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

Lei Fan[†] Jonathan Katz[‡] Zhenghao Lu[§] Phuc Thai[¶] Hong-Sheng Zhou[||]

August 23, 2024

Abstract

The proof-of-stake (PoS) protocols have been proposed to eliminate the unnecessary waste of computing power in Bitcoin. Multiple practical and provably secure designs have been developed, such as Ouroboros Praos (Eurocrypt 2018), Snow White (FC 2019), and more. However, an important security property called unpredictability has not been carefully studied in these provably secure PoS. Unpredictability property is critical for PoS since the attackers could use predictability to launch strengthened versions of multiple attacks (e.g., selfish-mining and bribing). Unpredictability has previously been investigated by Brown-Cohen et al. (EC 2019) in incentive-driven settings. In this paper, we investigate the property in the cryptographic setting, to achieve the "best possible" unpredictability for PoS.

First, we present an impossibility result for *all* proof-of-stake protocols under the *single-extension* design framework. In this framework, each honest player is allowed to extend exactly one chain in each 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. Then, to overcome the impossibility result, we introduce a new design framework, called *multi-extension* PoS, which allows each honest player to extend *multiple* chains in a round. We develop a novel strategy called "D-distance-greedy" strategy (where D is a positive integer), in which honest players are allowed to extend *a set of best chains that are "close" to the longest chain*. (Of course, malicious players are allowed to behave arbitrarily in the protocol execution.) This "D-distance-greedy" strategy enables us to construct a class of PoS protocols that achieve the best possible unpredictability. Plus, we design a new tiebreak rule for the multi-extension protocol to choose the best chain that can be extended faster. This ensures that the adversary cannot slowdown the chain growth of honest players. Note, these protocols can be proven secure, assuming a much smaller fraction (e.g., 57%) of stake to be honest.

To enable a thorough security analysis in the cryptographic setting, we develop several new techniques. As the players are allowed to extend multiple chains, the analysis of chain growth is highly non-trivial. We introduce a new analysis framework to analyze the chain growth of a multi-extension protocol. To prove the common prefix property, we introduce a new concept called "virtual chains", and then present a reduction from the regular version of the common prefix to "common prefix w.r.t. virtual chains".

[*]This is a replacement of https://eprint.iacr.org/2017/656 and other earlier versions of the work.

[†]Shanghai Jiao Tong University. Email: fanlei@sjtu.edu.cn
[‡]Google and University of Maryland. Email: jkatz@cs.umd.edu
[§]Shanghai Jiao Tong University. Email: zhenghao.lu.sh@gmail.com
[¶]Sky Mavis. Email: phuc.thai@skymavis.com This work was done while at Virginia Commonwealth University.
[||] Virginia Commonwealth University. Email: hszhou@vcu.edu

Contents

1 Introduction

Cryptocurrencies like Bitcoin [Nak08] 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 mechanism [DN93, Bac02]. 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).

Bitcoin-like consensus protocols.

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. As detailed in [GKL15], two distinct hash functions (treated as random oracles) are used to extract the chain context and solve the puzzle, determining who can generate new blocks. The newly created block is then appended to *the longest chain*.

From proof-of-work (PoW) to proof-of-stake (PoS). However, the PoW-based consensus requires substantial computing power. Utilizing alternative resources like *coins (also known as stake)* to secure a blockchain is desirable. If successful, the new system would be environmentally friendly, as it would not rely on extensive computing power for security. Several attempts have been made, with PoS mechanisms widely discussed in the cryptocurrency community (e.g., [NXT14, Kwo14, Vas14, BGM16]). In a PoS-based blockchain protocol, players must prove ownership of a specified number of stakes (coins); only those who can provide such proofs are permitted to participate in maintaining the blockchain.

In PoS-based consensus, to extend the chain, players attempt to find *solutions* to *the hash-based PoS puzzles*. It's important to note that PoS puzzles are defined based on "contexts", usually extracted from previous blocks on the blockchain. In our protocol, the context is the hash value of the last block on the longest chain. At a high level, this context serves as biased randomness to determine which players can generate the next block. The solution to the puzzle is based on stake information, a time step (round number), and the context. In comparison with PoW mechanisms, the computational cost of finding solutions in PoS mechanisms is very "cheap".

From ad hoc to rigorous designs. Early PoS designs (e.g., [NXT14, Kwo14, Vas14, BGM16]) and PoW-based designs, such as the original Bitcoin, were initially crafted in an *ad hoc* style. However, the contemporary trend leans towards a more rigorous approach where security concerns are precisely defined, and the designed protocols undergo mathematical analysis. Notable contributions include the work by Garay et al. [GKL15] and Pass et al. [PSs17], analyzing the PoW-based blockchain in Bitcoin within the *cryptographic setting*. In this context, malicious players may deviate arbitrarily from the protocol, while honest players strictly adhere to the protocol instructions. The analysis demonstrated that the Bitcoin blockchain can achieve crucial security properties, such as common prefix, chain quality, and chain growth. (Please refer to Section 2.2 for formal definitions.) Jumping ahead, as in [PSs17], the analysis of our design will be in the semi-synchronous network setting where the network communication can be delayed with a known bound.

Indeed, research efforts have also been devoted to PoS-based and Bitcoin-like consensus, as seen in [DGKR18, DPS19, BDK$^+$19]; more discussion about these rigorous designs can be found in the Related Work (Section 11). Nevertheless, these protocols are vulnerable to attacks due to predictability.

(Un)predictability. Intuitively, predictability in a protocol implies that certain players are aware they will be selected to generate blockchain blocks before actually doing so. Brown-Cohen et al. [BCNPW19] explored the predictability of PoS in incentive-driven scenarios, where players may deviate from the protocol for higher profits. The power of predictability can be exploited by attackers to reduce the difficulty or cost of incentive-driven attacks like selfish-mining [BCNPW19] or bribery [BDK$^+$19].

In a selfish-mining attack, an attacker gains an unfair advantage by selectively withholding or revealing blocks. Predictability enables the attacker to develop a more effective strategy for such attacks. In a bribery attack, an attacker attempts to influence players to work on specific chains for personal gain, supporting actions like double-spending or censorship attacks. By predicting which players are likely to mine new blocks, attackers can strategically attempt to bribe them. These attacks undermine blockchain fairness and discourage honest participation.

Therefore, it is crucial for a PoS protocol to minimize predictability and mitigate the risks of these attacks. Ideally, a PoS protocol should aim for the best possible unpredictability, enabling effective counteraction of predictability-based attacks. Achieving this goal ensures the maintenance of blockchain fairness and incentivizes honest players to participate in the protocol.

Proof-of-Stake via BFT techniques.

Before we present our results, we must note that, very recently conventional Byzantine Fault Tolerance (BFT) techniques have been significantly improved and then been used for constructing PoS consensus protocols. Notable examples of the BFT-based PoS include Algorand [CM19, GHM$^+$17] and Ethereum's Casper [BG17]; the list can be long. The main distinction between Bitcoin-like PoS and BFT-based PoS is that, the former is *non-interactive* while the latter requires *multiple rounds of interaction* among the protocol participants. Very different from Bitcoin-like PoS (e.g., [DGKR18, DPS19, BDK$^+$19], and the one in this paper), in the BFT-based PoS, a small size committee must be selected from a huge number of PoS protocol participants, and then the **small size** committee[1], through more rounds of interaction, determines who will generate the block. In addition, we note that in the Casper [BG17] protocol, the selected committee in each phase is known, making it susceptible to DDoS attacks. In Algorand [CM19, GHM$^+$17], the selected committee is hidden; however, it introduces more rounds of communication. In this paper, we focus on Bitcoin-like PoS only.

1.1 Our results

Our first result is that we formally define (best possible) unpredictability in the cryptographic setting (see Section 2.3 for the definitions). We say a protocol achieves the best possible unpredictability if it only allows the players to predict whether they can generate the next block (but not 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 that follow a *single-extension* design framework (see Subsection 1.1.1 for more details). Existing provably secure Bitcoin-like PoS protocols (e.g., [DGKR18, DPS19, BDK$^+$19]) are all in the single-extension framework. Secondly, to overcome the impossibility, we introduce a new design framework, called *multi-extension*. We develop a novel D-general-greedy strategy in the multi-extension framework, which allows us to design provably a secure Bitcoin-like PoS protocol (see Subsection 1.1.2). Finally, we present new analysis techniques to analyze the chain growth and the common prefix properties for PoS protocols in the multi-extension framework (see Subsection 1.1.3 and Subsection 1.1.4, respectively). We next elaborate on our major results.

1.1.1 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 [DGKR18], SnowWhite [DPS19], and Bagaria et al. [BDK$^+$19]. In a single-extension protocol, each honest player selects a chain using the best chain algorithm and then attempts to extend it as follows. The player first extracts a context from the chain using a context extraction algorithm. Then, the player uses the context, the current round number, and the secret key to determine whether or not they are allowed to generate a new block at that round.

We have identified an interesting impossibility result for single-extension PoS protocols: *For any single-extension PoS protocol that achieves the best possible unpredictability, it cannot achieve the common prefix property if the honest players control less than 73% of the stake.*

We prove the impossibility based on a new property that we formulated called *distinct-context-extension* in PoS protocols. The distinct-context-extension property states that the contexts of any two valid chains are different. Our proof consists of two steps as follows. First, we show that if the single-extension PoS protocol achieves the best possible unpredictability, it must have the distinct-context-extension property. Intuitively, if the extension of the two different chains is shared-context-extension (the opposite of the distinct-context-extension property), a player can predict whether or not she/he can extend one chain after attempting to extend the other chain. Thus, if the protocol does not achieve the distinct-context-extension property, it cannot achieve the best possible unpredictability. Secondly, we show that if the protocol has the distinct-context-extension property, then it cannot achieve the common prefix property if the honest players control less than 73% stake. We consider an adversary that extends a set of chains privately, and we can bound the chain growth of the adversary by using a random tree to model the chain extension of the adversary. We can show that the adversary can amplify its stake by a factor e, where $e = 2.72$ is the base of the natural logarithm. Therefore, if the honest players control less than 73% stake, the adversary can extend the chain faster than the honest players, thus breaking the common prefix property.

We remark that, we are the first to present the impossibility result for the single-extension protocols. Our previous version and the work in [BDK$^+$19] showed that some single-extension protocols can be secure with 73%

[1]Take Casper for example. In Casper, the recommended minimal committee size is 111, and in practice the committee size is hundreds, which is significantly smaller than the total number of PoS participants in Ethereum.

honest stake. However, those works never claim the impossibility result. To prove the impossibility result, we formally define the single-extension protocol and the new concept of distinct-context-extension property.

