'_1| > |L_1|$, we say the chain $\mathcal{C}'$ is better than the chain $\mathcal{C}$. If $|L_1| = |L'_1|$, we compare the number of chains in L_2, L'_2, and so on. If we still cannot break the tie, the first chain will be selected as the best chain for simplicity.

5. Security Analysis: Overview

In this section, we provide an overview of the security analysis for protocol $\Pi^\bullet$. Notably, while the techniques outlined in [8], [9], [10], [12] offer valuable insights for analyzing protocols based on the single-extension design framework, $\Pi^\bullet$ deviates from this framework, necessitating new analysis techniques. For simplicity, our concrete analysis assumes a synchronous network. However, we note that for the semi-synchronous setting, a similar approach to that discussed in [9] can be applied.

We can prove the security properties of protocol $\Pi^\bullet$ under the assumption of honest majority of *effective stake*. Recall that in Section 3, we obtain that the adversary can amplify its stake by a factor $e = 2.72$, so we define the effective stake of the adversary as $\beta^\bullet = 2.72\beta$. Similarly, following the D-distance-greedy strategy, honest players can amplify their stake by a amplification ratio $\hat{\mathtt{A}}^\bullet_D$, and we define $\alpha^\bullet = \hat{\mathtt{A}}^\bullet_D \cdot \alpha$. Now, we formally state the theorem.

Theorem 2. *Consider an execution of multi-extension protocol $\Pi^\bullet$ in the random oracle model, where honest players follow the D-distance-greedy strategy while adversarial players could follow any arbitrary strategy. Additionally, all players have their stake registered at the beginning of the execution. Assume $(\mathsf{uKeyGen}, \mathsf{uKeyVer}, \mathsf{uSign}, \mathsf{uVerify})$ is a unique digital signature scheme, and $\alpha^\bullet = \lambda\beta^\bullet$, $\lambda > 1$. Then protocol $\Pi^\bullet$ achieves 1) chain growth, 2) common prefix, 3) chain quality, and 4) the best possible unpredictability properties.*

5.1. Chain Growth

We first analyze the lower bound of chain growth for our protocol $\Pi^\bullet$. Unlike the existing single-extension protocols [8], [9], [10], in multi-extension protocols, honest players may extend multiple chains in a round, and the probability of extending the best chain can vary between rounds. Therefore, a new analysis framework needs to be developed.

We propose a novel approach involving *a random walk within a Markov chain composed of multiple states* to analyze the chain growth property. Each state within the Markov chain serves as a representation of the set of best chains in a protocol round. During protocol execution, the state (set of best chains) may change. By evaluating the transition probabilities between states, we can derive the stationary distribution for each state, representing the probabilities of states appearing in a random walk. The probability of extending the best chain can be determined for each state, and leveraging the stationary distribution of states allows us to compute the expected chain growth.

For better understanding of our proof technique, here we present a *simplified Markov chain* to analyze the chain growth of protocols employing the D-distance-greedy strategy, where $D = 1$. Recall that p is the probability that a player can extend a chain in a round, N is the number of players, ρ is the fraction of malicious players, and the probability that honest players extend a chain in a round is $\alpha = 1 - (1-p)^{N\cdot(1-\rho)}$. Let

$$\mathsf{w}(n) = 1 - (1-p)^{N\cdot n\cdot(1-\rho)} \approx n\alpha \text{ (if } p \text{ is sufficiently small)}$$

be the probability that honest players generate *at least* one new block by trying to extend n blocks in a round.

First, using the concept of *depth-based subsets*, we can divide the set of best chains into two subsets, where subset L_0 contains all the longest chains, with $|L_0| = n_0$, and the other subset L_1 has a size of n_1. Since $D = 1$, it's evident that n_1 must always be equal to 1. (For example, in Fig. 3, we divide the set $\{B_3, B_4, B''_4\}$ into $\{B_4, B'_4\}$ and $\{B_3\}$, thus $n_0 = 2$ and $n_1 = 1$). Therefore, we can ignore n_1 and use $\langle n_0 \rangle$ to represent the set of best chains.

We assume the Markov chain is represented by a state space S and a transition matrix $\mathbf{T}$, which is an $|S| \times$

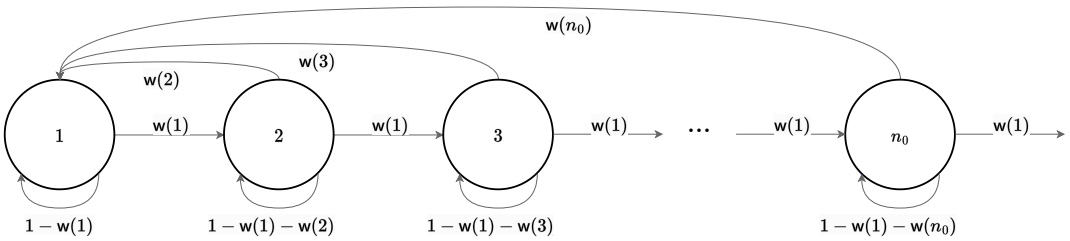

Figure 5. The complete state machine for the simplified Markov chain for $D = 1$.

$|S|$ matrix containing information on the probability of transitioning between states. We represent $\langle n_0 \rangle$ as the state of the Markov chain. Initially, we set all the values in $\mathbf{T}$ to 0. Then, for each state $\langle n_0 \rangle$, we update the transition matrix in a round based on the following three cases (see Fig. 5):

Case 1: A new best (longest) chain is generated. In other words, a chain in L_0 is extended. The probability that the honest players generate at least one new best chain is $\mathsf{w}(n_0)$. If p is sufficiently small, the extension of more than one chain in a round is negligible. In this case, the new subset L_0 only consists of one chain, i.e., the new longest chain. The state machine moves from state $\langle n_0 \rangle$ to state $\langle 1 \rangle$. We set $\mathbf{T}_{\langle n_0 \rangle, \langle 1 \rangle} := \mathsf{w}(n_0)$.

Case 2: A new chain is generated (and Case 1 does not happen). In other words, a chain in L_1 is extended. As the number of chains in L_1 is always one, the probability that the honest players generate a new chain is $\mathsf{w}(1)$. In this case, the state machine moves from state $\langle n_0 \rangle$ to state $\langle n_0 + 1 \rangle$. We set $\mathbf{T}_{\langle n_0 \rangle, \langle n_0+1 \rangle} := \mathsf{w}(1) \cdot (1 - \mathbf{T}_{\langle n_0 \rangle, \langle 1 \rangle})(\approx \mathsf{w}(1)$ if p is small).

Case 3: No new chain is generated. The state machine remains at the current state $\langle n_0 \rangle$ with a probability of $1 - \mathsf{w}(1) - \mathsf{w}(n_0)$. We set $\mathbf{T}_{\langle n_0 \rangle, \langle n_0 \rangle} := 1 - \mathsf{w}(1) - \mathsf{w}(n_0)$.

Let q_{n_0} be the stationary probability of the state $\langle n_0 \rangle$. For all state $s \in S$, we have

$$\sum_{s \in S} q_s = 1 \text{ and } q_s = \sum_{s' \in S} q_{s'} \cdot \mathbf{T}_{s',s}.$$

Additionally, for state $\langle n_0 \rangle$, the probability that the chain grows by 1 is $\mathsf{w}(n_0)$. Therefore, we can calculate the expected chain growth as

$$g_1 = \sum_{n_0=1}^{\infty} (q_{n_0} \cdot \mathsf{w}(n_0)) \approx \alpha \cdot \sum_{n_0=1}^{\infty} (q_{n_0} \cdot n_0) \approx 1.39\alpha.$$

Using the Chernoff bound, we can transform this expectation into a lower bound for chain growth. Next, we can extend this analysis to any arbitrary value of D.

We summarize our proof technique as follows. It can be seen that we design the simplified Markov chain using the information of the depth-based subsets. Recall that, by following the D-distance-greedy strategy, the honest players extend a set of best chains. The set of best chains can be partitioned into $D + 1$ subsets based on the depth of those chains, where the depth of a chain is computed based on the difference between its length and the length of the current best chain. For $i \in [0..D]$, the i-depth subset consists of all the chains that are i blocks behind the best chain. In each round, a new best chain is generated if a player extends a chain in the 0-depth subset, which consists of all the chains that have the same length as the current best chain. Further, a new chain is added to the i-depth subset if a chain in the $(i + 1)$-depth subset is extended, where $i \in [0..D - 1]$. Hence, we can analyze the chain growth based on the number of chains in those subsets. In the simplified Markov chain, each state represents a protocol round with specific numbers of chains in all depth-based subsets. The chain growth of our protocol can be estimated based on the expected number of chains in the 0-depth subset. The simplified Markov chain provides information about the number of chains in depth-based subsets, but it does not provide how many chains are removed from the set of best chains when a new best chain is generated. This leads to a worst-case scenario where the set of best chains only consists of the best chain and its prefixes, making it difficult to determine a good lower bound for chain growth, even with a large D.

To resolve the issue in the simplified Markov chain, we introduce an augmented Markov chain. The states of the augmented Markov chain contain more information. This helps us determine the number of chains that are removed after generating a new best chain. By doing so, we can avoid considering the worst-case scenario where the set of best chains only consists of the best chain and its prefixes. More concretely, we present the notion of *depth-distance-based subsets*. These subsets are selected based on *both* the length of the chains and their distance from the best chain. When a new best chain is generated, the distance from the new best chain to the chains in the subsets increases by one. As a result, we can obtain the number of chains in the new depth-distance-based subsets (when the new best chain is generated) based on the number of chains in the old depth-distance-based subsets (when the new best chain has not been generated). The states in the augmented Markov chain provide information about the number of chains in the depth-distance-based subset, allowing us to identify which chains belong to the new set of best chains when a new best chain is generated. This approach results in a better lower bound.

Detailed proofs and further elaborations are provided in Supplemental material D.

In Fig. 6, we summarize the lower bounds of the amplification ratio $\hat{\mathbf{A}}^{\bullet}_{D}$ (defined as g_D/α) using the simplified and augmented Markov chains. In Fig. 7, we show

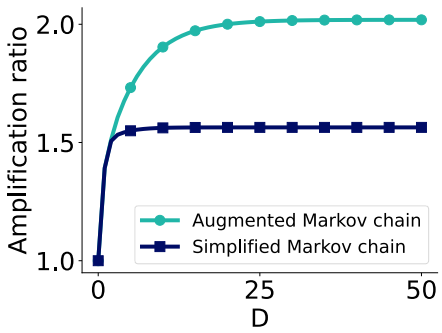

Figure 6. The lower bounds of the amplification ratio using the simplified and augmented Markov chains respectively.

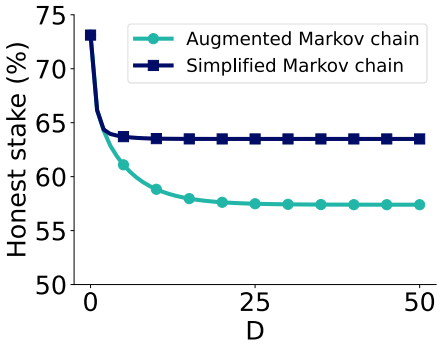

Figure 7. The upper bounds of the fraction on honest players using the simplified and augmented Markov chains respectively.

the fraction of honest stake to prove the security of the protocol based on the lower bounds of the amplification ratio. For example, by using the augmented Markov chain and set $D = 50$, we have $\hat{\mathbf{A}}^{\bullet}_{50} \geq 2.04$, i.e., protocol $\Pi^{\bullet}$ is secure if $57\%(= \frac{2.72}{2.04+2.72})$ of stake is honest.