1.1.2 New design: Overcoming the impossibility via multi-extension

To overcome the impossibility, we propose a new design framework, called *multi-extension*, for PoS protocols. In a multi-extension protocol, each honest player is allowed to extend multiple chains in a round, rather than just one. We remark that designing a secure and practical multi-extension PoS protocol is challenging. For example, Bagaria et al. [BDK+19] have shown that a protocol allowing honest players to extend slightly shorter chains than the best chain is vulnerable to "balance attacks" which can break the common prefix property. Fortunately, we can have a particular design that follows a novel "D-distance-greedy" strategy, that can be proven to be secure.

D-distance-greedy strategy. We propose a novel *D-distance-greedy* strategy that allows the honest player to a set of best chains, where D is a positive integer. By using the D-distance-greedy 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.

In the D-distance-greedy strategy, the players extend a set of best chains that are "close" to the best chain. We say a chain is "close" to the best chain if a common prefix can be obtained by removing the last D blocks from the best 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.

A new tiebreak rule. In a multi-extension protocol, the probability of generating a new best chain can change depending on the number of chains in the set of best chains. Consider a protocol execution round, where there are two longest chains; different strategies for choosing the best chain may increase or decrease the probability of generating a new one. This creates an opportunity for an adversary to *slow down* chain growth by publishing a chain with the same length but with fewer chains in the set of best chains, as the best chain. To address this issue, we introduce a new tiebreak rule: If there are multiple chains with the same length, then the chain that can be extended the fastest will be selected as the best chain.

Intuitively, honest players will generate a new best chain faster if there are more chains in the set of best chains. To make this more concrete, we can partition the set of best chains into $D + 1$ subsets based on their depth. The depth of a chain is the difference between its length and the length of the current best chain. For $i \in [0..D]$, we denote the i-depth subset as the set of chains at depth i. A new best chain is generated if a player extends a chain in the 0-depth subset, which consists of all chains with the same length as the current best chain. Additionally, for $i \in [0..D-1]$, a new chain is added to the i-depth subset if a chain in the $(i+1)$-depth subset is extended. Therefore, we compare the number of chains in each depth-based subset, from the 0-depth subset up to the D-depth subset, to break ties between chains of equal length. This way, players can select the best chain that can be extended the fastest.

1.1.3 New analysis: Chain growth in multi-extension

To analyze the chain growth property, we propose a new Markov chain analysis framework to study the chain growth in multi-extension protocols. Then, we apply the new analysis framework to analyze the chain growth of our protocol. We consider a *hybrid experiment*, where all messages sent by the adversary are removed. Based on our tiebreak rule, we can show that the chain growth in a real execution is lower bound by the chain growth in a hybrid execution. Note that, the hybrid experiment has been introduced in the analysis in [PSs17] to analyze the chain growth of Bitcoin protocol. In our protocol, the honest players may extend multiple chains in a round. Thus, we design a *random walk* on a *Markov chain* to analyze the chain growth in the hybrid experiment for our protocol.

We start by designing a *simplified Markov chain* and then proceed to design an *augmented Markov chain*. Recall that, the set of best chains can be partitioned into $D + 1$ subsets based on the depth of those chains. 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. We can analyze chain growth by analyzing the expected number of chains in the 0-depth subset of each round. In the simplified Markov chain, each state represents a protocol round with specific numbers of chains in all depth-based subsets. The transitions on the Markov chain depend on how the set of best chains is updated after each round. The simplified Markov chain only gives information about the depth-based subsets, but it doesn't show how many chains are removed when a new chain is generated. This makes it hard to find a good lower bound for the amplification ratio, even with a large D.

To solve the issue in the simplified Markov chain, we propose an augmented Markov chain. We use *depth-distance-based subsets* that select chains based on both their length and distance from the best chain. The augmented Markov chain shows a more detailed representation of the best chains, so we can identify which chains belong to the new set when a new best chain is generated. This gives a better lower bound for the amplification ratio.

We remark that the Markov chain technique has been used to analyze the common prefix property [KRs18] for Bitcoin protocol, which follows a single-extension fashion. However, our Markov chain is very different from the Markov chain in [KRs18]. Since our protocol follows the multi-extension framework, a more complex Markov chain is needed to analyze the chain growth property.

1.1.4 New analysis: Common prefix in multi-extension

Previous analysis of Bitcoin's PoW consensus [GKL15, PSs17] showed that the key factor for establishing the common prefix property is that honest participants can contribute only one block at a block height. Breaking this property requires the adversary to generate more blocks than the honest participants, which is infeasible due to the majority control of mining power by the honest participants. 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 participants 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. In detail, a virtual block-set consists of multiple blocks with the same height that are "close" to each other. We first define two chains are "close", and then define two blocks are "close." We say two chains are close if they share a common prefix in the recent past. When two chains are close, the last blocks of the two chains are also "close". We define the virtual chain consists of multiple virtual block-sets that are linked together. Note that, the adversary cannot use the blocks of honest players to break the common prefix property, the adversary needs to generate more virtual block-sets than the honest players to break the common prefix property (w.r.t virtual chains). This requires the adversary to control the majority of the stake (which contradicts the assumption that the honest players control the majority of the stake). Finally, we show the common prefix w.r.t. virtual chains implies the regular common prefix property. This is given by the fact that the blocks in the same virtual block-set are "close".

1.1.5 Additional analysis and design considerations

Best possible unpredictability. Our protocol minimizes predictability by using the hash value of the last block to extract the context. This ensures that the contexts of any two different chains in the execution of our protocol are different, i.e., our protocol achieves distinct-context-extension property. As a result, players cannot predict the extendibility of a future chain based on the extensions of a current chain. Players can only predict whether or not they can extend the current best chain. Our protocol provides the best possible unpredictability for PoS protocols.

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 [BGK$^+$18], 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

The remainder of the paper is organized as follows. 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.

Next, we provide the security analysis of our new PoS protocol. In Section 5, we provide an overview of the security analysis. In Section 6, we provide a new analysis framework to analyze the chain growth property of a PoS protocol in the multi-extension design framework. Then, in Section 7, we apply the new analysis framework to analyze the chain growth property for our new PoS protocol. In Section 8, we propose a new analysis framework to analyze the common prefix property for the new protocol. In Section 9, we show the chain quality and the best possible unpredictability properties of the new protocol.

Finally, in Section 10, we discuss how to upgrade our protocol to a full-fledged blockchain protocol and show how to enable players to register their key-pairs adaptively. In Section 11, we provide the related work. Supplemental materials for Section 2, Section 3, Section 4, and Section 8 are provided in Appendices A, B, C, and D, respectively.

2 Security Model

2.1 Blockchain protocol executions

The security of Bitcoin-like PoW-based protocols has been rigorously investigated by Garay et al. [GKL15] and then by Pass et al. [PSs17] 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 [GKL15, PSs17].

The execution of a PoS blockchain protocol. Following Canetti's formulation of the "real world" executions [Can00], we present an abstract model for a PoS blockchain protocol Π in the hybrid world of the partially synchronous network communication functionality, the random oracles, and certain initialization functionality.

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

The environment $\mathcal{Z}$ can "manage" protocol 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. The environment $\mathcal{Z}$ also 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 or the functionalities. Players are equipped with (roughly synchronized) clocks that indicate the current round.

In any round r, each PoS-player $P \in \mathcal{P}$, with a local state $state_r$, receives a message from $\mathcal{Z}$, 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 controlled by the adversary, i.e., the adversary $\mathcal{A}$ is responsible for delivering all messages sent by players. The adversary $\mathcal{A}$ can reorder or delay the messages. However, it cannot modify the messages. Plus, any message, that is broadcasted by an honest player, is guaranteed to arrive at all other honest players within a maximum delay of Δ rounds.

At any round r of the execution, $\mathcal{Z}$ can send message $(\textsc{Corrupt}, P)$, where $P \in \mathcal{P}$, to adversary $\mathcal{A}$. Then, 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 `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 $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$. In the current version of our modeling, we assume that when (honest) PoS-players leave the protocol execution, they will *erase* their own local internal information.[2]

Random oracles. As mentioned, we assume the availability of random oracles that capture the idealization of hash functions. Two hash functions are used for computing the context of the chains, and for creating the puzzles, respectively; note, to create a puzzle, the context of (certain) chains will be computed first.

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 initial block (genesis block). We use $\mathsf{len}(\mathcal{C})$ to denote *blockchain length*, i.e., the number of blocks in blockchain $\mathcal{C}$; and here $\mathsf{len}(\mathcal{C}) := \ell$. (Note that since all chains must consist of the genesis block, we do not count it as part of the chain's length. In other words, a chain $\mathcal{C}$ with length ℓ actually has $\ell + 1$ blocks in total.) We use sub blockchain (or subchain) for referring to a segment of a chain; here for example, $\mathcal{C}[0, \ell]$ refers to

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

an entire blockchain, whereas $\mathcal{C}[j,m]$, with $j \geq 0$ and $m \leq \ell$ would refer to a sub blockchain $B_j \| \cdots \| B_m$. We use $\mathcal{C}[i]$ to denote the i-th block, B_i in blockchain $\mathcal{C}$; here i denotes the *block height* of B_i in chain $\mathcal{C}$.

If blockchain $\mathcal{C}$ is a prefix of another blockchain $\mathcal{C}_1$, we write $\mathcal{C} \preceq \mathcal{C}_1$. If a chain $\mathcal{C}$ is truncated the last κ blocks, we write $\mathcal{C}[\neg\kappa]$. For some $\mathcal{A}, \mathcal{Z}$, consider some $\mathtt{VIEW}$ in the support of $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$. We use the notation $\mathtt{VIEW}^r$ to denote the prefix of $\mathtt{VIEW}$ up until round r. Let $\mathbb{C}^r$ be the set of chains in $\mathtt{VIEW}^r$, and let $\mathcal{C}_i^r$ be the chain in the view of player i at round r.

2.2 Chain growth, common prefix, and chain quality

Previously, several fundamental security properties for Bitcoin-like PoW-based blockchain protocols have been defined: *common prefix property* [GKL15, PSs17], *chain quality property* [GKL15], and *chain growth property* [KP15]. Intuitively, the chain growth property states that the chains of honest players should grow linearly to the number of rounds. The common prefix property indicates the consistency of any two honest chains except the last κ blocks. The chain quality property aims to indicate the number of honest blocks' contributions that are contained in a sufficiently long and continuous part of an honest chain. Specifically, for parameters $\ell \in \mathbb{N}$ and $\mu \in (0,1)$, the ratio of blocks, that are generated by honest players, in a 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 2.1 (Chain growth). *Consider a blockchain protocol Π with a set $\mathcal{P}$ of players. The chain growth property with parameter $g \in \mathbb{R}$, states: for any honest player P_1 with local chain $\mathcal{C}_1$ at round r_1, and honest player P_2 with local chain $\mathcal{C}_2$ at round r_2, where $P_1, P_2 \in \mathcal{P}$ and $r_2 - r_1 = \Omega(\kappa)$, 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 2.2 (Common prefix). *Consider a blockchain protocol Π with a set $\mathcal{P}$ of players. The common prefix property states the following: for any honest player P_1 adopting local chain $\mathcal{C}_1$ at round r_1, and honest player P adopting local chain $\mathcal{C}$ at round r, in the execution $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$, where $P_1, P \in \mathcal{P}$ and $r \leq r_1$, it holds that $\mathcal{C}[\neg\kappa] \preceq \mathcal{C}_1$.*

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

2.3 Unpredictability

The unpredictability property has been investigated by Brown-Cohen et al. [BCNPW19] 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 materials A.1.)

In this subsection, we investigate the unpredictability property in the cryptographic setting. For any environment $\mathcal{Z}$ and any adversary $\mathcal{A}$, consider some $\mathtt{VIEW}$ in the support of $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$. 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 the prediction: here, $z_P^{r'} = 1$ means that the adversary $\mathcal{A}$ 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 $\bar{z}_P^{r'} := 0$.

We say a prediction $z_P^{r'}$ by the adversary is considered **accurate** if $z_P^{r'} = \bar{z}_P^{r'}$.

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.

Definition 2.4 (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,*

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

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

The best possible unpredictability for any PoS protocol is 2-unpredictability. As already shown in their Observation 1 by Brown-Cohen et al. [BCNPW19], 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, i.e., 2-unpredictability is the best possible unpredictability. We formally define the best possible unpredictability as follows.

Definition 2.5 (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,*

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

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

3 An Impossibility Result

We investigate an impossibility result for a group of proof-of-stake protocols, namely, *single-extension PoS protocols*. The result states that a single-extension PoS protocol cannot achieve both the common prefix and the best possible unpredictability properties if honest players control less than 73% of the stake.

We studied existing Bitcoin-like PoS protocols, such as Ouroboros Praos [DGKR18] and SnowWhite [DPS19]. These protocols use the same context to generate multiple blocks on the blockchain during an epoch. This means that any two chains within the same epoch share the same context. We refer to this as a *shared-context-extension*. Players attempt to extend the chain by solving puzzles based on the chain's context. If two chains share the same context, players can extend both chains simultaneously. Thus, the shared-context-extension property allows for predictability in the PoS protocol execution. When a player receives one chain with shared-context-extension, they can predict whether or not they can extend the other chain in the future.

In contrast to the existing PoS designs, in Bitcoin, the contexts of two different chains are always different. Therefore, the extension of one chain does not affect the extension of any other chain. This property is referred to as *distinct-context-extension*, and it provides the best possible unpredictability for PoS protocols. Players can only obtain the context of a chain when they receive it, thus allowing them to determine whether they can extend the chain or not.

Based on the distinct-context-extension property, we investigate an impossibility result for single-extension PoS protocols. We first describe the single-extension proof-of-stake framework in Subsection 3.1. Then, in Subsection 3.2, we state the impossibility result for the single-extension proof-of-stake protocols. In Subsection 3.3, we present a new definition of distinct-context-extension property and prove the impossibility result for the single-extension proof-of-stake protocols in Subsection 3.4 and Subsection 3.5.

3.1 Single-extension proof-of-stake protocols

We now describe a design framework, called *single-extension* framework, for Bitcoin-like PoS protocols. Intuitively, in a single-extension protocol, in each round, each honest player identifies only a single "best chain", and then extends the chosen best chain. We remark that the state-of-the-art PoS protocols (e.g., [DPS19, DGKR18, BDK+19]) can be categorized as single-extension PoS protocols (see Supplemental material B.1 for more details).

We emphasize that we focus on Bitcoin-like PoS protocols only; the players can generate new blocks in a *non-interactive* fashion by solving PoS puzzles. Our design framework *cannot* be applied for BFT-like protocols (e.g., Algorand [CM19, GHM+17]); in those protocols, the players must interact with each other to generate new blocks.

Definition 3.1 (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* r *as input and returns* 1 *if the chain* $\mathcal{C}$ *is valid at round* r*, and returns* 0 *otherwise.*

- *The context extraction algorithm* 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* hash : $\{0,1\}^* \to \{0,1\}^*$. *Note that hash function* hash *will be treated as a random oracle in our analysis.*

  More concretely, the input chain $\mathcal{C}$ is parsed into $B_0, B_1, \cdots, B_\ell$, where $\ell \in \mathbb{N}$. The algorithm Context *returns a context $\eta := \mathsf{hash}(B_{i_1}\|\cdots\|B_{i_t})$. Here, $i_j \in [0..\ell]$ for all $j \in \{1,\ldots,t\}$. (Note that, the algorithm* Context *returns $\perp$ when the input chain $\mathcal{C}$ is invalid.)*

  We remark that the state-of-the-art PoW (e.g., Bitcoin [GKL15, PSs17]) and PoS (e.g., [DPS19, DGKR18, BDK+19]) use the context extraction algorithms in the above format. For example, in Bitcoin [GKL15, PSs17], the context is computed as the hash value of the last block on the chain. In Ouroboros Praos [DGKR18], the context is computed as the hash value of one or multiple blocks from the previous epoch. In Snow White [DPS19], 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. More details can be found in the Supplemental material B.1.*
- *The extension algorithm* Extend *is parameterized by a probability $p \in (0,1)$. The algorithm* Extend *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* Extend(η, r, SK) *returns a block B with probability p.*
- *The best chain algorithm* 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 $\mathcal{C}_{\mathrm{best}}$. Thus, we name the protocol* ***single-extension****.*