5.2. Common Prefix

Then we prove that our protocol $\Pi^{\bullet}$ achieves common prefix property. Note that the adversary cannot use the blocks of honest players to compromise the security property. In other words, to break the common prefix property, the adversary must find a way to create two divergent chains, i.e., two chains that do not share a common prefix after removing the last $\kappa + D$ blocks. Recall that, we use the D-distance-greedy strategy, where honest players in the protocol execution will only extend the set of best chains. Based on the definition, the set of best chains must be "close." Therefore, the chains produced by honest players share a common prefix with the best chain, after removing the last D blocks. This prevents the adversary from using the chain of honest players to break the common prefix property.

To formally prove the common prefix property, we first introduce the notions of *virtual block-sets* and *virtual chains*. Intuitively, a virtual block-set consists of multiple blocks with the same height that are "close" to each other. More concretely, we first define two chains as "close" if they share a common prefix after removing the last D blocks. When two chains are "close," the last blocks of the two chains are also "close." Now the virtual chain consists of multiple virtual block-sets that are linked together. See Fig. 8 for a concrete example.

Then we can prove the *common prefix property w.r.t. virtual chains*. First, we introduce the concept of *honest virtual block-sets*. We say a virtual block-set is honest if the first generated block in the virtual block-set is generated by an honest player. As the honest players follow the D-greedy strategy, at each block height, there is at most one honest virtual block-set. Thus, to break the common prefix w.r.t virtual chains, the adversary needs to generate more virtual block-sets than the honest players. This requires the adversary to control the majority of the effective stake, which contradicts the assumption.

Finally, we show that the standard common prefix property can be reduced to common prefix w.r.t. virtual chains. The common prefix w.r.t. virtual chains property

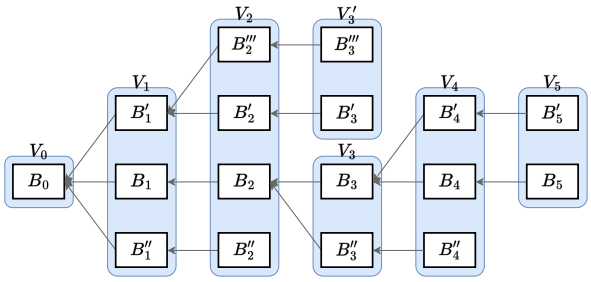

Figure 8. A toy example for the virtual block-sets and virtual chains with $D = 2$. Each block is represented by a solid rectangle and each virtual block-set is represented by a blue area that consists of multiple blocks. Here $V_0 = \{B_0\}$, $V_1 = \{B_1, B_1', B_1''\}$, $V_2 = \{B_2, B_2', B_2'', B_2'''\}$, $V_3 = \{B_3, B_3'\}$, $V_3' = \{B_3', B_3''\}$, $V_4 = \{B_4, B_4', B_4''\}$, $V_5 = \{B_5, B_5'\}$, In this case, the best chain is $\mathcal{C}_{\text{best}} = B_0 \| B_1 \| B_2 \| B_3 \| B_4 \| B_5$. Here, for all $i \in [0..5]$, we have $B_i \in V_i$. Thus, the best virtual chain is $\mathcal{V}_{\text{best}} = V_0 \| V_1 \| V_2 \| V_3 \| V_4 \| V_5$.

states that the best virtual chains of honest players share the same common prefix after removing the last κ virtual block-sets. Plus, based on the definition of the virtual block-set, the blocks in the same virtual block-set are "close" together. In other words, the chains that have those blocks as the last blocks share the common prefix after removing the last D blocks. Hence, the best chains of honest players share the same common prefix after removing the last $\kappa + D$ blocks.

We present more details in Supplemental material E.

5.3. Chain Quality and the Best Possible Unpredictability

Chain quality. After proving the chain growth and common prefix properties, the proof of chain quality will be very similar to the proof in [9]. Intuitively, to break the chain quality property, the adversary must generate κ consecutive blocks on the best chains, which requires the adversary to control the majority of the stake.

The best possible unpredictability. This property is evident in our protocol. Specifically, the players extract the context of a chain based on the hash value of the last block. Therefore, a player cannot know the hash value of the next block unless he generates the block himself. Hence, the player can only predict whether or not he can generate the next block.

In Supplemental material F, we present the analysis for these properties in detail.

6. Extensions

We introduce extensions to make our protocol more practical. In Section 6.1, we "upgrade" the protocol to a regular blockchain protocol, enabling it to include payloads (e.g., transactions). Next, in Section 6.2, we extend the protocol to a more realistic "non-flat" model. Finally, in Section 6.3, we adopt a strategy similar to that in [17], allowing new players to join and participate in extending the chain, provided their stake was registered a specified number of rounds earlier.

6.1. Full-Fledged Blockchain

We extend protocol $\Pi^{\bullet}$ to a full blockchain protocol by using it to generate a random beacon that selects the PoS-players that can generate new main-blocks with payloads. The blocks are linked together as a hash chain called the main-chain. Each PoS-player holds a pair of keys, $(\mathrm{SK}, \mathrm{PK})$, from a unique signature scheme $(\mathsf{uKeyGen}, \mathsf{uSign}, \mathsf{uVerify})$ and a pair of keys, $(\tilde{\mathrm{SK}}, \tilde{\mathrm{PK}})$, from a regular digital signature scheme $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$. We note that to achieve adaptive security, this regular signature scheme will be replaced by a forward-secure digital signature scheme [20].

More concretely, consider a best chain $\mathcal{C} = B_0 \| B_1 \| \cdots \| B_\ell$ with the corresponding main-chain $\tilde{\mathcal{C}} = \tilde{B}_0 \| \tilde{B}_1, \cdots \| \tilde{B}_\ell$. Here, the genesis block B_0 and the genesis main-block $\tilde{B}_0$ are the same. Once a new block $B_{\ell+1}$ is generated by a PoS-player, then the same PoS-player is selected to generate the new main-block $\tilde{B}_{\ell+1}$, in the following format $\tilde{B}_{\ell+1} = \langle \tilde{h}_\ell, B_{\ell+1}, X_{\ell+1}, \tilde{\mathrm{PK}}, \tilde{\sigma} \rangle$ where $\tilde{\sigma} \leftarrow \mathsf{Sign}_{\tilde{\mathrm{SK}}}(\tilde{h}_\ell, B_{\ell+1}, X_{\ell+1})$, $\tilde{h}_\ell := \mathsf{hash}(\tilde{B}_\ell)$, $X_{\ell+1}$ is payload. By linking the blocks in the main blockchain to the blocks in the blockchain, the security of the main blockchain protocol can be reduced to the security of the blockchain protocol.

6.2. Blockchain in the Non-Flat Model

Previously, we presented our ideas in the "flat" PoS model, where all players are assumed to hold the same number of stake and have an equal chance of being selected as the winning player in each round. However, in reality, PoS players have varying amounts of stake. Here, we will extend our design to reflect the more realistic "non-flat" model.

The genesis block, B_0, in the "non-flat" model consists of the public keys of the players, their respective stake distribution, and a randomness. Specifically, we have, $B_0 = \langle (\langle \mathrm{PK}_1, s_1 \rangle, \langle \mathrm{PK}_2, s_2 \rangle, \cdots, \langle \mathrm{PK}_N, s_N \rangle), \mathsf{rand} \rangle$. The number of stakes held by each player can vary in this model. For a PoS player with the key pair $(\mathrm{PK}, \mathrm{SK})$ holding s stake at round r, the hash inequality is changed as follows:

$$\mathsf{H}(\eta, r, \mathrm{PK}, \sigma) < s \cdot \mathtt{T}.$$

It is straightforward to see that the probability of a PoS player being selected to generate a new block is proportional to the amount of stake they control. If the player puts all their s stake in one account, their probability of being selected to sign a PoS block is $s \cdot p$. On the other hand, if they divide their s stake into s accounts, each with one stake, the probability of an individual account being selected is p. As the outputs of the hash function are independent for different verification keys, the total probability of the player being selected is $1 - (1-p)^s \approx s \cdot p$.

6.3. Defending Against Adaptive Registration

Our design can be further improved to allow for the dynamic registration and de-registration of players during the protocol's execution. As a reminder, the chain extension process relies on the hash inequality $\mathsf{H}(context, solution) < \mathtt{T}$, where $solution$ takes the form of (PK, σ). However, malicious players can use a "rejection re-sampling" strategy to generate their keys adaptively, taking advantage of the known $context$. In this strategy, a malicious player generates a key-pair $(\mathrm{PK}, \mathrm{SK})$ and then checks if the resulting (PK, σ) is a valid solution to the hash inequality. If it's not, the player repeats the key generation process. This increases the probability that the malicious player will be selected to extend the chain.

Adaptive registration. In line with the approach in [17], we defend against rejection re-sampling attacks by requiring new players to have their stake registered for a specified number of rounds before they can extend the chain. To join the protocol, player P generates two key pairs: $(\mathrm{SK}, \mathrm{PK}) \leftarrow \mathsf{uKeyGen}(1^\kappa)$ and $(\tilde{\mathrm{SK}}, \tilde{\mathrm{PK}}) \leftarrow \mathsf{KeyGen}(1^\kappa)$. P keeps SK and $\tilde{\mathrm{SK}}$ secret and broadcasts a registration transaction. Once registered, a player becomes eligible to extend the chain after η blocks have been added to the blockchain. Notably, players registered before the protocol begins (i.e., in the genesis block) are exempt from waiting η blocks to participate in chain extension. By enforcing this registration delay, malicious players are prevented from immediately extending the chain after registering new key pairs. While they could still attempt to register biased key pairs and extend the chain after several rounds, their inability to predict future events means they cannot select biased key pairs that would increase their chances of extending the chain at a distant point in time.

7. Related Work

We summarize the existing results for the designs and analysis of PoS protocols. The idea of using coins/stakes to construct cryptocurrency has been intensively considered. Since the inception of the idea in an online forum [21], several proof-of-stake proposals have been introduced or implemented (e.g., [4], [5], [6], [22], [7]). These proposals are *ad hoc* without formal security. Recently, several provably secure proof-of-stake-based blockchain proposals have been developed. More details can be found below.

Bitcoin-like proof-of-stake protocols. We focus on Bitcoin-like PoS protocols; these are closely related to the results in the current write-up. All these related protocols follow the single-extension framework. These related protocols include Snow White [11], Ouroboros Praos [10]

and Genesis [17], and a protocol by Bagaria et al. [12]. Note that all of the above protocols are single-extension protocols, and therefore *subject to the impossibility result discussed in Section 3*.

In Snow White [11], the protocol execution is divided into epochs, where each epoch consists of $\Omega(\kappa)$ blocks (for security parameter κ). 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). The Snow White protocol is based on the Sleepy protocol [23] (in which the new players are not allowed to join the system during the execution). The Snow White protocol allows new players to join the system but relies on external trust.

In Ouroboros Praos [10], similar to Snow White, the protocol execution is divided into epochs of $\Omega(\kappa)$ blocks. In each round, the player queries a verifiable random function (VRF) [24] 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. 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 [17], new players are allowed to join the protocol execution securely.

In Bagaria et al. [12], similar to 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. The authors also adopt the technique in [17] to allow new players to join the system.

BFT-based PoS protocols. Besides Bitcoin-like PoS protocols, in which the players generate blocks in a non-interactive fashion, BFT-like PoS protocols (including Algorand [14], [15], EOS [25], Dfinity [26], Casper [16]) have been constructed in an interactive fashion.