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.*

Algorithm 1: A single-extension proof-of-stake 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.
1 Upon receiving a chain $\mathcal{C}'$, set $\mathbb{C} := \mathbb{C} \cup \{\mathcal{C}'\}$ after verifying Validate$(\mathcal{C}', r) = 1$;
2 Set $\mathcal{C}_{\mathrm{best}} :=$ BestChain$(\mathbb{C})$;
3 Set $\eta :=$ Context$(\mathcal{C}_{\mathrm{best}})$; Set $B :=$ Extend$(\eta, r, \mathrm{SK})$;
4 if $B \neq \perp$ then
5     Set $\mathcal{C} := \mathcal{C}_{\mathrm{best}} \| B$; Add $\mathcal{C}$ to the set $\mathbb{C}$; Broadcast $\mathcal{C}$;
```

Blockchain extension phase. *A single-extension proof-of-stake protocol Π is described in Algorithm 1. In each round r, a player P with the secret key* SK *proceeds as follows. First, the player P computes $\mathcal{C}_{\mathrm{best}} :=$* BestChain$(\mathbb{C}, r)$. *Here the local set of chains $\mathbb{C}$ consists of all valid chains that are received (or generated) by P. Then, the player P uses the function* Context *to compute the context η in the best chain $\mathcal{C}_{\mathrm{best}}$, i.e., $\eta :=$* 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.*

3.2 Impossibility result for single-extension proof-of-stake protocols

We present an impossibility result for single-extension PoS protocols. More 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 simultaneously maintain fundamental security properties, such as the common prefix, when honest players control less than 73% of the stake.

We will prove the impossibility of the result through the distinct-context-extension property. This property is essential for the protocol to achieve the best possible unpredictability. Specifically, it ensures that the contexts of any two different chains in the execution are different, making it impossible for an adversary to predict future chain extensions based on past extensions. However, this property also allows an adversary to amplify their stake by a factor of $e = 2.72$, enabling them to break the common prefix property with only 27% of the stake. Therefore, if the honest players control less than 73% of the stake, it becomes impossible to simultaneously achieve both the distinct-context-extension and common prefix properties, resulting in our impossibility result.

We remark that the aforementioned 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 needed 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 Proof-of-Stake (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 (e.g., common prefix property) when 51% of the mining power is honest.

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 $\beta = 1-(1-p)^{N\cdot\rho}$. Later, we will show that if the protocol Π achieves the distinct-context-extension property, the adversary can amplify its stake by a factor of $e = 2.72$. Therefore, if $\alpha < 2.72\beta$ (i.e., less than 73% of the total stake is honest), the protocol cannot achieve the common prefix property. We are now ready to state the impossibility theorem for protocol Π.

Theorem 3.2. *Consider a single-extension PoS protocol Π that achieves the best possible unpredictability. If $\alpha < 2.72\beta$, then protocol Π cannot achieve common prefix property.*

We describe in Figure 1, the roadmap for the proof of the impossibility theorem.

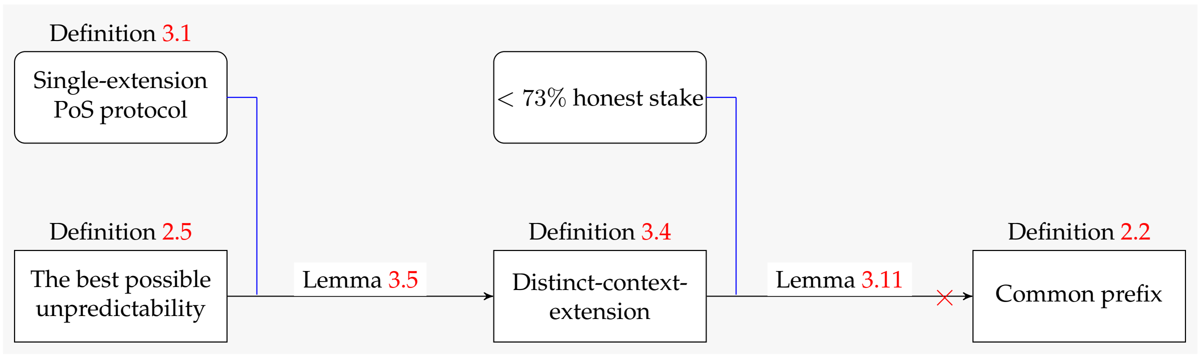


Figure 1: The roadmap for the proof of our impossibility result (Theorem 3.2). First, we show in Lemma 3.5 that if a single-extension PoS protocol achieves the best possible unpredictability, it must achieve distinct-context-extension property. Secondly, we show in Lemma 3.11 that if a single-extension PoS protocol achieves distinct-context-extension property and the honest players control less than 73% of stake, the protocol *cannot* achieve common prefix property.

Proof. We prove Theorem 3.2 in the following two steps. First, we show in Lemma 3.5 that if the single-extension PoS protocol Π achieves the best possible unpredictability, it must achieve *distinct-context-extension* property. Secondly, we show in Lemma 3.11 that if the single-extension PoS protocol Π achieves distinct-context-extension property, it cannot achieve common prefix property if $\alpha < 2.72\beta$. Specifically, if 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. Thus, if $\alpha < 2.72\beta$, the adversary can extend the chain faster than the honest player. Hence, they 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 best 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. □

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. Then, based on the distinct-context-extension property, we can prove the impossibility result in two steps, as mentioned above. The proof of the second step has been shown in [BDK+19]. However, the impossibility result has not been presented in those works.

3.3 Distinct-context-extension

We introduce a new definition for the *distinct-context-extension* property. As previously stated, we will use this property to prove an impossibility result for single-extension PoS protocols. 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. At the same time, this also allows the adversary to amplify their stake by a factor of $e = 2.72$, enabling them to break the common prefix property if they control more than 27% of the stake. Therefore, it is impossible to simultaneously achieve the distinct-context-extension and common prefix properties if the adversary controls more than 27% of the stake.

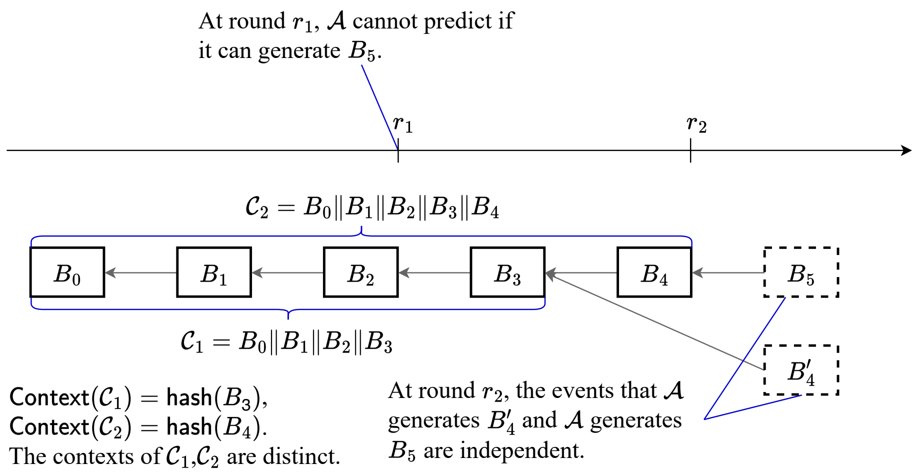

At round r_1, $\mathcal{A}$ cannot predict if it can generate B_5.
r_1
r_2
$\mathcal{C}_2 = B_0\|B_1\|B_2\|B_3\|B_4$
B_0
B_1
B_2
B_3
B_4
B_5
B'_4
$\mathcal{C}_1 = B_0\|B_1\|B_2\|B_3$
$\mathsf{Context}(\mathcal{C}_1) = \mathsf{hash}(B_3)$, $\mathsf{Context}(\mathcal{C}_2) = \mathsf{hash}(B_4)$. The contexts of $\mathcal{C}_1$,$\mathcal{C}_2$ are distinct.
At round r_2, the events that $\mathcal{A}$ generates B'_4 and $\mathcal{A}$ generates B_5 are independent.

Figure 2: 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.

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 [DGKR18] and SnowWhite [DPS19], 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. The toy examples of distinct-context-extension and shared-context-extension can be seen in Figure 2 and Figure 3, respectively.

Definition 3.3 (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, namely, 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.*

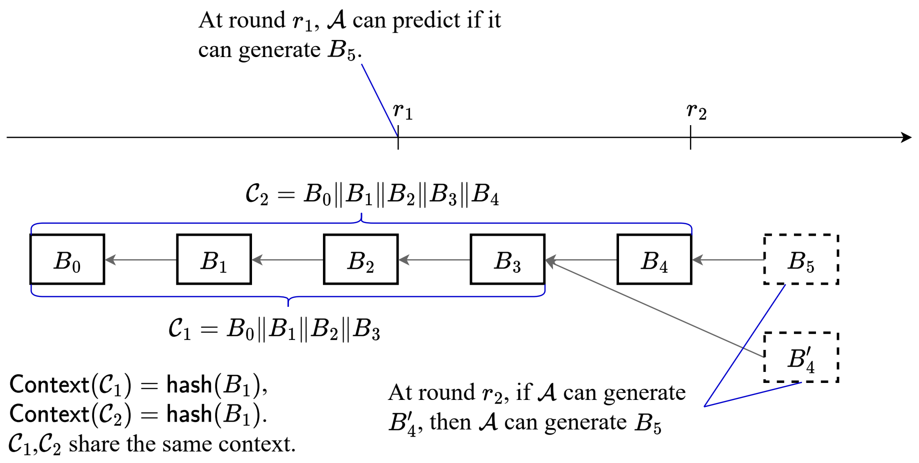

At round r_1, $\mathcal{A}$ can predict if it can generate B_5.
r_1
r_2
$\mathcal{C}_2 = B_0\|B_1\|B_2\|B_3\|B_4$
B_0
B_1
B_2
B_3
B_4
B_5
B_4'
$\mathcal{C}_1 = B_0\|B_1\|B_2\|B_3$
$\mathsf{Context}(\mathcal{C}_1) = \mathsf{hash}(B_1)$,
$\mathsf{Context}(\mathcal{C}_2) = \mathsf{hash}(B_1)$.
$\mathcal{C}_1$,$\mathcal{C}_2$ share the same context.
At round r_2, if $\mathcal{A}$ can generate B_4', then $\mathcal{A}$ can generate B_5

Figure 3: A toy example of shared-context-extension for two chains. The definition of function Context here is very different from that in Figure 2. 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 Figure 2, $\mathsf{Context}(\mathcal{C}_1) = \mathsf{hash}(B_3)$ 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.

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's execution have distinct contexts.

Definition 3.4 (Distinct-context-extension for all chains in the executions)**.** *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.*

3.4 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, as shown in Figure 3, if two chains are shared-context-extension, the adversary can predict whether or not it can extend a chain in the future (i.e., the chain is not yet generated). This contradicts the best possible unpredictability. More concretely, 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 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 3.5. *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.*

Proof. We assume toward contradiction that there exist two chains $\mathcal{C}_1$ and $\mathcal{C}_2$ such that $\mathcal{C}_1 \neq \mathcal{C}_2$ and the extensions of those two chains are shared-context-extension. We will prove that the protocol Π cannot achieve the best possible unpredictability, i.e., there exists a round r and a malicious player P such that the player P is 2-predictable at round r.

Let $\eta_1 = \mathsf{Context}(\mathcal{C}_1)$ and $\eta_2 = \mathsf{Context}(\mathcal{C}_2)$. We have, $\eta_1 = \eta_2$. We parse $\mathcal{C}_1$ into $B_0 \| B_1 \| \cdots \| B_{\ell_1}$ and parse $\mathcal{C}_2$ into $B'_0 \| B'_1 \| \cdots \| B'_{\ell_2}$. Here, $\ell_1 = \mathsf{len}(\mathcal{C}_1)$ and $\ell_2 = \mathsf{len}(\mathcal{C}_2)$ are the lengths of $\mathcal{C}_1, \mathcal{C}_2$, respectively. Without loss of generality, we assume $\ell_1 \geq \ell_2$. Let r_1, r_2 be the round where $\mathcal{C}_1$, $\mathcal{C}_2$ are generated. Here, $\mathcal{C}_1$ is the best chain at round r_1.

Let $\mathcal{C}$ be the longest common prefix of $\mathcal{C}_1$ and $\mathcal{C}_2$. Let $\ell = \mathsf{len}(\mathcal{C})$ be the length of chain $\mathcal{C}$. We have, for all $i \in [0..\ell]$, $B_i = B'_i$; and for all $i \in [\ell+1, \ell_2]$, $B_i \neq B'_i$. Let r be the round where $\mathcal{C}$ is generated. We will prove that player P is 2-predictable at round r. In other words, at round r, the player P can predict whether or not she/he can extend the chain $\mathcal{C}_1$ at round r.

We will show that the contexts of $\mathcal{C}_1$ and $\mathcal{C}_2$ can be computed when the chain $\mathcal{C}$ is generated. As shown in Definition 3.1, the contexts are computed based on the hash values of some blocks on the chain. Let $B_{i_1}, \cdots, B_{i_t}$ be the blocks that are used to compute the context of $\mathcal{C}_1$, where $t \in \mathbb{N}$ and for all $j \in [t]$, $i_j \in [0..\ell_1]$. We have, $\eta_1 = \mathsf{hash}(B_{i_1} \| \cdots \| B_{i_t})$. Let $B'_{i'_1}, \cdots, B'_{i'_{t'}}$ be the blocks that are used to compute the context of $\mathcal{C}_2$, where $t \in \mathbb{N}$ and for all $j \in [t']$, $i'_j \in [0..\ell_2]$. We have, $\eta_2 = \mathsf{hash}(B'_{i'_1} \| \cdots \| B'_{i'_{t'}})$.

The function hash is treated as a random oracle[3]. Note that $\eta_1 = \eta_2$. Now we have, $B_{i_1} \| \cdots \| B_{i_t} = B'_{i'_1} \| \cdots \| B'_{i'_{t'}}$. Thus, we have $t = t'$ and for all $j \in [t]$, $B_{i_j} = B'_{i'_1}$. Hence, all blocks $B_{i_1}, \cdots, B_{i_t}$ must belong to the chain $\mathcal{C}$, the common prefix of $\mathcal{C}_1$ and $\mathcal{C}_2$. In other words, the contexts η_1 and η_2 must be obtained from the chain $\mathcal{C}$.

Next, we will show that chain $\mathcal{C}_1$ is longer than chain $\mathcal{C}$, i.e., $\mathsf{len}(\mathcal{C}_1) > \mathsf{len}(\mathcal{C})$. We consider two cases based on the length of $\mathcal{C}_1$ and $\mathcal{C}_2$ as follows:

- Case 1: $\mathsf{len}(\mathcal{C}_1) > \mathsf{len}(\mathcal{C}_2)$. As $\mathcal{C}$ is a prefix of $\mathcal{C}_2$, we have, $\mathsf{len}(\mathcal{C}_2) \geq \mathsf{len}(\mathcal{C})$. Thus $\mathsf{len}(\mathcal{C}_1) > \mathsf{len}(\mathcal{C}_2) \geq \mathsf{len}(\mathcal{C})$.
- Case 2: $\mathsf{len}(\mathcal{C}_1) = \mathsf{len}(\mathcal{C}_2)$. Assume toward contradiction that $\mathsf{len}(\mathcal{C}_1) = \mathsf{len}(\mathcal{C})$. As $\mathcal{C}$ is a prefix of $\mathcal{C}_1$, we have $\mathcal{C} = \mathcal{C}_1$. Similarly, we have $\mathcal{C} = \mathcal{C}_2$. Thus, we have $\mathcal{C}_1 = \mathcal{C}_2$ (this contradicts the condition that $\mathcal{C}_1 \neq \mathcal{C}_2$). Hence, we have, $\mathsf{len}(\mathcal{C}_1) > \mathsf{len}(\mathcal{C})$.

Let r be the round where player P receives $\mathcal{C}$. At round r, player P can calculate the context η_1 of the chain $\mathcal{C}_1$. At round r, the adversary makes a prediction $z_P^{r_1}$, where

$$z_P^{r_1} = \begin{cases} 0, \text{ if } \mathsf{Extend}(\eta_1, r_1, \mathtt{SK}) = \perp, \\ 1, \text{ if } \mathsf{Extend}(\eta_1, r_1, \mathtt{SK}) \neq \perp. \end{cases}$$

If the function $\mathsf{Extend}(\eta_1, r_1, \mathtt{SK})$ returns a block, i.e., $\mathsf{Extend}(\eta_1, r_1, \mathtt{SK}) \neq \perp$, player P can extend the chain $\mathcal{C}_1$ at round r_1. Hence, the prediction $z_P^{r_1} = 1$ is always accurate. Furthermore, since $\mathsf{len}(\mathcal{C}_1) - \mathsf{len}(\mathcal{C}) \geq 1$, i.e., $\mathsf{len}(\mathcal{C}_1) + 1 - \mathsf{len}(\mathcal{C}) \geq 2$, we have, $\mathsf{predictable}(\mathtt{VIEW}, P, 2, r, r_1, z_P^{r_1}) = 1$. In other words, the malicious player P is 2-predictable at round r. This contradicts the fact that protocol Π achieves the best possible unpredictability, i.e., player P must be 2-unpredictable at every round. $\square$

We note that the single-extension protocols in [DPS19, DGKR18] 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. [BDK+19] 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 Subsection 3.5, we will show that the protocol requires 73% of honest stake to achieve the security properties.

[3]It is sufficient to assume that hash is collision resistant hash function in this proof.

3.5 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 3.6 (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* $\texttt{VIEW}$ *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,*

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

Proof. Consider an adversary $\mathcal{A}$ that corrupts $N \cdot \rho$ players at the start of the protocol. The adversary $\mathcal{A}$ extends all chains in $\mathbb{C}^r$ for each round r. For each malicious player P, let $\textsc{sk}$ be their secret key. The adversary instructs player P to run $\mathsf{Extend}(\mathsf{Context}(\mathcal{C}), r, \textsc{sk})$. Recall that, the algorithm $\mathsf{Extend}(\mathsf{Context}(\mathcal{C}), r, \textsc{sk})$ returns $\perp$ if no new block is generated. Otherwise, if $\mathsf{Extend}(\mathsf{Context}(\mathcal{C}), r, \textsc{sk}) \neq \perp$, i.e., the algorithm returns a new block, the player P can generate a new block.

The adversary can generate a new block in a round r if there exists a malicious player P' successfully generates a new block, i.e., $\mathsf{Extend}(\mathsf{Context}(\mathcal{C}), r, \textsc{sk}') \neq \perp$, where $\textsc{sk}'$ is the secret key of P'. In this case, the adversary returns the block that is output by the algorithm $\mathsf{Extend}(\mathsf{Context}(\mathcal{C}), r, \textsc{sk}')$. Otherwise, if all malicious players cannot generate a new block at round r, the adversary returns $\perp$.

Recall that, the algorithm $\mathsf{Extend}(\mathsf{Context}(\mathcal{C}), r, \textsc{sk})$ returns a new block with probability p. As the number of malicious players is $N \cdot \rho$, the probability that there exists a malicious play can extend the chain $\mathcal{C}$ at round r is $1 - (1-p)^{N \cdot \rho} = \beta$. In this case, the adversary returns a block core that is generated by a malicious player at round r. □

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. The extension of the adversary is modeled as a random tree, in which each branch represents the extension of a chain. Based on Lemma 3.6, the probability of the adversary extending a chain in each round is at least β. Additionally, as stated in Lemma 3.5, 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 3.7 (Theorem 1 in [Can17]). *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 3.8 (Theorem 3 in [Goe15]). *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 3.9). Then, we use the union bound for the maximum length of all branches.

Claim 3.9. *Consider the set of branches Z_t at time t. For any $\epsilon'' > 0$, we have $\Pr[G_t < (\beta+1)^{(1-\epsilon'')\cdot t}] < e^{-\Omega(t)}$.*

Proof. In each round, on average, the adversary can create $\beta + 1$ new branches from a branch in the previous rounds. Thus, for $j \in \mathbb{N}$, we have $\mathbb{E}\big[\frac{G_{j+1}}{G_j}\big] = \beta + 1$. In other words, we have $\mathbb{E}\big[\log(G_{j+1}) - \log(G_j)\big] = \log(\beta+1)$.

Let $Q_1, Q_2, \cdots, Q_t$ be independent and identically distributed random variables with the expected value of $\log(\beta+1)$. We have, $\log G_t = \sum_{j=1}^{t} Q_j$. Therefore,

$$\Pr\big[G_t < (\beta+1)^{(1-\epsilon'')\cdot t}\big] = \Pr\big[\log(G_t) < (1-\epsilon'')\cdot t \cdot \log(\beta+1)\big]$$
$$= \Pr\Big[\sum_{j=1}^{t} Q_j < (1-\epsilon'')\cdot t\cdot \log(\beta+1)\Big] < e^{-\Omega(t)}.$$

□

Claim 3.10. *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 $\mathtt{VIEW}$ 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,*

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

where $e = 2.72$.

Proof. Let $Y := \sum_{j=1}^{t} X_j$ be a Poisson random variable with the expected value of $k \cdot \beta$. We have,

$$\begin{aligned} \Pr\big[T_t < (1-\epsilon)\cdot t\cdot\beta\cdot e\big] \leq & \sum_{i\in\{1,2,\cdots,G_t\}} \Pr\big[\ell_{t,i} < (1-\epsilon)\cdot t\cdot\beta\cdot e\big] && \text{(Using union bound)} \\ \leq & G_t \cdot \Pr\big[Y < (1-\epsilon)\cdot t\cdot\beta\cdot e\big] && \text{(Based on Claim 3.8)} \\ \leq & (\beta+1)^{(1+\epsilon'')\cdot t} \cdot \Pr\big[Y < (1-\epsilon)\cdot t\cdot\beta\cdot e\big] + e^{-\Omega(\kappa)} && \text{(Based on Claim 3.9)} \\ \leq & \Pr\big[Y < (1-\epsilon')\cdot t\cdot\beta\big] + e^{-\Omega(\kappa)} = e^{-\Omega(\kappa)} && \text{(Based on Claim 3.7).} \end{aligned}$$

□

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 3.11. *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.*

Proof. Consider an adversary that extends a set of chains and keeps them private from the beginning of the protocol execution. Consider a round $r = \Omega(\kappa)$, and let $\mathcal{C}_1$ be the best chain that is generated by the adversary. Here, $\mathcal{C}_1$ is private, i.e., it is hidden from the honest players. From Claim 3.10, we have $\Pr[\mathsf{len}(\mathcal{C}_1) < (1-\epsilon_1)\cdot r\cdot e\cdot\beta] < e^{-\Omega(\kappa)}$, where $e = 2.72$. Let $\mathcal{C}_2$ be the best public chain at round r. Using Chernoff bound, we have, $\Pr[\mathsf{len}(\mathcal{C}_2) > (1+\epsilon_2)\cdot r \cdot \alpha] < e^{-\Omega(\kappa)}$. Here, we choose ϵ_1, ϵ_2 such that $(1-\epsilon_1)\cdot e \cdot \beta > (1+\epsilon_2)\cdot \alpha$. We have, $\Pr[\mathsf{len}(\mathcal{C}_1) > \mathsf{len}(\mathcal{C}_2)] > \Pr[(1-\epsilon_1)\cdot r \cdot e \cdot \beta > (1+\epsilon_2)\cdot \alpha] - e^{-\Omega(\kappa)} = 1 - e^{-\Omega(\kappa)}$.

At some round $r' = \Omega(\kappa)$ such that $\mathsf{len}(\mathcal{C}_1) > \kappa$, the adversary publishes the best private chain $\mathcal{C}_1$. Recall that, $\Pr[\mathsf{len}(\mathcal{C}_1) > \mathsf{len}(\mathcal{C}_2)] > 1 - e^{-\Omega(\kappa)}$. In other words, with overwhelming probability, the private chain $\mathcal{C}_1$ is longer than the best public chain $\mathcal{C}_2$. Therefore, the honest players will adopt $\mathcal{C}_1$ as the best public chain.

Note that, all blocks in the private chains $\mathcal{C}_1$ (except the genesis block) do not belong to the public chain $\mathcal{C}_2$. Thus, we have, $\mathcal{C}_1[\neg\kappa] \not\preceq \mathcal{C}_2$. Therefore, the common prefix property does not hold. □

4 Greedy Strategies: How to overcome the impossibility

In Section 3, we have demonstrated the impossibility of single-extension PoS protocols. Specifically, we have shown that a single-extension PoS protocol that achieves the best possible unpredictability cannot simultaneously achieve the common prefix property if honest players control less than 73% of the stake. In this section, we introduce a *multi-extension* framework that allows honest players to extend multiple chains (see Supplemental Material C.2 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 multi-extension protocol can achieve the best possible unpredictability while only requiring a much smaller fraction (e.g., 57%) of the honest stake to achieve the 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 a non-flat model where the PoS-players may have different numbers of stakes, we can set the difficulty of the hash inequality based on the number of stakes that the player controls (see more details in Section 10.2). Plus, we assume all protocol players have their stake registered at the beginning of the protocol execution. In Section 10.3, we will turn to consider a dynamic stake distribution and use a similar strategy as in [BGK$^+$18] to allow new players to join the system during the protocol execution.

4.1 Greedy strategies

We allow the PoS 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 the 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. Next, we will formally study the greedy strategies.

Distance-greedy strategies. We first introduce *distance-greedy strategies* for honest protocol players. Consider a blockchain protocol execution. In each player's local view, there are multiple chains, which can be viewed as a tree. More concretely, the genesis block is the root of the tree, and each path from the root to another node is essentially a chain. The tree will "grow": the length of each existing chain may increase, and new chains may be created, round after round. Before giving the formal definition for distance-greedy strategies, we define the "distance" between two chains in a tree. 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 4.1 (Distance between two chains). *Let $\mathcal{C}$ be a chain of length ℓ. We view $\mathcal{C}$ as the "reference" chain, and now consider $\mathcal{C}_1$ to be a "branch" chain (of the reference chain). Let ℓ_1 be the length of $\mathcal{C}_1$. Next, we define the distance between the reference chain $\mathcal{C}$ and the branch chain $\mathcal{C}_1$, and we use* $\mathsf{distance}(\textit{branch chain} \rightarrow \textit{reference chain})$, *i.e.,* $\mathsf{distance}(\mathcal{C}_1 \rightarrow \mathcal{C})$ *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* $\mathsf{distance}(\mathcal{C}_1 \rightarrow \mathcal{C}) = d$.

Remark 4.2. *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* $\mathsf{distance}(\mathcal{C} \rightarrow \mathcal{C}_1) \neq \mathsf{distance}(\mathcal{C}_1 \rightarrow \mathcal{C})$. *For example, in Figure 4, the distance of $\mathcal{C}$ from $\mathcal{C}_1$ is 3, i.e.* $\mathsf{distance}(\mathcal{C} \rightarrow \mathcal{C}_1) = 3$, *while the distance of $\mathcal{C}_1$ from $\mathcal{C}$ is 2, i.e.,* $\mathsf{distance}(\mathcal{C}_1 \rightarrow \mathcal{C}) = 2$. *We also note that the distance of $\mathcal{C}$ from itself is always 0, i.e.,* $\mathsf{distance}(\mathcal{C} \rightarrow \mathcal{C}) = 0$.

After explaining the concept of the *distance between two chains*, we are ready to introduce the distance-greedy strategies. Intuitively, a player who plays a distance-greedy strategy will make attempts to extend a *set of best*

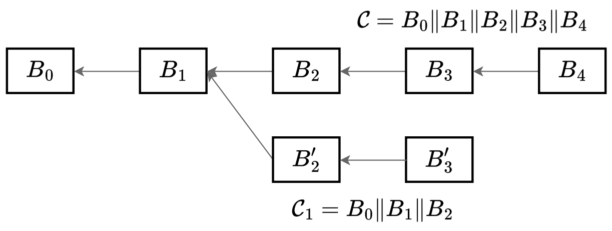

$\mathcal{C} = B_0\|B_1\|B_2\|B_3\|B_4$
B_0
B_1
B_2
B_3
B_4
B'_2
B'_3
$\mathcal{C}_1 = B_0\|B_1\|B_2$

Figure 4: 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.

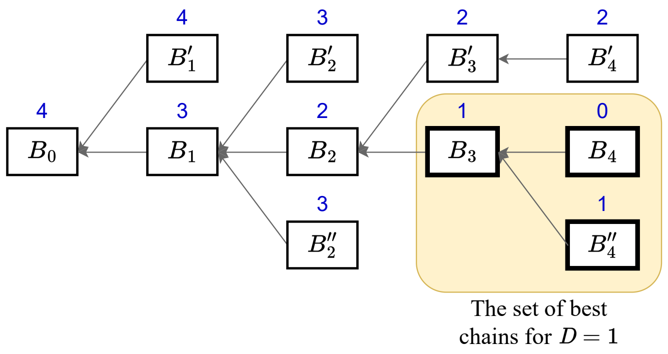


Figure 5: A toy example of 1-distance-greedy strategy. Here, the best chain is $\mathcal{C}_{\text{best}} = B_0\|B_1\|B_2\|B_3\|B_4$. The number in blue on top of each block denotes the distance from the *best chain* $\mathcal{C}_{\text{best}}$ (i.e., the branch chain) to the *reference chain* that consists 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.

chains in which those chains should be very "close" to the best chain, i.e., the distance from the best chain to the chain must be 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. More formally, we have the following definition.

Definition 4.3 (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*

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

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

In Figure 5, a pictorial illustration for the toy example of the 1-distance-greedy strategy can be found. The honest players will extend the bold blocks (B_3, B_4, B''_4).

4.2 The 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. To simplify the presentation, we construct a protocol in which the payloads in all blocks are empty. We will extend our protocol to include the payload in Section 10.1. 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.1.

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 $(\textsc{sk}_j, \textsc{pk}_j) \leftarrow \mathsf{uKeyGen}(1^\kappa)$, publishes the public key $\textsc{pk}_j$ and keeps $\textsc{sk}_j$ 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 $B_0 = \langle(\textsc{pk}_1, \textsc{pk}_2, \cdots, \textsc{pk}_n), \mathsf{rand}\rangle$[4].

Algorithm 2: Algorithms Context$^\bullet$, Mining$^\bullet$, Validate$^\bullet$, and D-BestChainSet$^\bullet$.