In Algorand [14], [15], a verifiable random function (VRF) has been used for selecting a committee of players. For each player, the opportunity to be selected is proportional to the number of stakes in the player's account. Then, the committee members run a Byzantine Agreement (BA) sub-protocol to jointly generate a block.

EOS [25] introduces a delegated proof-of-stake protocol, in which stakeholders (those who hold the stake on the blockchain) can select block producers through a continuous approval voting system. At the beginning of each round, 21 unique block producers are chosen based on the preference of votes cast by token holders. The selected block producers can create new blocks as long as 15 or more block producers agree.

Dfinity [26] proposes a four-layer consensus protocol to achieve consensus among players. The first layer registers the players. The second layer provides randomness for all higher layers. The third layer generates blocks. In each round, the protocol ranks the players based on the random beacon of that round. All players can generate new blocks, but each block has a different weight. The weight of the block is assigned based on the rank of the block procedure in that round. The best chain is selected as the "heaviest" chain in terms of accumulated block weight. The fourth layer provides fast finality of the block by using a threshold signature.

For Casper [16], two voting rounds are required. In each round, participants send signed messages to the leader. The leader then aggregates the signatures and forwards them to all participants. This process results in $2N$ message exchanges per round, totaling $4N$ for two rounds. The use of a pipeline technique can reduce the number of voting rounds by one. However, due to the known committee in each phase, this protocol is susceptible to DDoS attacks.

Note that, the above protocols (Algorand [14], [15], EOS [25], Dfinity [26], Casper [16]) require quadratic communication complexity to generate new blocks. Hot-Stuff [27] uses a threshold signature scheme to achieve linear communication complexity. Specifically, the block producer, i.e., the player who generates a new block, collects votes from other players. Then, they compute and broadcast a single threshold signature that proves at least $2/3$ of the players have voted for their block.

Hybrid PoS protocols using verifiable delay function (VDF). Reference [28] proposed the PoSAT protocol, which is a hybrid consensus using both proof-of-stake and verifiable delay function (VDF). In the PoSAT protocol, the VDF acts as a random beacon to generate blocks. After computing a VDF, players can instantly attempt to solve a hash puzzle to check if they can extend a PoS block from the output of the VDF. Since players cannot predict the output of VDFs, the PoSAT protocol is completely unpredictable, similar to Bitcoin.

We remark that the PoSAT protocol [28] is not a "pure" PoS protocol. In "pure" PoS protocols, the process of generating new blocks involves only the competition of "stake" (no other resources such as computing power). VDF-based PoS protocols allow the competition of sequential computation. The PoSAT protocol is based on the following assumptions: (1) both adversary and honest players have the same capability to execute sequential work: they take the same time to execute a VDF; and (2) honest players hold more stake than the adversary does. If one of the two assumptions does not hold, then the security of the PoSAT protocol cannot be ensured. However, in practice assumption (1) may not hold; it is possible that the adversary can have faster dedicated hardware for executing sequential work.

Security analysis for Bitcoin-like PoS protocols. Bagaria et al. [12] present a possible "balance attack" on multi-extension proof-of-stake protocols. We emphasize that **their "balance attacks" cannot be launched on our protocol**. There, the adversary will try to balance the length of the two chains *by publishing the block on the shorter chain*, to maintain two longest chains that are diverted for a long period. If the protocol is not carefully designed, the honest players may extend two chains that are diverted for a long period. Since the adversary only publishes the blocks on the shorter chain, the shorter chain will be extended faster and eventually catch up to have the same length as the other chain. Note that, in our protocol, the honest players only extend the chains that share a common prefix after removing the last few blocks. That is, the honest players will *never* extend two chains that

are diverted for a long period. Thus, the adversary is not able to launch the "balance attacks" on our protocol.

Based on the analysis in [11], [10], 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. Reference [29] improve the analysis for the consistency (i.e. common prefix property) of proof-of-stake-based blockchain protocols in the 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 [29], 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 player that can generate new blocks). Reference [30] extends the result from [29]. 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). Reference [31] introduces a new technique to analyze blockchain protocols (including Bitcoin and proof-of-stake-based protocols). The analysis shows that the best strategy for the adversary to break consistency is a 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.

The proof-of-stake 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. Reference [13] exploit the (un)predictability of proof-of-stake based protocols in a incentive-driven setting. The predictability allows the adversary to perform many incentive-driven attacks such as predictable selfish mining and predictable bribing. In this work, we investigate the unpredictability in a cryptographic setting.

Acknowledgment

This project was conducted when Phuc Thai was a PhD student at Virginia Commonwealth University. Phuc Thai and Hong-Sheng Zhou were supported in part by NSF grant CNS-1801470 and a VCU Research Quest grant. Lei Fan was supported by the National Natural Science Foundation of China (grant no. 62372293) and the Natural Science Foundation of Shanghai (grant no. 24BC3201300).

References

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

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

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

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

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

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

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

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

[9] Rafael Pass, Lior Seeman, and abhi shelat. Analysis of the blockchain protocol in asynchronous networks. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 643–673. Springer, Heidelberg, April / May 2017.

[10] Bernardo David, Peter Gazi, Aggelos Kiayias, and Alexander Russell. Ouroboros Praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 66–98. Springer, Heidelberg, April / May 2018.

[11] Phil Daian, Rafael Pass, and Elaine Shi. Snow White: Robustly reconfigurable consensus and applications to provably secure proof of stake. In Ian Goldberg and Tyler Moore, editors, *FC 2019*, volume 11598 of *LNCS*, pages 23–41. Springer, Heidelberg, February 2019.

[12] Vivek Bagaria, Amir Dembo, Sreeram Kannan, Sewoong Oh, David Tse, Pramod Viswanath, Xuechao Wang, and Ofer Zeitouni. Proof-of-stake longest chain protocols: Security vs predictability. *arXiv preprint arXiv:1910.02218*, 2019.

[13] Jonah Brown-Cohen, Arvind Narayanan, Alexandros Psomas, and S Matthew 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.

[14] Jing Chen and Silvio Micali. Algorand: A secure and efficient distributed ledger. *Theoretical Computer Science*, 777:155–183, 2019. https://arxiv.org/abs/1607.01341.

[15] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *SOSP*, 2017. https://eprint.iacr.org/2017/454.

[16] Vitalik Buterin and Virgil Griffith. Casper the friendly finality gadget. *CoRR*, abs/1710.09437, 2017.

[17] Christian Badertscher, Peter Gazi, Aggelos Kiayias, Alexander Russell, and Vassilis Zikas. Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 913–930. ACM Press, October 2018.

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

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

[20] Mihir Bellare and Sara K. Miner. A forward-secure digital signature scheme. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 431–448. Springer, Heidelberg, August 1999.

[21] 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.

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

[23] Rafael Pass and Elaine Shi. The sleepy model of consensus. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part II*, volume 10625 of *LNCS*, pages 380–409. Springer, Heidelberg, December 2017.

[24] Silvio Micali, Michael O. Rabin, and Salil P. Vadhan. Verifiable random functions. In *40th FOCS*, pages 120–130. IEEE Computer Society Press, October 1999.

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

[26] Timo Hanke, Mahnush Movahedi, and Dominic Williams. Dfinity technology overview series, consensus system. *arXiv preprint arXiv:1805.04548*, 2018.

[27] Maofan Yin, Dahlia Malkhi, Michael K. Reiter, Guy Golan-Gueta, and Ittai Abraham. HotStuff: BFT consensus with linearity and responsiveness. In Peter Robinson and Faith Ellen, editors, *38th ACM PODC*, pages 347–356. ACM, July / August 2019.

[28] Soubhik Deb, Sreeram Kannan, and David Tse. Posat: Proof-of-work availability and unpredictability, without the work. *FC 2021,*, 2020.

[29] Erica Blum, Aggelos Kiayias, Cristopher Moore, Saad Quader, and Alexander Russell. The combinatorics of the longest-chain rule: Linear consistency for proof-of-stake blockchains. In Shuchi Chawla, editor, *31st SODA*, pages 1135–1154. ACM-SIAM, January 2020.

[30] Aggelos Kiayias, Saad Quader, and Alexander Russell. Consistency of proof-of-stake blockchains with concurrent honest slot leaders. *ICDCS*, 2020.

[31] Amir Dembo, Sreeram Kannan, Ertem Nusret Tas, David Tse, Pramod Viswanath, Xuechao Wang, and Ofer Zeitouni. Everything is a race and nakamoto always wins. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 20*, pages 859–878. ACM Press, November 2020.

[32] Yevgeniy Dodis and Aleksandr Yampolskiy. A verifiable random function with short proofs and keys. In Serge Vaudenay, editor, *PKC 2005*, volume 3386 of *LNCS*, pages 416–431. Springer, Heidelberg, January 2005.

[33] Clément L Canonne. A short note on poisson tail bounds. 2017. http://www.cs.columbia.edu/~ccanonne/files/misc/2017-poissonconcentration.pdf.

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

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

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

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

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

[39] Kai-Min Chung, Henry Lam, Zhenming Liu, and Michael Mitzenmacher. Chernoff-hoeffding bounds for markov chains: Generalized and simplified. *arXiv preprint arXiv:1201.0559*, 2012.

Appendix A. Supplemental materials for Section 2

A.1. Predictability-Based Attacks

We now describe the attacks where the attackers rely on the power of predictability.

Predictable selfish mining attacks. In a selfish mining attack, a player chooses to not immediately publish the blocks they have generated to the rest of the network, which undermines the fairness of the blockchain. This type of attack is more prevalent in proof-of-stake protocols, as they allow players to predict their chances of successfully mining multiple blocks in the future. Brown-Cohen et al. [13] have demonstrated a predictable selfish mining attack in proof-of-stake protocols, where players predict a specific time period in which they will generate a certain number of blocks. If the probability that other players will not generate the same number of blocks during that time period is high enough, the player can choose to keep those blocks hidden until the last block is mined. This increases the likelihood that the player's blocks will be included in the longest chain.

To better illustrate the impact of unpredictability on selfish mining attacks, we will analyze specific examples below to examine the conditions for the attacker's success and their profits under different levels of unpredictability. For simplicity and better understanding, we adopt a simplified selfish mining strategy. In this strategy, the attacker only attempts to succeed in competing for a single block, without considering the competition for longer blockchains. After the attacker generates a new block B_1, they choose not to immediately publish it but instead attempt to generate another block independently. The following process can be divided into two cases:

Case 1: If other miners successfully generate a block B_1' first, the attacker immediately publishes their block B_1 to compete with B_1'.

Case 2: If the attacker successfully generates another block B_2 first, they wait until other miners generate and publish a block B_1', then simultaneously publish their two blocks$B_1 \| B_2$ to beat block B_1'.

We assume that the attacker holds a resource share proportion of β, then the probability of Case 1 occurring is $1-\beta$, and the probability of Case 2 occurring is β.

In Case 1, the competition occurs between two blocks of the same height. For simplification, we assume that the probability of either block winning the competition and being eventually retained is 1/2. Therefore, if Case 1 occurs, the expectation of the reward obtained by the attacker is 1/2 of a block reward. In Case 2, the attacker will receive the rewards for both blocks at these two heights, so the expected reward is 2 block rewards.

If the consensus protocol is best unpredictable, the expected reward for the attacker, holding a stake proportion of β, from generating a block and adopting the above selfish mining strategy is:

$$\beta(\frac{1}{2}(1-\beta)+2\beta) = \beta(\frac{3}{2}\beta+\frac{1}{2})$$

If $\beta > \frac{1}{3}$,the attacker can acquire rewards exceeding their proportion of stake β.