```
Context•(C):
    $\ell := \mathsf{len}(\mathcal{C})$; $\eta := h(\mathcal{C}[\ell])$; Return $\eta$;
Mining•($\eta, r, \textsc{sk}, \textsc{pk}$):
    $\sigma := \mathsf{uSign}(\textsc{sk}, \langle\eta, r\rangle)$
    if $\mathsf{H}(\eta, r, \textsc{pk}, \sigma) < \mathtt{T}$ then
        Create new block $B := \langle\eta, r, \textsc{pk}, \sigma\rangle$; Return $B$;
    else Return $\perp$
Validate•($\mathcal{C}, r$):
    Parse $\mathcal{C}$ into $B_0 \| B_1 \| \cdots \| B_\ell$;
    for $i \in [1, \ell]$ do
        Parse $B_i$ into $\langle\eta_i, r_i, \textsc{pk}_i, \sigma_i\rangle$;
        if $h(B_{i-1}) \neq \eta_i$ or $\mathsf{H}(\eta_i, r_i, \textsc{pk}_i, \sigma_i) \geq \mathtt{T}$ or $\mathsf{uVerify}(\textsc{pk}_i, \langle\eta_i, r_i\rangle, \sigma_i) = 0$ or $r_i > r$ then
            Return 0;
    Return 1;
$D$-BestChainSet•($\mathbb{C}$):
    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
            $\mathbb{C}_{\text{best}} := \mathbb{C}_{\text{best}} \cup \{\mathcal{C}\}$;
```

Blockchain extension phase. Following the design of a multi-extension framework, our protocol is parameterized by four algorithms: Context$^\bullet$, Mining$^\bullet$, Validate$^\bullet$, and D-BestChainSet$^\bullet$. (Please see Algorithm 2 for the pseudocode of the four algorithms.) 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-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-BestChainSet$^\bullet$ iterates through all chains in the local state and uses algorithm Validate$^\bullet$ to remove the invalid chains. Here, the algorithm 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-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-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 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 $(\textsc{sk}, \textsc{pk})$, 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 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: $\mathsf{H}(\eta, r, \textsc{pk}, \sigma) < \mathtt{T}$, where $\sigma := \mathsf{uSign}(\textsc{sk}, \langle\eta, r\rangle)$. The new block B_{i+1} is defined as $B_{i+1} := \langle\eta, r, \textsc{pk}, \sigma\rangle$.

Algorithm 3: 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 key
        pair $(\textsc{sk}, \textsc{pk})$ 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 $\mathbb{C}_{\text{best}} := D$-BestChainSet•($\mathbb{C}$);
for $\mathcal{C} \in \mathbb{C}_{\text{best}}$ do
    $\eta := \mathsf{Context}^\bullet(\mathcal{C})$; $B := \mathsf{Mining}^\bullet(\eta, r, \textsc{sk}, \textsc{pk})$;
    if $B \neq \perp$ then
        $\mathcal{C}_1 := \mathcal{C} \| B$; Broadcast $\mathcal{C}_1$;
```