If the consensus protocol is 2-predictable, meaning the attacker can predict who will generate the next two blocks, they can avoid the occurrence of Case 1. Specifically, if they cannot generate two consecutive blocks, they will immediately release B_1 to secure the reward for 1 block. Therefore, the mathematical expectation of their reward is:

$$\beta((1-\beta)+2\beta) = \beta(1+\beta)$$

This means that regardless of the proportion β of stake held, the attacker can gain greater rewards through selfish mining.

Predictable bribing attacks. In bribery attacks, an attacker pays players to work on specific chains in order to benefit themselves, such as supporting double spending or censorship attacks. These attacks are more dangerous

in proof-of-stake protocols, as players can predict their chances of successfully mining blocks in the future. In epoch-based proof-of-stake protocols, this is particularly true at the beginning of each epoch, when an attacker can attempt to bribe players who are likely to mine new blocks. If the attacker is able to bribe enough players, they can control the majority of the blocks mined during that epoch. There are two cases to consider:

Case 1: *The confirmation time is shorter than the length of each epoch.* In this case, the attacker can perform a double spending attack by issuing transactions at the beginning of the epoch and then hiding their blocks. At the end of the epoch, these transactions will be confirmed on the best public chain. The attacker can then publish their hidden blocks and revert the transactions they issued at the beginning.

Case 2: *The confirmation time is longer than the length of each epoch.* The attacker can perform censorship attacks by preventing certain transactions from being included on the blockchain. In each epoch, the attacker can perform a predictable bribing attack to control a majority of the blocks, which means controlling the longest chain. Since all blocks on the longest chain belong to the attacker, they can prevent any transaction from being added to the blockchain.

A.2. The Definition of L-Unpredictability

Below is the formal definition of L-unpredictability.

Definition 8 (L-unpredictability)**.** *Consider a blockchain protocol Π. For $L \in \mathbb{N}$, we say a malicious player P is L-unpredictable at round r if for all* PPT $\mathcal{Z}$*, for all* PPT $\mathcal{A}$*, we have,*

$$\Pr\left[\begin{array}{l}\mathtt{VIEW} \leftarrow \mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}} \\ (r', z_P^{r'}) \leftarrow \mathcal{A}(P, r, \mathtt{VIEW}^r)\end{array} \middle| \mathsf{predictable}(\mathtt{VIEW}, P, L, r, r', z_P^{r'}) = 0\right] > 1 - \mathsf{negl}(\kappa),$$

where $\mathsf{negl}(\cdot)$ *is a negligible function.*

Appendix B. Supplemental materials for Section 3

B.1. Existing Single-Extension PoS Protocols

We now describe the existing state-of-the-art PoS protocols in [11], [10], [12] as single-extension PoS protocols.

Snow White [11]. The Snow White protocol [11] is divided into epochs, each consisting of $T_{epoch} = \Omega(\kappa)$ rounds. When players generate new blocks, they embed random seeds in those blocks. The random seeds are then used to determine which players will generate blocks in the next epoch. The four algorithms Validate, BestChain, Context, and Extend are constructed as follows:

- The algorithm Context takes as input a chain $\mathcal{C}$ at round r and outputs the context η as the context is the concatenation of the random seeds from multiple blocks in the previous epoch. Here, the function hash first truncates the blocks in the previous epoch and obtains the random seeds from those blocks. Then it concatenates all the random seeds to obtain the context. Note that, hash can be treated as a random oracle. The random seeds in those blocks are random; thus the probability that two random seeds are the same is negligible.
- The algorithm Extend takes as input a context η, a round r, and a public key PK. The algorithm returns a new block if the hash value of the context, the round number, and the public key are smaller than a given threshold.
- The algorithm Validate takes as input a chain $\mathcal{C}$, a round number r, and outputs 1 if each block in the chain $\mathcal{C}$ satisfies the following: 1) the context is correctly computed, 2) the hash inequality in the blocks holds, and 3) the round number of the block is smaller than the current round r.
- The algorithm BestChain takes as input a set of chains $\mathbb{C}$ and a round r. It outputs the longest valid chain in $\mathbb{C}$.

Ouroboros Praos [10]. The Ouroboros Praos protocol is constructed using a *Verifiable Random Function* [32] (VRF). The VRF generates a pseudorandom number with a proof of its correctness. The VRF is specified by three algorithms $(\mathsf{Gen}, \mathsf{Prove}, \mathsf{Ver})$. The algorithm Gen takes the security parameter κ as input and outputs a key pair $(\mathrm{SK}, \mathrm{PK})$. The algorithm Prove takes the secret key SK and a message msg as input and returns a pseudorandom output σ along with a proof π. We write $(\sigma, \pi) := \mathsf{Prove}_{\mathrm{SK}}(msg)$. The algorithm Ver takes a public key PK, a message msg, an output σ, and a proof π as input and returns 1 if the output and the proof are correct. Similar to the Snow White protocol [11], the Ouroboros Praos protocol [10] separates rounds into epochs; each epoch has $T_{epoch} = \Omega(\kappa)$ rounds. The four algorithms Validate, BestChain, Context, and Extend is constructed as follows:

- The algorithm Context takes as input a chain $\mathcal{C}$ at round r and outputs the context η as the hash value of the VRF output in the blocks in $\mathcal{C}$ that are generated in the previous epochs. Here, the function hash can be treated as a random oracle.
- The algorithm Extend takes as input a context η, a round r, and a secret key SK. The algorithm computes $(\sigma, \pi) := \mathsf{Prove}_{\mathrm{SK}}(\eta, r)$ and returns a new block if it holds that $\sigma < \mathtt{T}$, where T is the difficulty.
- The algorithm Validate takes as input a chain $\mathcal{C}$ and a round number r, and outputs 1 if each block in the chain $\mathcal{C}$ satisfies the following conditions: 1) the context is computed correctly, 2) the VRF output in the block is computed correctly using algorithm $\mathsf{Ver}_{(\cdot)}(\cdot)$, 3) the output of the VRF is smaller than the difficulty T, and 4) the round number in the block is smaller than the current round r.
- The algorithm BestChain takes as input a set of chains $\mathbb{C}$ and a round r. It outputs the longest valid chain in $\mathbb{C}$.

Bagaria et al. [12]. The protocol described in Bagaria et al. [12] is divided into epochs, each of which consists of $c \in \mathbb{N}$ blocks. The protocol uses a VRF to determine which players can generate new blocks. The following four algorithms are defined: Validate, BestChain, Context, and Extend.

- The algorithm Context takes as input a chain $\mathcal{C}$ at round r and outputs the context η as the VRF output of the last block from the previous epoch. The probability that two blocks have the same VRF output equals the probability of selecting two random numbers in $\{0,1\}^\kappa$ that are equal. As the number of blocks is polynomial in κ, the probability that there exist two blocks that have the same VRF output is negligible.
- The algorithm Extend takes as input a context η, a round r, and a secret key SK. The algorithm computes $(\sigma, \pi) := \mathsf{Prove}_{\mathrm{SK}}(\eta, r)$ and returns a new block if it holds that $\sigma < \mathtt{T}$, where T is the difficulty.
- The algorithm Validate takes as input a chain $\mathcal{C}$ and a round number r, and outputs 1 if each block in the chain $\mathcal{C}$ satisfies the following conditions: 1) the context is computed correctly, 2) the VRF output in the block is computed correctly using algorithm $\mathsf{Ver}_{(\cdot)}(\cdot)$, 3) the output of the VRF is smaller than the difficulty T, and 4) the round number in the block is smaller than the current round r.
- The algorithm BestChain takes as input a set of chains $\mathbb{C}$ and a round r. It outputs the longest valid chain in $\mathbb{C}$.

B.2. Proof of Theorem 1

Here, we provide the complete proof of Theorem 1. First, we present the formal definition of the *distinct-context-extension* property. Then, we show in Lemma 1 that if the single-extension PoS protocol Π achieves the best possible unpredictability, it must achieve *distinct-context-extension* property. Finally, we show in Lemma 2 that if the single-extension PoS protocol Π achieves distinct-context-extension property, it cannot achieve common prefix property if $\alpha < 2.72\beta$.

B.2.1. Distinct-context-extension. The *distinct-context-extension* property states that the contexts of any chains in the execution must be distinct in order to achieve the best possible unpredictability. This prevents the adversary from predicting the extension of a chain in the future based on an existing chain in the past.

Additionally, we define shared-context-extension, which is the counterpart of distinct-context-extension. We say two chains in the execution are shared-context-extension if the contexts of the two chains are the same. The shared-context-extension allows the adversary to predict whether or not it can extend a future chain that has not yet been generated. Specifically, if the future chain shares the same context as the current chain, the adversary can base its prediction (for the future chain) on the extension of the current chain. We note that existing protocols, such as Ouroboros Praos [10] and SnowWhite [11], have the shared-context-extension property, while our protocol in Section 4 achieves the distinct-context-extension property.

We now present the definition of distinct-context-extension and shared-context-extension for two chains. (Toy examples of distinct-context-extension and shared-context-extension can be seen in Fig. 9 and Fig. 10, respectively.)

Definition 9 (Distinct and shared-context-extension for two chains)**.** *Consider a single-extension proof-of-stake protocol* Π *that is parameterized by four algorithms:* Validate, BestChain, Context, *and* Extend. *Let* $\mathcal{P}$ *be the set of players. For any adversary* $\mathcal{A}$ *and environment* $\mathcal{Z}$*, consider some* VIEW *in the support of* $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$*. Consider two chains* $\mathcal{C}_1$ *and* $\mathcal{C}_2$ *in* VIEW*.*

- *We say the extensions of* $\mathcal{C}_1$ *and* $\mathcal{C}_2$ *are* distinct-context-extension *if the contexts of* $\mathcal{C}_1$ *and* $\mathcal{C}_2$ *are distinct, i.e.* $\mathsf{Context}(\mathcal{C}_1) \neq \mathsf{Context}(\mathcal{C}_2)$*. We write* $\mathsf{distinct\text{-}context\text{-}extension}(\mathcal{C}_1, \mathcal{C}_2) = 1$*. In this case, the event that the adversary* $\mathcal{A}$ *can extend the chain* $\mathcal{C}_1$ *and the event that* $\mathcal{A}$ *can extend the chain* $\mathcal{C}_2$ *are independent.*
- *We also consider the flip side of distinct-context-extension, i.e. shared-context-extension. We say the extensions of* $\mathcal{C}_1$ *and* $\mathcal{C}_2$ *are* shared-context-extension *if the contexts of* $\mathcal{C}_1$ *and* $\mathcal{C}_2$ *are the same, i.e.,* $\mathsf{Context}(\mathcal{C}_1) = \mathsf{Context}(\mathcal{C}_2)$*. We write* $\mathsf{shared\text{-}context\text{-}extension}(\mathcal{C}_1, \mathcal{C}_2) = 1$*. In this case, if the adversary* $\mathcal{A}$ *can extend the chain* $\mathcal{C}_1$*, it can also extend the chain* $\mathcal{C}_2$*, and vice versa.*

We are now ready to define the distinct-context-extension property for a PoS protocol. Intuitively, we say that a protocol achieves the distinct-context-extension property if all different chains in the protocol execution have distinct contexts.