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

4.3 A new tiebreak rule for our multi-extension protocol

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.

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 $\mathcal{C}_{\mathrm{best}}$ denote the best chain and $\mathbb{C}_{\mathrm{best}}$ 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 Figure 6, we partition the set $\mathbb{C}_{\mathrm{best}}$ into $D+1$ number of *disjoint subsets* based on the length of those chains. Let $\ell = \mathsf{len}(\mathcal{C}_{\mathrm{best}})$ 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 $\mathbb{C}_{\mathrm{best}}$. That is, $L_i = \{\mathcal{C} \in \mathbb{C}_{\mathrm{best}} : \mathsf{len}(\mathcal{C}) = \ell - i\}$.

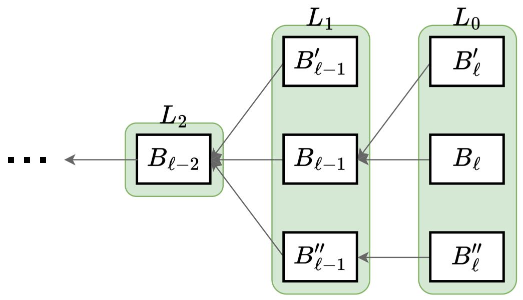


Figure 6: Partitioning a set of best chains $\mathbb{C}_{\mathrm{best}}$ into multiple disjoint depth-based subsets for $D = 2$. Here, the set of best chains $\mathbb{C}_{\mathrm{best}}$ is partitioned into 3 subsets L_0, L_1, L_2. Let ℓ be the length of the best chain. The 0-depth subset L_0 consists 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 $B_{\ell-2}$.

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 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 4 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 $\mathbb{C}_{\mathrm{best}}$ and $\mathbb{C}'_{\mathrm{best}}$ 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 $\mathbb{C}_{\mathrm{best}}$ with length

Algorithm 4: Tiebreak rule

```
Input  : Two chains $\mathcal{C}_{\text{best}}$, $\mathcal{C}'_{\text{best}}$ of length $\ell$, 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 distance($\mathcal{C}_{\text{best}} \to \mathcal{C}$) $\le D$ then
        $\mathbb{C}_{\text{best}} := \mathbb{C}_{\text{best}} \cup \{\mathcal{C}\}$
    if distance($\mathcal{C}'_{\text{best}} \to \mathcal{C}$) $\le D$ then
        $\mathbb{C}'_{\text{best}} := \mathbb{C}'_{\text{best}} \cup \{\mathcal{C}\}$
for $i \in [0..D]$ do
    $L_i := \{\mathcal{C} \in \mathbb{C}_{\text{best}} : \text{len}(\mathcal{C}) = \ell - i\}$
    $L'_i := \{\mathcal{C} \in \mathbb{C}'_{\text{best}} : \text{len}(\mathcal{C}) = \ell - i\}$
$i := 0$
while $i \le D$ 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 $\mathcal{C}_{\text{best}}$
```

$\ell - i$, 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 the overview of security analysis for protocol $\Pi^\bullet$. Then, in Section 6, we propose a new analysis framework to study the chain growth in multi-extension protocols. In Section 7, we use the above analysis framework to examine the chain growth of our protocol. In Section 8, we present a new analysis framework to analyze the common prefix property. Finally, in Section 9, we present the analysis of chain quality and the best possible unpredictability.

As in the previous section, assuming the underlying scheme (uKeyGen, uKeyVer, uSign, uVerify) is a unique digital signature scheme, a malicious player for a given context, can create *exactly one* signature. Now, we can prove the security properties of protocol $\Pi^\bullet$ under the assumption of honest majority of *effective stake* based on $\alpha^\bullet = \hat{\mathtt{A}}^\bullet_D \cdot \alpha$ (e.g., $\hat{\mathtt{A}}^\bullet_{50} \ge 2.04$) and $\beta^\bullet = 2.72\beta$, where $\hat{\mathtt{A}}^\bullet_D$ is the amplification ratio caused by the greedy strategy.

It is important to note that the techniques described in [GKL15, PSs17, DGKR18, BDK$^+$19] can provide valuable insights for analyzing the security properties of protocols based on the single extension design framework. However, our protocol $\Pi^\bullet$ *does not* follow this framework and requires new analysis techniques to prove its security properties.

Theorem 5.1. *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* (uKeyGen, uKeyVer, uSign, uVerify) *is a unique digital signature scheme, and $\alpha^\bullet = \lambda\beta^\bullet$, $\lambda > 1$. Then protocol $\Pi^\bullet$ achieves 1) chain growth, chain quality, and common prefix properties; and 2) the best possible unpredictability.*

Chain growth. We propose a new analysis framework to study the chain growth in multi-extension protocols in Section 6. We develop *a random walk in a Markov chain that consists of multiple states* to analyze the chain growth property. In the Markov chain, each state provides a representation of the set of best chains in a protocol round. Note that, for the existing single-extension protocols [GKL15, PSs17, DGKR18], since the honest players only extend a single best chain, the probability that honest players extend the best chain is the same for every round. Hence, the analysis of chain growth for such protocols is quite simple. On the other hand, in multi-extension protocols, honest players may extend multiple chains in a single round, and the probability of extending the best chain can vary between rounds. Therefore, a new analysis framework is necessary to evaluate the chain growth in multi-extension protocols.

In Section 7, we use the above analysis framework to examine the chain growth of our protocol. Note that, in a multi-extension protocol, the adversary may attempt to slow down the chain growth by launching attacks. Fortunately, as we mentioned in the previous section, our tiebreak rule prevents the adversary from launching such attacks. We start our analysis with the design of a *simplified Markov chain* and then extend it to design an *augmented Markov chain*.

Simplified Markov chain. 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 the amplification ratio, even with a large D.

Augmented Markov chain. 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 on the amplification ratio.

Common prefix. To analyze the common prefix property, we first demonstrate that the adversary can increase their stake by a factor of $e = 2.72$ if they extend the chain themselves. It is worth noting that the adversary cannot use the blocks of honest players to compromise the security property. 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, in our protocol, 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 of the D-distance-greedy strategy, the set of best chains must be "close". That is, these chains share a common prefix after removing the last D blocks. Thus, the chains produced by the 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 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.

Virtual block-sets and virtual chains. 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 few blocks, say D, where D is a parameter as mentioned above. 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.

Common prefix w.r.t. virtual chains. To prove the common prefix w.r.t. virtual chains, 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 a D-greedy strategy that we mentioned above, at each block height, there is at most one honest virtual block-set. Thus, to break the common prefix property 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 stake, which contradicts the assumption that honest players control the majority.

Common prefix. Finally, we demonstrate that the common prefix property can be achieved from the common prefix w.r.t. virtual chains. Recall that, the common prefix w.r.t. virtual chains property 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.

Chain quality. After proving the chain growth and common prefix properties, the proof of chain quality will be very similar to the proof in [PSs17]. Intuitively, to break the chain quality property, the adversary must generate κ consecutive blocks on the best chains. This requires the adversary to control the majority of the stake.

The best possible unpredictability. In our protocol, the players extract the context of a chain based on the hash value of the last block. Thus, 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 she/he can generate the next block. In other words, our protocol achieves the best possible unpredictability.

6 Chain Growth in Multi-Extension: A New Analysis Framework

In this section, 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 Section 6.1 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.

6.1 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 [CLLM12]. 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 6.1 (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* $\mathsf{growth}(s)$.

6.2 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 6.2 (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 $\delta > 0$.*

Proof. Consider a random walk from round r to round r_1. Let s_i denote that state at round i in the random walk, where $i \in [r..r_1]$ and $s_i \in S$. Since the Markov chain $(S, \mathbf{T})$ and the chain growth function $\mathsf{growth} : S \to [0,1]$ that is compatible to protocol Π°, the probability that the players extend the best chain at round i is $\mathsf{growth}(s_i)$. Using the Chernoff bound on the Markov chain in [CLLM12], we have,

$$\Pr[\mathsf{len}(\mathcal{C}_1) - \mathsf{len}(\mathcal{C}) < (1-\delta) \cdot \bar{g} \cdot t] < e^{-\Omega(\kappa)}.$$

□

We remark that our analysis framework can be applied to other multi-extension protocols, beyond the protocol in Section 4.2. 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.

7 Chain Growth in Multi-Extension: Security analysis details

We will now analyze the chain growth property of our protocol using the analysis framework in Section 6. Before constructing the Markov chains for our protocol, we will first consider a hybrid execution in which the malicious players will not contribute to the chain extension. We demonstrate that by utilizing the tiebreak method, honest players will always follow the best chain with the fastest expected time for extension. As a result, the adversary is unable to slow down the growth of the chain for honest players. The chain growth in this hybrid scenario serves as a lower bound for the chain growth in the real execution. We will then construct two Markov chains to analyze the chain growth in the hybrid execution for the protocol.

We start with a simplified Markov chain. Then, we extend the simplified Markov chain to design an augmented Markov chain. In Figure 7, we show the lower bounds of the amplification ratio using the simplified and augmented Markov chains. In Figure 8, we show the corresponding 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, we have $\hat{\mathtt{A}}^{\bullet}_{50} \geq 2.04$, i.e., protocol $\Pi^{\bullet}$ is secure if 57% of stake is honest.

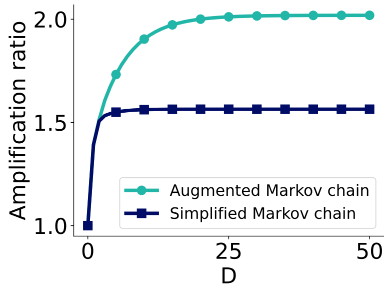


Figure 7: The lower bounds of the amplification ratio using the simplified and augmented Markov chains respectively.

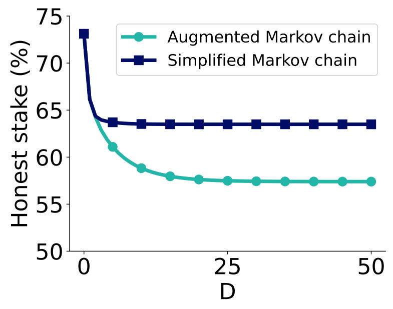


Figure 8: The upper bounds of the fraction on honest players using the simplified and augmented Markov chains respectively.

We facilitate the analysis of the chain growth property for our protocol by defining a new notion, called *amplification ratio*. The amplification ratio is the ratio between the chain growth when the honest players follow D-distance-greedy strategy and 0-distance-greedy strategy. In our protocol, in each round, for each chain in the set of best chains, each honest player makes one attempt to generate a new block by making one query to the random oracle. The event where an honest player successfully generates a new block (from a given chain in the set of best chains) can be modeled as an (independent) Bernoulli random variable which takes the value 1 with probability $p = \frac{\mathtt{T}}{2^{\kappa}}$. Hence, in each round, the probability that an honest player extends a chain in the set of best chains is $\alpha = 1 - (1-p)^{N\cdot(1-\rho)}$, where N is the number of players and ρ is the fraction of malicious players. Let N_0 and N_D be the average increased length of the longest chain that is extended by the honest players, following the 0-distance-greedy, and D-distance-greedy strategies, respectively. In the 0-distance-greedy, since the honest players only extend the best chain, we have, $N_0 = \alpha$. We define the amplification ratio in the presence of an adversary for the D-distance-greedy strategy as $\mathtt{A}^{\bullet}_D = \frac{N_D}{\alpha}$. We can compute the amplification ratio as

$$\hat{\mathtt{A}}^{\bullet}_D = \frac{\bar{g}}{\alpha} = \mathbb{E}_{s\sim Q}\left[\frac{\mathsf{growth}(s)}{\alpha}\right].$$

7.1 A hybrid experiment: Ignoring the adversarial extension

We consider a hybrid experiment where all messages sent by the adversary are removed. Through this experiment, we demonstrate that the adversary cannot slow down the growth of the honest player's chain. We note that hybrid experiments were introduced in the analysis of the Bitcoin protocol in [PSs17].

Let $\mathsf{REAL}(\omega) = \mathsf{EXEC}_{\Pi^\bullet,\mathcal{A},\mathcal{Z}}(\omega)$ denote the standard execution of $\Pi^\bullet$, where ω is the randomness involved in the execution. Let $\mathsf{HYB}^r(\omega)$ denote the hybrid execution, which is identical to the real execution up until round r, with the following modifications: 1) the randomness is fixed to ω, and 2) honest players eliminate all new messages sent by the adversary.

No slow-down tiebreak. First, we show that by using the tiebreak rule in Subsection 4.3, the adversary cannot slow down the chain growth of honest players. More specifically, we will demonstrate that, based on the tiebreak rule, honest players always choose the best chain that can be extended more quickly.

We consider the expected time for the honest players to generate a new best chain by extending a set of best chain $\mathbb{C}_{\text{best}}$. The set of best chains $\mathbb{C}_{\text{best}}$ is partitioned into $D+1$ depth-based subsets $L_0, L_1, \cdots, L_D$. The length of the best chain increases by 1 if a chain in the subset L_0 (i.e., a chain that has the same length as the best chain) is extended. Plus, for $i \in [0..D-1]$, the number of chains in a subset L_i increases by 1 if a chain in the subset L_{i+1} is extended.

Let $\mathsf{w}(n) = 1-(1-p)^{N\cdot n\cdot(1-\rho)}$ be the probability that the honest players generate *at least* one block by extending n blocks in a round. Furthermore, we can calculate the probability of only one honest player generating just one block is

$$\mathsf{w}'(n) = \binom{N\cdot(1-\rho)}{1}\cdot\binom{n}{1}\cdot p\cdot(1-p)^{N\cdot n\cdot(1-\rho)-1}.$$

Since p is sufficiently small (which is negligible in κ), then we have $\mathsf{w}'(n) \approx N\cdot n\cdot(1-\rho)\cdot p \approx w(n) \approx n\cdot\alpha$. In other words, in a single round, the probability of generating more than one block can be safely disregarded because p is small enough.