Definition 10 (Distinct-context-extension for all chains in a protocol execution)**.** *Consider a single-extension proof-of-stake protocol* Π *that is parameterized by four algorithms:* Validate, BestChain, Context, *and* Extend. *Let* $\mathcal{P}$ *be the set of players. For any adversary* $\mathcal{A}$ *and environment* $\mathcal{Z}$*, consider some* VIEW *in the support of* $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$*. Consider some round* r*; let* $\mathbb{C}^r$ *be the set of all chains that appear in the view of some players (or the adversary) in* $\mathtt{VIEW}^r$*. Here,* $\mathtt{VIEW}^r$ *is the prefix of* VIEW *up until round* r*. We overload the predicate* distinct-context-extension *for a view* VIEW*. Intuitively, a view* VIEW *is distinct-context-extension if all different chains in* VIEW *have distinct contexts. More concretely, we say a view* VIEW *is distinct-context-extension if and only if for any round* r*, for any chains* $\mathcal{C}_1, \mathcal{C}_2 \in \mathbb{C}^r$ *such that* $\mathcal{C}_1 \neq \mathcal{C}_2$*, we have,* $\mathsf{distinct\text{-}context\text{-}extension}(\mathcal{C}_1, \mathcal{C}_2) = 1$*. We write* $\mathsf{distinct\text{-}context\text{-}extension}(\mathtt{VIEW}) = 1$*. We say protocol* Π *achieves* distinct-context-extension property *if for every* PPT $\mathcal{Z}, \mathcal{A}$*, we have,*

$$\Pr[\mathtt{VIEW} \leftarrow \mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}} \mid \mathsf{distinct\text{-}context\text{-}extension}(\mathtt{VIEW}) = 1] = 1 - \mathsf{negl}(\kappa),$$

where $\mathsf{negl}(\cdot)$ *is a negligible function.*

B.2.2. Achieving the best possible unpredictability via distinct-context-extension. We prove that a single-extension PoS protocol can only achieve the best possible unpredictability if it satisfies the distinct-context-extension property. Intuitively, consider two chains $\mathcal{C}_1$ and $\mathcal{C}_2$ that share a context extension. Without loss of generality, we assume that the length of $\mathcal{C}_1$ is greater than the length of $\mathcal{C}_2$. We define $\mathcal{C}$ as the longest common prefix of $\mathcal{C}_1$ and $\mathcal{C}_2$. Recall that, the contexts of $\mathcal{C}_1$ and $\mathcal{C}_2$ are computed using a hash function (which will be treated as a random oracle) over some blocks on the two chains. Since the contexts of $\mathcal{C}_1$ and $\mathcal{C}_2$ are the same, the shared context must be extracted from the longest common prefix $\mathcal{C}$ of $\mathcal{C}_1$ and $\mathcal{C}_2$. Upon receiving the chain $\mathcal{C}$ at round r, player P can predict whether or not he can extend $\mathcal{C}_1$. As the

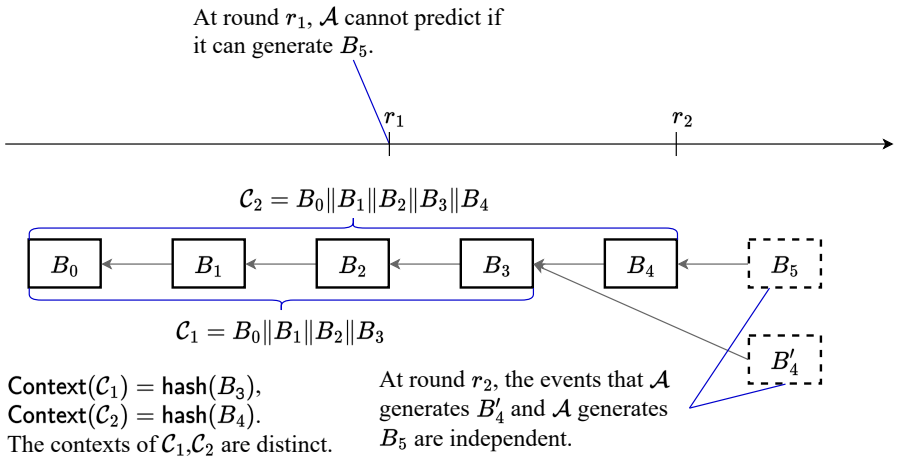

Figure 9. A toy example of distinct-context-extension for two chains. Consider two chains $\mathcal{C}_1 = B_0\|B_1\|B_2\|B_3$ and $\mathcal{C}_2 = B_0\|B_1\|B_2\|B_3\|B_4$. Here, $\mathsf{Context}(\mathcal{C}_1) = \mathsf{hash}(B_3)$ and $\mathsf{Context}(\mathcal{C}_2) = \mathsf{hash}(B_4)$. As $B_3 \neq B_4$, we have, $\mathsf{Context}(\mathcal{C}_1) \neq \mathsf{Context}(\mathcal{C}_2)$. In other words, $\mathcal{C}_1$ and $\mathcal{C}_2$ are distinct-context-extension. At round r_2, the event that the adversary $\mathcal{A}$ can extend the chain $\mathcal{C}_1$ to generate a new block B'_4 and the event that $\mathcal{A}$ can extend the chain $\mathcal{C}_2$ to generate a new block B_5 are independent. At round r_1, the adversary $\mathcal{A}$ has not yet received the chain $\mathcal{C}_2$. Therefore, it cannot predict whether it can extend $\mathcal{C}_2$ to generate block B_5.

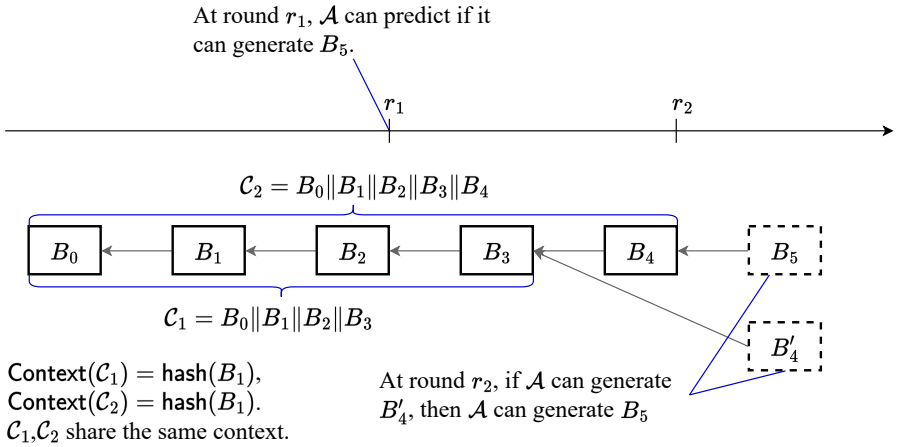

Figure 10. A toy example of shared-context-extension for two chains. The definition of function Context here is very different from that in Fig. 9. Consider two chains $\mathcal{C}_1 = B_0\|B_1\|B_2\|B_3$ and $\mathcal{C}_2 = B_0\|B_1\|B_2\|B_3\|B_4$. We stress that, here, $\mathsf{Context}(\mathcal{C}_1) = \mathsf{hash}(B_1)$ and $\mathsf{Context}(\mathcal{C}_2) = \mathsf{hash}(B_1)$. While in Fig. 9, $\mathsf{Context}(\mathcal{C}_1) = \mathsf{hash}(B_3)$ and $\mathsf{Context}(\mathcal{C}_2) = \mathsf{hash}(B_4)$. We have, $\mathsf{Context}(\mathcal{C}_1) = \mathsf{Context}(\mathcal{C}_2)$. In other words, $\mathcal{C}_1$ and $\mathcal{C}_2$ are shared-context-extension. At round r_2, if the adversary $\mathcal{A}$ can extend the chain $\mathcal{C}_1$ to generate a new block B'_4, it can also extend the chain $\mathcal{C}_2$ to generate a new block B_5. Thus, at round r_1, when the adversary $\mathcal{A}$ has received $\mathcal{C}_1$ but not yet received $\mathcal{C}_2$, the adversary can predict whether or not it can extend $\mathcal{C}_2$ to generate block B_5 at round r_2.

length of $\mathcal{C}_1$ is greater than the length of $\mathcal{C}$, player P is 2-predictable at round r, which contradicts the best possible unpredictability.

Lemma 1. *Consider a single-extension proof-of-stake protocol* Π *that is parameterized by four algorithms:* Validate, BestChain, Context, *and* Extend. *If the protocol* Π *achieves the best possible unpredictability, then it achieves distinct-context-extension property.*

The proof of Lemma 1 can be found in the full version of our paper.

We note that the single-extension protocols in [11], [10] cannot achieve the best possible unpredictability. The execution of these protocols is divided into epochs, each consisting of $O(\kappa)$ blocks. In these protocols, the context is computed based on the hash values of the blocks in the previous epoch. As a result, all chains in the same epoch share the same context. Thus, at the beginning of each epoch, malicious players can predict whether or not they can extend their chains in the current epoch. Bagaria et al. [12] proposed a single-extension protocol with a constant-sized epoch. In this protocol, the context of a chain is computed as the hash value of the last block in the previous epoch. If each epoch consists of at least two blocks, the protocol cannot achieve the best possible unpredictability. This is because all the chains in the same epoch share the same context. On the other hand, if each epoch consists of only one block, the protocol can achieve the best possible unpredictability. Now,

the protocol achieves distinct-context-extension property. Jumping ahead, in Section B.2.3, we will show that the protocol requires 73% of honest stake to achieve the security properties.

B.2.3. Breaking the common prefix property via distinct-context-extension. We show that if a single-extension proof-of-stake (PoS) protocol achieves the distinct-context-extension property, the adversary can violate the common prefix property if the honest players control less than 73% of the stake. More specifically, based on the distinct-context-extension property, the adversary can amplify their stake by at least a factor of $e = 2.72$. Therefore, if the honest players control less than 73% of the stake, the adversary can extend chains faster than the honest players and thus break the common prefix property of the protocol.

The chain growth of the adversary. We establish a bound on the chain growth of the adversary as follows. We demonstrate that for any valid chain $\mathcal{C}$ at any round r, there exists an adversary who can extend the chain C with a probability of at least β. Given that the protocol achieves the distinct-context-extension property, the extensions of all chains are independent. We model the chain extension of the adversary as a random tree, where each branch of the tree represents a chain in the block tree and the extensions of the branches are modeled as independent random variables.

Claim 1 (Adversarial extension)**.** *Consider any proof-of-stake protocol Π with a set of player P. For some adversary $\mathcal{A}$ and environment $\mathcal{Z}$, consider some* VIEW *in the support of* $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$*. Let $\mathbb{C}^r$ be the set of chains in the view of all players at round r. At round r, the adversary attempts to extend a chain $\mathcal{C} \in \mathbb{C}^r$. Specifically, the adversary $\mathcal{A}$, takes inputs as a chain $\mathcal{C} \in \mathbb{C}^r$, the round number r, and output a block B. For every,* PPT $\mathcal{Z}$*, there exists an adversary $\mathcal{A}$ such that, at any round r, we have,*

$$\Pr\left[\begin{array}{l|l} \mathtt{VIEW} \leftarrow \mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}; & \\ \mathcal{C} \leftarrow \mathbb{C}^r; & \mathsf{Validate}(\mathcal{C}\|B, r) = 1 \\ B \leftarrow \mathcal{A}(\mathcal{C}, r); & \end{array}\right] \geq \beta.$$

The proof of Claim 1 is detailed in the full version.

Next, we show that the adversary can amplify its stake by a factor of $e = 2.72$. To do this, we consider an adversary that extends all chains. Based on Lemma 1, the probability of the adversary extending a chain in each round is at least β. Additionally, as stated in Lemma 1, to achieve the best possible unpredictability, the extensions of all chains must be distinct-context-extension. Thus, the probabilities of the adversary extending the chains are independent. The chain extension of the adversary is modeled as a random tree with independent extensions in each branch. To bound the growth rate of the chain, we first bound the number of branches in the random tree, and then, based on the number of branches and the growth rate of each branch, we can determine the maximum length of all branches in the random tree.

Before presenting the detailed proofs, we introduce a useful inequality as follows.

Claim 2 (Theorem 1 in [33])**.** *Consider a Poisson random variable X that has the expected value of λ. We have the following inequalities.*

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

Claim 3 (Theorem 3 in [34])**.** *Let $X_1, X_2, \cdots, X_t$ be identical independent random variables in range $[0,1]$ with an expected value of λ. Then, for any $\epsilon > 0$, we have* $\Pr[\sum_{i=1}^{t} X_i < (1-\epsilon) \cdot t \cdot \lambda] \leq e^{-\Omega(t)}$.

We describe the adversary's chain extension as a branching process, as follows. Let Z_t be the set of all branches at round t, where $t \in \mathbb{N}$, and G_t be the number of branches in Z_t. At the beginning, there is only one branch of length 0, i.e., $Z_0 = \{0\}$ and $G_0 = 1$. Let X be a Poisson random variable with an expected value of β. Let $X_{t,i}$ be the random variable that represents the random process in the i-th branch in Z_t. Here, $X_{t,i}$ are independent and identically distributed random variables of X. Let $\ell_{t,i}$ be the length of the i-th branch in Z_t. We will add $X_{t,i}+1$ branches with the length $\ell_{t,i}, \ell_{t,i}+1, \cdots, \ell_{t,i}+X_{t,i}$ into Z_{t+1}. We denote T_t as the maximum length of all branches in Z_t, i.e., $T_t = \max_{i \in \{1,2,\cdots,G_t\}} \ell_{t,i}$. The maximum length T_t is equivalent to the length of the longest chain. To bound the adversary's chain growth, we first bound the number of different branches at the end of the process (see Lemma 4). Then, we use the union bound for the maximum length of all branches.

Claim 4. *Consider the set of branches Z_t at time t. For any $\epsilon'' > 0$, we have* $\Pr[G_t < (\beta+1)^{(1-\epsilon'')\cdot t}] < e^{-\Omega(t)}$.

Claim 5. *Consider a single-extension proof-of-stake protocol Π satisfies distinct-context-extension property. For some adversary $\mathcal{A}$ and environment $\mathcal{Z}$, consider some* VIEW *in the support of* $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$*. Let $\mathbb{C}^{r_1}$ be the set of chains at round r_1. The adversary takes inputs as a chain $\mathcal{C}_1 \in \mathbb{C}^{r_1}$ and the round numbers r_1, r_2 and outputs a chain $\mathcal{C}_2$. For every,* PPT $\mathcal{Z}$*, there exists an adversary $\mathcal{A}$ such that, at any round r_1, r_2, where $r_2 - r_1 = t$ and $t = \Omega(\kappa)$, we have,*

$$\Pr\left[\begin{array}{l|l} \mathtt{VIEW} \leftarrow \mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}};\ \mathcal{C}_1 \leftarrow \mathbb{C}^{r_1}; & (\mathsf{Validate}(\mathcal{C}_2, r_2) = 1) \\ \mathcal{C}_2 \leftarrow \mathcal{A}(\mathcal{C}_1, r_1, r_2); & \wedge\ (\mathcal{C}_1 \preceq \mathcal{C}_2) \\ \ell_1 := \mathsf{len}(\mathcal{C}_1); \ell_2 := \mathsf{len}(\mathcal{C}_2); & \wedge\ (\ell_2 - \ell_1 > (1-\epsilon)\cdot e \cdot \beta \cdot t) \end{array}\right]$$

$$\geq 1 - e^{-\Omega(\kappa)},$$

where $e = 2.72$.

Breaking common prefix. Since the adversary can amplify its stake by a factor $e = 2.72$, the adversary can extend the chain faster than the honest players it controls more than 27% of stake. Thus, the adversary can keep its blocks hidden and then publish those blocks when the length of the hidden chain is bigger than κ. Now, the hidden chain will become the new best chain and it does not share a common prefix with the previous best chain.

Lemma 2. *Assume $\alpha < e \cdot \beta$, where $e = 2.72$. Consider a single-extension proof-of-stake protocol Π that is parameterized by four algorithms:* Validate*,* BestChain*,* Context*, and* Extend*. If protocol Π achieves distinct-context-extension property, then it cannot achieve common prefix property.*

Appendix C.
Supplemental materials for Section 4

C.1. Multi-Extension Proof-of-Stake Protocols

We say a PoS protocol is a multi-extension protocol if, in each round, each honest player is allowed to extend multiple chains. By extending multiple chains, honest players can extend the best chain faster, compared to the single-extension protocol.

Definition 11 (Multi-extension framework for PoS protocols)**.** *A multi-extension PoS protocol* Π° *is parameterized by 4 deterministic algorithms* $(\mathsf{Validate}^{\circ}, \mathsf{BestChainSet}^{\circ}, \mathsf{Context}^{\circ}, \mathsf{Extend}^{\circ})$ *as follows:*

- *The validation algorithm* $\mathsf{Validate}^{\circ}$ *takes a chain* $\mathcal{C}$ *and a round* r *as input and returns* 1 *if the chain* $\mathcal{C}$ *is valid at round* r*, and returns* 0 *otherwise.*
- *The context extraction algorithm* $\mathsf{Context}^{\circ}$ *takes a valid chain* $\mathcal{C}$ *as input and returns a context* η*. If the input chain is invalid, the algorithm returns* $\perp$*.*
- *The extension algorithm* Extend° *is parameterized by a probability* $p \in (0,1)$*. The algorithm takes input as a context* η*, a round* r*, and a secret key* SK*, and returns a new block* B *or* $\perp$ *(if no new block is generated). Here, the secret key* SK *is generated by a player* P *in the blockchain initialization phase, and the corresponding public key of* SK *will be stored in the genesis block. The function* $\mathsf{Extend}^{\circ}(\eta, r, \mathrm{SK})$ *returns a block* B *with probability p.*
- *The best chain set algorithm* $\mathsf{BestChainSet}^{\circ}$ *takes a set of chains* $\mathbb{C}$ *and returns a set of the best chains* $\mathbb{C}_{\mathrm{best}}$*. Here, the honest players will* ***extend multiple chains****, i.e., all the chains in the set of the best chains* $\mathbb{C}_{\mathrm{best}}$*. Thus, we name the protocol* ***multi-extension****.*

We note that the descriptions of algorithms $\mathsf{Validate}^{\circ}$*,* $\mathsf{Context}^{\circ}$*, and* Extend° *are identical to the descriptions of algorithms* Validate*,* Context*, and* Extend *in the single-extension PoS protocol, defined in Definition 5. The difference between the single-extension and multi-extension protocols is the utilization of different algorithms, namely* BestChain *and* $\mathsf{BestChainSet}^{\circ}$*, for determining the chains to be extended. The single-extension protocol uses the best chain algorithm* BestChain *to select a single best chain. Meanwhile, the multi-extension protocol employs the best chain set algorithm* $\mathsf{BestChainSet}^{\circ}$*, which returns a set of multiple best chains. The advantage of extending multiple chains is that honest players can extend the best chain faster compared to the single-extension protocol.*

Blockchain initialization phase. *In this phase, the genesis block will be created; the genesis block consists of a randomness, public information, and the stake distribution of the players. Consider an (initial) group of PoS-players* $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ *and a security parameter* κ*. Each player* $P_j \in \mathcal{P}$ *generates a pair of public key* PK_j *and private key* SK_j*. The public keys of all players are stored in the genesis block of the blockchain system. We let* B_0 *denote the genesis block. .*

Blockchain extension phase. *A multi-extension proof-of-stake protocol* Π *is described in Algorithm 4. In each round* r*, a player* P *with the secret key* SK *proceeds as follows. First, the player* P *set* $\mathbb{C}_{\mathrm{best}} := \mathsf{BestChainSet}^{\circ}(\mathbb{C})$*.*

Algorithm 4: A multi-extension PoS protocol Π°.

State : Initially, the set of chains $\mathbb{C}$ only consists of the genesis block. At round r, the PoS-player $P \in \mathcal{P}$, with key pair $(\mathrm{SK}, \mathrm{PK})$ and local chain set $\mathbb{C}$, proceeds as follows.

Upon receiving a chain $\mathcal{C}'$, verify $\mathsf{Validate}^{\circ}(\mathcal{C}', r) = 1$ and set $\mathbb{C} := \mathbb{C} \cup \{\mathcal{C}'\}$;

Set $\mathbb{C}_{\mathrm{best}} := \mathsf{BestChainSet}^{\circ}(\mathbb{C})$;

for $\mathcal{C} \in \mathbb{C}_{\mathrm{best}}$ **do**

  $\eta := \mathsf{Context}^{\circ}(\mathcal{C})$; $B := \mathsf{Extend}^{\circ}(\eta, r, \mathrm{SK})$;

  if $B \neq \perp$ **then**

    $\mathcal{C}' := \mathcal{C} \| B$; Add $\mathcal{C}'$ to $\mathbb{C}$; Broadcast $\mathcal{C}'$;

Here the local set of chains $\mathbb{C}$ *consists of all valid chains that are received (or generated) by* P*. Then, for each chain* $\mathcal{C} \in \mathbb{C}_{\mathrm{best}}$*, the player* P *uses the function* $\mathsf{Context}^{\circ}$ *to extract the context* η *in the best chain* $\mathcal{C}$*, i.e.,* $\eta := \mathsf{Context}^{\circ}(\mathcal{C}, r)$*. Finally, based on the context* η*, the current round number* r*, and the secret key* SK*, the player* P *uses the function* Extend° *to determine whether or not it can generate a new block. If the player* P *can generate a new block* B*, it creates a new chain* $\mathcal{C}' := \mathcal{C} \| B$*, adds* $\mathcal{C}'$ *to the set of chains* $\mathbb{C}$ *and broadcasts* $\mathcal{C}'$ *to all other players.*

Remark 2. *We remark that there are other ways to design multi-extension protocols. However, to simplify the presentation, we focus on the design in Definition 11. We used this design of multi-extension protocols for our protocol in Section 4.*

C.2. Unique Signature Scheme

In a unique signature scheme, for every possible verification key, and every message to be signed, there is a unique signature. Please see Section 6.5.1 of Goldreich's textbook [35] 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 12. *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

$$\Pr\left[\ (\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{uKeyGen}(1^{\kappa}) \mid \mathsf{uKeyVer}(\mathrm{PK}, \mathrm{SK}) = 1\ \right] = 1.$$

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

$$\Pr\left[\ (\mathrm{PK}, \mathrm{SK}_1, \mathrm{SK}_2) \leftarrow \mathcal{A}(1^{\kappa}) \ \middle|\ \begin{array}{l}(\mathsf{uKeyVer}(\mathrm{PK}, \mathrm{SK}_1) = 1) \\ \bigwedge\ (\mathsf{uKeyVer}(\mathrm{PK}, \mathrm{SK}_2) = 1) \\ \bigwedge\ (\mathrm{SK}_1 \neq \mathrm{SK}_2)\end{array}\ \right] \leq \mathsf{negl}(\kappa).$$

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

$$\Pr\left[\begin{array}{l|l}(\mathrm{PK},\mathrm{SK})\leftarrow \mathsf{uKeyGen}(1^\kappa) & \\ \sigma := \mathsf{uSign}(\mathrm{SK},x) & (\mathsf{uVerify}(\mathrm{PK},x,\sigma)=1)\end{array}\right] \geq 1-\mathsf{negl}(\kappa).$$

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

$$\Pr\left[\begin{array}{l|l}(\mathrm{PK},x,\sigma_1,\sigma_2)\leftarrow \mathcal{A}(1^\kappa) & \begin{array}{l}(\mathsf{uVerify}(\mathrm{PK},x,\sigma_1)=1)\\ \bigwedge (\mathsf{uVerify}(\mathrm{PK},x,\sigma_2)=1)\\ \bigwedge (\sigma_1\neq\sigma_2)\end{array}\end{array}\right] \leq \mathsf{negl}(\kappa).$$

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

$$\Pr\left[\begin{array}{l|l}(\mathrm{PK},\mathrm{SK})\leftarrow \mathsf{uKeyGen}(1^\kappa); & (\mathsf{uVerify}(\mathrm{PK},x,\sigma)=1)\\ (x,\sigma)\leftarrow \mathcal{A}^{\mathsf{uSign}(\mathrm{SK},\cdot)}(\mathrm{PK}) & \bigwedge ((x,\sigma)\notin Q)\end{array}\right], \leq \mathsf{negl}(\kappa),$$

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

Unique signature schemes and related notions have been investigated in literature (e.g., [36], [24], [37]). Please see Section 6.5.1 of Goldreich's textbook [35] for detailed discussions about the constructions. Several efficient constructions can be found in the literature. For example, the well-known BLS signature [38] can be a good candidate.

Appendix D. Supplemental materials for Section 5.1

We present a framework for analyzing the chain growth property in multi-extension protocols using the Markov chain. We develop a random walk in a Markov chain. The Markov chain consists of multiple states, where each state provides some information on the set of best chains in a protocol round. Then, we will apply this framework to analyze the chain growth property of our protocol $\Pi^\bullet$.

We construct a *Markov chain* with a *state space* S and a *transition matrix* $\mathbf{T}$. Each state $s \in S$ provides some information on the view of the players in a protocol round. The transition matrix $\mathbf{T}$ is a $|S| \times |S|$ matrix that reflects how the set of best chains is updated after one protocol round. We define a *chain growth function* $\mathsf{growth} : S \to [0,1]$ to represent the chain growth rate on each state. Then, we consider a *random walk* of t states $s_1, s_2, \cdots, s_t$, where $t \in \mathbb{N}$ and for all $i \in [t], s_i \in S$. This random walk represents how the set of best chains is updated in t protocol rounds. The chain growth is computed as the sum of outputs of the chain growth function over all the states in the random walk. (See Supplemental material D.1 for more details.)

We remark that, in a single-extension protocol, the probability of honest players extending the best chain remains constant in every round. As a result, the chain growth function can be simplified to a constant value, making the analysis of chain growth property easier, compared to the analysis of a multi-extension protocol.

D.1. Defining a Markov Chain

Before presenting our analysis for the chain growth property, let us summarize the definition of a Markov chain and the random walk on a Markov chain [39]. We will use the Markov chain to analyze the chain growth property of our protocol.

Markov chain. A *Markov chain* is a mathematical model that describes a sequence of events in which the probability of each event depends only on the state preceding it. The defining characteristic of a Markov chain is that no matter how the process arrived at its present state, the possible future states are fixed. In other words, the probability of transitioning to any particular state is dependent solely on the current state. A Markov chain is specified by a *state space* S and a *transition matrix* $\mathbf{T}$. For simplicity, we will refer to the Markov chain as $(S, \mathbf{T})$. Each state $s \in S$ is a tuple of d integers $\langle n_0, \cdots, n_{d-1}\rangle$, where $d \in \mathbb{N}$ and for all $i \in [0..d-1], n_i \in \mathbb{N}$. In our analysis, each state provides some information on the view of the players in a protocol round. For example, we can define each state $s \in S$ in the format of $\langle n_0\rangle$, where $n_0 \in \mathbb{N}$. Here, n_0 is the number of chains that have the same length as the current best chain. The state space is given as $S = \{\langle n_0\rangle\}_{n_0\in\mathbb{N}}$. The transition matrix $\mathbf{T}$ is an $|S|\times|S|$ matrix that contains information on the probability of transitioning between states, where $|S|$ is the size of the state space S. For any two states s and s', the probability of transitioning from state s to state s' is $\mathbf{T}_{s,s'}$.

Random walk. A *random walk* in the Markov chain is a sequence of t states $s_1, s_2, \cdots, s_t$, where $t \in \mathbb{N}$ and for all $i \in [t]$, $s_i \in S$. The random walk starts at some state s_1, traverses to a new state s_2, based on the transition matrix $\mathbf{T}$, and then repeats the process.

Stationary distribution. A *stationary distribution* $Q = [q_s]_{s\in S}$ of a Markov chain is a probability distribution that represents the probabilities that states appear in a random walk. Here, the probability that a state s is drawn from the stationary distribution Q is q_s. If the state s is randomly drawn from the stationary distribution Q, we write $s \sim Q$. The sum of the probabilities in Q equals 1, i.e., $\sum_{s\in S} q_s = 1$. For every state $s \in S$, the probability q_s is computed based on the transitions from other states to the state s, i.e., $q_s = \sum_{s'\in S} q_{s'} \cdot \mathbf{T}_{s',s}$. Hence, we can obtain the stationary distribution by solving the following equations.

$$\begin{cases}\sum_{s\in S} q_s = 1,\\ q_s = \sum_{s'\in S}\left(q_{s'}\cdot \mathbf{T}_{s',s}\right), \forall s \in S.\end{cases} \tag{1}$$

Chain growth function. We define a *chain growth function* $\mathsf{growth} : S \to [0,1]$ to represent the chain growth rate on each state. For example, if each state $s \in S$ is in the format of $\langle n\rangle$, where n is the number of chains that have the same length as the current best chains, we have, $\mathsf{growth}(n) = n \cdot \alpha$. Consider a random walk $s_1, s_2, \cdots, s_t$. The chain growth, i.e., the increasing length of the best chain, in those t protocol rounds is computed as $\sum_{i=1}^{t} \mathsf{growth}(s_i)$. As the stationary distribution Q represents the probabilities that states appearing in a random

walk, the expected chain growth in a protocol round is given by

$$\bar{g} = \mathbb{E}_{s\sim Q}[\mathsf{growth}(s)].$$

Compatible Markov chain and chain growth function for a protocol execution. To analyze the chain growth of a multi-extension protocol, we need to design a *compatible* Markov chain and chain growth function. We say the Markov chain $(S, \mathbf{T})$ and the chain growth function growth is *compatible* to the execution of protocol Π° if for every state $s \in S$ that represents the view of the players at round r, the probability that the honest players generate a new best chain is $\mathsf{growth}(s)$.

Definition 13 (A compatible Markov chain and chain growth function for a protocol execution)**.** *Consider a multi-extension proof-of-stake protocol Π°. Consider a Markov chain $(S, \mathbf{T})$ and a chain growth function* $\mathsf{growth} : S \to [0, 1]$. *For a protocol round r, the view $\mathtt{VIEW}^r$ of players at round r can be mapped to a state s in the state space S. We say the Markov chain $(S, \mathbf{T})$ and the chain growth function* $\mathsf{growth} : S \to [0, 1]$ *are compatible to the execution of protocol Π° if for every round r with the view $\mathtt{VIEW}^r$, which is represented by a state $s \in S$, we have the probability that the length of the best chain increases by 1 at round r is* $\mathsf{growth}(s)$.

D.2. Chain Growth Property for a Multi-Extension Protocol

Consider a multi-extension proof-of-stake protocol Π°. Consider a Markov chain $(S, \mathbf{T})$ and a chain growth function $\mathsf{growth} : S \to [0, 1]$ that are compatible to the execution of the protocol Π°. We can bound the chain growth, i.e., the increasing length of the best chain, in a multi-extension proof-of-stake protocol Π° using the compatible Markov chain and chain growth function as follows.

Lemma 3 (Chain growth property for a multi-extension protocol)**.** *Consider a Markov chain $(S, \mathbf{T})$ and a chain growth function* $\mathsf{growth} : S \to [0, 1]$ *that are compatible to a multi-extension protocol Π°. Let Q be the stationary distribution over S, and $\bar{g} = \mathbb{E}_{s\sim Q}[\mathsf{growth}(s)]$ be the expected chain growth. Consider an honest player P with the best chain $\mathcal{C}$ in round r, and an honest player P_1 with the best chain $\mathcal{C}_1$ in round r_1, where $r_1 = r + t$, for some $t = \Omega(\kappa)$. Then we have $\Pr\left[\mathsf{len}(\mathcal{C}_1) - \mathsf{len}(\mathcal{C}) \geq (1-\delta)\cdot\bar{g}\cdot t\right] \geq 1 - e^{-\Omega(\kappa)}$ where $t = r_1 - r$, and $\delta > 0$.*

By designing the compatible Markov chain, and chain growth function, we can use the Chernoff bound to analyze the chain growth over a sufficiently long period. This approach provides a general and flexible way to study the chain growth of multi-extension protocols.

In the full version of our paper, we use the above analysis framework to examine the chain growth of our protocol.

Appendix E. Supplemental materials for Section 5.2

We present a new analysis framework for examining the common prefix property in multi-extension protocols. We introduce the concepts of *virtual block-sets* and *virtual chains*. Then, we define the common prefix property w.r.t. virtual chains and prove that our protocol can achieve this property. Finally, we demonstrate that the standard common prefix property can be reduced to the common prefix w.r.t. virtual chains. The proofs of lemmas can be found in the full version of our paper.

E.1. Virtual Block-Sets and Virtual Chains

We construct virtual block-sets and then form virtual chains based on them. Intuitively, blocks with the same height that are "close" to each other are grouped into a virtual block-set. Then, a virtual chain is formed by concatenating these virtual block-sets that are linked together. This method is intended to ensure that, at each height, honest players will only extend blocks that belong to the same virtual block-set.

We define two blocks to be "close" as follows: Given two blocks B and B' with the same height, let $\mathcal{C}$ and $\mathcal{C}'$ be the chains from the genesis block to B and B', respectively. The blocks B and B' are considered "close" to each other if the distance from $\mathcal{C}$ to $\mathcal{C}'$ is less than D, i.e. $\mathsf{distance}(\mathcal{C} \to \mathcal{C}') \leq D$.

In our protocol $\Pi^\bullet$, consider an honest player and let $\mathbb{C}_{\mathrm{best}}$ be the set of the player's best chains. For any two chains $\mathcal{C}, \mathcal{C}' \in \mathbb{C}_{\mathrm{best}}$, the distance between $\mathcal{C}$ and $\mathcal{C}'$ is less than D, i.e. $\mathsf{distance}(\mathcal{C} \to \mathcal{C}') \leq D$. Hence, if $\mathsf{len}(\mathcal{C}) = \mathsf{len}(\mathcal{C}')$, the last blocks on $\mathcal{C}$ and $\mathcal{C}'$ are "close" to each other. Note that an honest player only extends chains in the set of best chains. Thus, at each height, an honest player will only extend blocks that belong to the same virtual block-set.

Recall that in Fig. 8, we provide an example of virtual block-sets with $D = 2$. In this example, the set of virtual block-sets is $\mathbb{V} = V_0, V_1, V_2, V_3, V_3', V_4, V_5$. A virtual chain is formed by linking several virtual block-sets together. A virtual block-set V is considered linked to a virtual block-set V' if there exists a block $B \in V$ and a block $B' \in V'$ such that B is linked by B'.

Definition 14 (Virtual block-sets and virtual chains)**.** *Consider an execution of protocol $\Pi^\bullet$, and consider an honest player with the set $\mathbb{C}$ of local chains. Let $\mathbb{B}$ be the set of all blocks on the chains in $\mathbb{C}$.*

Based on the set of block $\mathbb{B}$, we define a set $\mathbb{V}$ of **virtual block-sets***, as follows. Initially, we set $\mathbb{V} := \emptyset$. For each block $B \in \mathbb{B}$, let $\mathcal{C}$ be the chain from the genesis block to the block B. If the block B has not been added to any virtual block-set (i.e., for all $V' \in \mathbb{V}$, we have, $B \notin V'$), we build a virtual block V based on the block B as follows. Initialize that $V := \{B\}$. For any block $B' \in \mathbb{B}$, let $\mathcal{C}'$ be the chain from the genesis block to the block B'. If $\mathsf{len}(\mathcal{C}) = \mathsf{len}(\mathcal{C}')$ and $\mathsf{distance}(\mathcal{C} \to \mathcal{C}') \leq D$, we set $V := V \cup \{B'\}$. Finally, we set $\mathbb{V} := \mathbb{V} \cup \{V\}$.*

Based on the above information, we can further define a set $\mathbb{VC}$ of **virtual chains***, as follows. Initialize that $\mathbb{VC} := \{\mathcal{V}_0\}$, where $\mathcal{V}_0 = V_0 = \{B_0\}$ and B_0 is the genesis block. For each virtual chain $\mathcal{V} = V_0 \| V_1 \| \cdots \| V_\ell$ (where ℓ is a non-negative integer) in the set $\mathbb{VC}$, we construct new virtual chains as follows. First, we define that V_ℓ is linked by the $V_{\ell+1}$ if there exists a block $B_{\ell+1} \in V_{\ell+1}$ and a block $B_\ell \in V_\ell$ such that B_ℓ is linked by*

$B_{\ell+1}$. *For each such virtual block-sets* $V_{\ell+1} \in \mathbb{V}$ *such that* V_ℓ *is linked by* $V_{\ell+1}$*, we construct a new virtual chain* $\mathcal{V}' := \mathcal{V} \| V$ *and set* $\mathbb{VC} := \mathbb{VC} \cup \{\mathcal{V}'\}$. *In the example in Fig. 8,* $V_0 \| V_1 \| V_2 \| V_3'$ *and* $V_0 \| V_1 \| V_2 \| V_3 \| V_4 \| V_5$ *are two virtual chains.*

We define the *best virtual chain* as the virtual chain in which each virtual block-sets in the virtual chain contains a block in the best chains. In Fig. 8, the best virtual chain is $\mathcal{V}_{\text{best}} = V_0 \| V_1 \| V_2 \| V_3 \| V_4 \| V_5$ since the best chain is $\mathcal{C}_{\text{best}} = B_0 \| B_1 \| B_2 \| B_3 \| B_4 \| B_5$. We formally define the best virtual chain as follows.

Definition 15 (The best virtual chain)**.** *Let* $\mathcal{C}_{\text{best}} = B_0 \| B_1 \| \cdots \| B_\ell$ *be the best chain. A virtual chain* $\mathcal{V}_{\text{best}} = V_0 \| V_1 \| \cdots \| V_\ell$ *is the best virtual chain if for all* $i \in [0..\ell]$, $B_i \in V_i$.

Virtual block-set and virtual chain basics. Consider a virtual chain $\mathcal{V}$ consists of a sequence of ℓ concatenated blocks $V_0 \| V_1 \| V_2 \| \cdots \| V_\ell$, where $\ell \in \mathbb{N}$. We use $\mathcal{V}[i]$ to denote the i-th virtual block-set V_i in virtual chain $\mathcal{V}$. Here, the subscript i denotes the block height of the virtual block-set V_i in the virtual chain $\mathcal{V}$. The block height of a virtual block-set V_i is equal to the block height of all blocks in V_i. We refer to $\mathcal{V}[j, m]$, with $j \geq 0$ and $m \leq \ell$, as a sub virtual chain $V_j \| \cdots \| V_m$. If a virtual chain $\mathcal{V}$ is truncated by the last κ virtual block-sets, we write $\mathcal{V}[\neg\kappa]$.

E.2. Common Prefix Property w.r.t. Virtual Chains

We are now ready to define the common prefix property w.r.t. virtual chains.

Definition 16 (Common prefix w.r.t. 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.*

To demonstrate the common prefix property with respect to virtual chains, we introduce the concept of an *honest virtual block-set*, where the first block generated in the virtual block-set is honest. Our analysis shows that there is at most one honest virtual block-set at any block height. This means that to violate the common prefix property, the adversary must extend the virtual chain as quickly as the honest players.

Definition 17 (Honest virtual block-sets)**.** *Consider a virtual block-set* V*, let* B *be the earliest block in* V*, i.e.,* B *is the block with the smallest round number. We say* V *is honest if the earliest block* B *is generated by an honest player.*

We override the equal operator for virtual block-sets since new blocks may be added to the existing virtual block-sets through time. Intuitively, we say two virtual block-sets are equal if all the blocks in the two virtual block-sets are "close".

Definition 18 (Equal operator for virtual block-sets)**.** *Consider two virtual block-sets* V_i *and* V_i' *at the same block height* i*. We say* V_i *equals* V_i' *(i.e.,* $V_i = V_i'$*) if the following constraint is satisfied: For any block* $B \in V_i$ *and any block* $B' \in V_i'$*, let* $\mathcal{C}$ *and* $\mathcal{C}'$ *be the chains from the genesis block to* B *and* B'*, respectively. We have,* $\mathsf{distance}(\mathcal{C} \to \mathcal{C}') \leq D$.

The equal operator for virtual block-sets is symmetric. Given that $V_i = V_i'$. For any block $B \in V_i$ and any block $B' \in V_i'$, we have, $\mathsf{distance}(\mathcal{C}' \to \mathcal{C}) = \mathsf{distance}(\mathcal{C} \to \mathcal{C}') \leq D$. Thus, based on Definition 18, we have, $V_i' = V_i$.

We will demonstrate that there is at most one honest virtual block-set at each block height.

Lemma 4. *Consider an honest player* P*. Let* $\mathcal{V}_{\text{best}} = V_0 \| V_1 \| \cdots \| V_\ell$ *be the best virtual chain in the local state of player* P *at the beginning of round* r*, where* $\ell \in \mathbb{N}$ *is the length of the best chain. If player* P *generates a new chain* $\mathcal{C} = B_0 \| B_2 \| \cdots \| B_{\ell'}$*, where* $\ell' \in \mathbb{N}$*. Then, one of the following two conditions is true: 1)* $\ell' = \ell + 1$ *(a new longest chain is generated); or 2)* $\ell' \leq \ell$ *and* $B_{\ell'} \in V_{\ell'}$ *(the last block of the new chain is added to an existing virtual block-set).*

We assume $\alpha^\bullet = \lambda \cdot \beta^\bullet$, $\lambda > 1$. If $r_1 - r$ is big enough, the adversary cannot extend the virtual chain as fast as the honest players. Now, we are ready to prove the common prefix property on virtual chains.

Lemma 5 (Common prefix w.r.t. virtual chains)**.** *Assume* $\alpha^\bullet = \lambda \cdot \beta^\bullet$, $\lambda > 1$*. Consider an execution of protocol* $\Pi^\bullet$ *with an arbitrary adversary. Consider an honest player* P *in round* r *with the local best virtual-chain* $\mathcal{V}$*, and an honest player* P_1 *in round* r_1 *with the local best virtual-chain* $\mathcal{V}_1$*, respectively, where* $r_1 \geq r$*. Then, we have,*

$$\Pr\left[\mathcal{V}[\neg\kappa] \preceq \mathcal{V}_1\right] \geq 1 - e^{-\Omega(\kappa)}.$$

E.3. From Common Prefix w.r.t. Virtual Chains, to the Standard Common Prefix Property

We prove the common prefix property w.r.t. virtual chains. Lemma 5 establishes that the virtual chains of any two honest players share a common prefix after removing the last κ virtual blocks. All blocks in a virtual block-set have the same common prefix after removing the last D blocks. Let V denote the last common virtual block-set between the two virtual chains. All chains in the virtual block-set V have the same common prefix after removing the last D blocks. Therefore, the chains of any two honest players share the same common prefix after removing the last $\kappa + D$ blocks, thus achieving the common prefix property.

Lemma 6 (Common prefix)**.** *Assume* $\alpha^\bullet = \lambda \cdot \beta^\bullet$, $\lambda > 1$*. Consider an execution of protocol* $\Pi^\bullet$ *with an arbitrary adversary. Consider two honest players,* P *in round* r *with the local best chain* $\mathcal{C}$*, and* P' *in round* r' *with the local best chain* $\mathcal{C}'$*, respectively, where* $r' \geq r$*. Then, we have,*

$$\Pr\left[\mathcal{C}[\neg(\kappa + D)] \preceq \mathcal{C}'\right] \geq 1 - e^{-\Omega(\kappa)}.$$

Appendix F.
Supplemental materials for Section 5.3

We show that our protocol can achieve the chain quality and the best possible unpredictability properties. The proofs of lemmas can be found in the full version of our paper.

F.1. Chain Quality

After proving the chain growth and common prefix properties, the proof of chain quality will be very similar to the proof in [9]. Intuitively, the adversary cannot extend the chain as fast as the growth rate of the best chain. Thus, some blocks on the best chain must be generated by honest players.

Lemma 7 (Chain quality)**.** *Assume $\alpha^\bullet = \lambda \cdot \beta^\bullet$, $\lambda > 1$, and $\delta > 0$. Consider an execution of protocol $\Pi^\bullet$ with an arbitrary adversary. Consider an honest player with chain $\mathcal{C}$. Consider that ℓ consecutive blocks of $\mathcal{C}$, where ℓ_{good} blocks are generated by honest players. Then we have $\Pr\left[\frac{\ell_{good}}{\ell} \geq \mu\right] \geq 1 - e^{-\Omega(\ell)}$ where $\mu = 1 - (1+\delta) \cdot \frac{1}{\lambda}$.*

F.2. Best Possible Unpredictability

We now show that protocol $\Pi^\bullet$ can achieve the best possible unpredictability. In our protocol, players only predict whether or not they can generate the next block, i.e., they are 2-unpredictable. In our protocol, the context of a chain is computed as the last block on the chain. Thus, the contexts of any two different chains in our protocol execution are different. In other words, our protocol $\Pi^\bullet$ archives distinct-context-extension property. Hence, protocol $\Pi^\bullet$ achieves the best possible unpredictability.

Lemma 8. *Consider an execution of protocol $\Pi^\bullet$ with a set of player $\mathcal{P}$. For every* PPT *$\mathcal{Z}, \mathcal{A}$, for any player $P \in \mathcal{P}$ at any round r, we have,*

$$\Pr\left[\begin{array}{l} \mathtt{VIEW} \leftarrow \mathsf{EXEC}_{\Pi^\bullet,\mathcal{A},\mathcal{Z}} \\ (r', z_P^{r'}) \leftarrow \mathcal{A}(P, r, \mathtt{VIEW}^r) \end{array} \middle| \mathsf{predictable}(\mathtt{VIEW}, P, 2, r, r', z_P^{r'}) = 0\right] > 1 - \mathsf{negl}(\kappa).$$