For $i \in [0..D]$, let $n_i = |L_i|$ be the number of chains in i-depth subset. The probability that the honest players generate a new best chain of length $\ell+1$ is $\mathsf{w}(n_0) \approx n_0\cdot\alpha$. For $i \in [0..D-1]$, the probability that the honest players generate a new chain in L_i is $\mathsf{w}(n_{i+1}) \approx n_{i+1}\cdot\alpha$.

Let $\mathsf{BlockTime}(n_0,\cdots,n_D)$ be the expected time for the honest players to generate a new best chain by extending a set of best chains $\mathbb{C}_{\text{best}}$. By the following lemma, we can show that honest players always choose the best chain that can be extended more quickly.

Lemma 7.1. *We consider two chains, $\mathcal{C}$ and $\mathcal{C}'$ of the same length. Let $\mathbb{C}_{\text{best}}$ and $\mathbb{C}'_{\text{best}}$ be the corresponding sets of best chains for $\mathcal{C}$ and $\mathcal{C}'$, respectively. In other words, the distance from $\mathcal{C}$ to the chains in $\mathbb{C}_{\text{best}}$ is smaller than D, and the distance from $\mathcal{C}'$ to the chains in $\mathbb{C}'_{\text{best}}$ is smaller than D. Here, $\mathbb{C}_{\text{best}}$ is partitioned into $D+1$ depth-based subsets $L_0, L_1, \cdots, L_D$ and $\mathbb{C}'_{\text{best}}$ is partitioned into $D+1$ depth-based subsets $L'_0, L'_1, \cdots, L'_D$. For $i \in [0..D]$, let $n_i = |L_i|$ be the number of chains in L_i and $n'_i = |L'_i|$ be the number of chains in L'_i. If the tiebreak rule in Algorithm 4 return the chain $\mathcal{C}$, then, we have,*

$$\mathsf{BlockTime}(n_0,\cdots,n_D) \le \mathsf{BlockTime}(n'_0,\cdots,n'_D).$$

Proof. If $n_i = n'_i$ for all $i \in [0..D]$, then we have $\mathsf{BlockTime}(n_0,\cdots,n_D) = \mathsf{BlockTime}(n'_0,\cdots,n'_D)$. If there exists a smallest i such that $n_i \ne n'_i$, then according to Algorithm 4, we have $n_i > n'_i$, and $n_j = n'_j$ for all $j \in [0..i-1]$. Next, we will employ induction on i to show that $\mathsf{BlockTime}(n_0,\cdots,n_i,\cdots,n_D) < \mathsf{BlockTime}(n'_0,\cdots,n'_i,\cdots,n'_D)$.

The base case is when $i=0$, meaning $n_0 > n'_0$. The probability that honest players can extend one best chain in $\mathbb{C}_{\text{best}}$ (resp. $\mathbb{C}_{\text{best}}$) after one round is $\mathsf{w}(n_0)$ (resp. $\mathsf{w}(n'_0)$). On the flip side, the probability of not successfully extending in the first round is $1-\mathsf{w}(n_0)$ (resp. $1-\mathsf{w}(n'_0)$). In the second round, the expected value of $|L_0|$ (resp. $|L'_0|$) is $n_0 + w(n_1)\cdot 1 \approx n_0 + n_1\cdot\alpha$ (resp. $n'_0 + n'_1\cdot\alpha$). Therefore, in the second round, the probability that honest parties can successfully extend the longest chain is $(1-\mathsf{w}(n_0))\cdot\mathsf{w}(n_0+n_1\cdot\alpha) \approx (1-n_0\cdot\alpha)(n_0\cdot\alpha+n_1\cdot\alpha^2)$ (resp. $(1-n'_0\cdot\alpha)(n'_0\cdot\alpha+n'_1\cdot\alpha^2)$). Since p is small enough, we can neglect higher-order terms involving α. As a result, in the second round, the probability that honest players can extend a best chain in $\mathbb{C}_{\text{best}}$ (resp. $\mathbb{C}'_{\text{best}}$) is also $\mathsf{w}(n_0)$ (resp. $\mathsf{w}(n'_0)$), which is the same as the first round. For subsequent rounds, we can obtain the same result. Finally, since $n_0 > n'_0$, it follows that $\mathsf{w}(n_0) > \mathsf{w}(n'_0)$, and therefore $\mathsf{BlockTime}(n_0,\cdots,n_D) < \mathsf{BlockTime}(n'_0,\cdots,n'_D)$.

Assuming $\mathsf{BlockTime}(n_0,\cdots,n_i,\cdots,n_D) < \mathsf{BlockTime}(n'_0,\cdots,n'_i,\cdots,n'_D)$ holds for $i=s$ (where s is in $[0..D-1]$), let's proceed to prove that the same conclusion holds for $i=s+1$. In this case, we have $n_j = n'_j$ for all j in the range $[0..s]$, and $n_{s+1} > n'_{s+1}$. We just need to prove that honest players take less time to generate a new chain in L_s than in L'_s. Since $n_{s+1} > n'_{s+1}$, this step is straightforward, as it aligns with the same reasoning presented in the base case proof. Therefore, we conclude this proof. $\square$

Analyzing chain growth in the hybrid experiment. Next, we show that the chain growth rate in the execution $\mathsf{REAL}(\omega)$ is always bigger than or equal to the chain growth rate in the execution $\mathsf{HYB}^r(\omega)$. The tiebreak rule ensures that honest players always follow the best chain with the fastest expected time to extend it. Therefore, if the adversary broadcasts its chain to the honest players, the growth of the best chain will speed up, making it impossible for the adversary to slow down the chain growth of the honest players.

Lemma 7.2. *For all randomness ω and all round r, consider two executions* $\mathsf{REAL}(\omega)$ *and* $\mathsf{HYB}^r(\omega)$*. Consider an honest player P at round r_1, where $r_1 > r$. Let $\mathcal{C}$ be the best chain at round r_1 in the execution* $\mathsf{REAL}(\omega)$*, and let $\mathcal{C}_{\mathsf{hyb}}$ be the best chain at round r_1 in the execution* $\mathsf{HYB}^r(\omega)$*. Then, we have,* $\mathsf{len}(\mathcal{C}) \geq \mathsf{len}(\mathcal{C}_{\mathsf{hyb}})$.

Proof. As demonstrated in Lemma 7.1, the tiebreak rule described in Subsection 4.2 guarantees that honest players always follow the best chain with the fastest expected time for extension. Therefore, if the adversary broadcasts its chain to the honest players, the growth of the best chain will accelerate, making it impossible for the adversary to impede the chain growth of the honest players. □

To analyze the chain growth from round r to round r_1 in the real execution $\mathsf{REAL}(\omega)$, we consider the hybrid execution $\mathsf{HYB}^r(\omega)$. Since the first r rounds in the hybrid execution $\mathsf{HYB}^r(\omega)$ are the same as in the real execution $\mathsf{REAL}(\omega)$, the best chain at round r in both executions is the same. Moreover, as stated in Lemma 7.2, the best chain at round r_1 in the real execution $\mathsf{REAL}(\omega)$ is longer than the best chain in the hybrid execution $\mathsf{HYB}^r(\omega)$. Thus, the chain growth from round r to round r_1 in the real execution $\mathsf{REAL}(\omega)$ is larger than the chain growth in the hybrid execution $\mathsf{HYB}^r(\omega)$.

7.2 Analyzing the chain growth property via a simplified Markov chain

Next, we design Markov chains and a chain growth function to analyze the chain growth in the hybrid execution. In this subsection, we introduce a simplified Markov chain. Then, in Subsection 7.3, we extend the simplified Markov chain to design a more complex augmented Markov chain. Using the augmented Markov chain, we can obtain a tighter bound for chain growth.

We will use the definition of the *depth-based subsets* to design the simplified Markov chain. We show how to use the depth-based subsets to analyze the chain growth in Subsection 7.2.1. Here, each state in the simplified Markov chain contains information about the number of chains in the depth-based subsets. Then, we present a simplified Markov chain to analyze the chain growth property for $D = 1$ in Subsection 7.2.2. Finally, we present a simplified Markov chain to analyze the chain growth property for an arbitrary D in Subsection 7.2.3.

7.2.1 Depth-based subsets in the set of best chains in the execution

We represent the definition of depth-based subsets in the set of best chains. 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. 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, $L_i = \{\mathcal{C} \in \mathbb{C}_{\text{best}} : \mathsf{len}(\mathcal{C}) = \ell - i\}$.

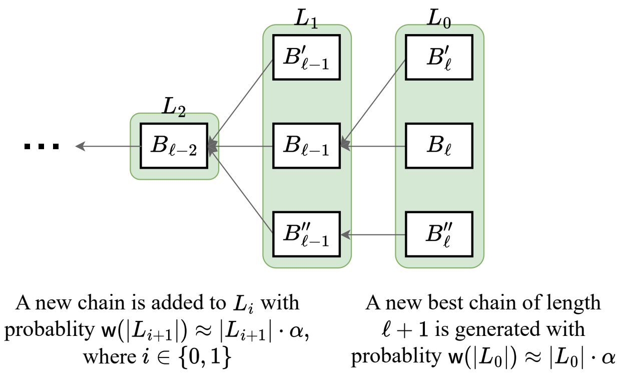

L_1
L_0
L_2
$B'_{\ell-1}$
B'_ℓ
$B_{\ell-2}$
$B_{\ell-1}$
B_ℓ
$B''_{\ell-1}$
B''_ℓ
...
A new chain is added to L_i with probability $\mathsf{w}(|L_{i+1}|) \approx |L_{i+1}| \cdot \alpha$, where $i \in \{0,1\}$
A new best chain of length $\ell+1$ is generated with probability $\mathsf{w}(|L_0|) \approx |L_0| \cdot \alpha$

Figure 9: Partitioning a set of best chains $\mathbb{C}_{\text{best}}$ into multiple disjoint depth-based subsets for $D = 2$. Note that, the set of best chains $\mathbb{C}_{\text{best}}$ here is identical to the one in Figure 6. 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. Here, the probability that honest players generate a new best chain of length $\ell+1$ is $\mathsf{w}(|L_0|)$. The probability that honest players generate a new chain in L_0 is $\mathsf{w}(|L_1|)$. The probability that honest players generate a new chain in L_1 is $\mathsf{w}(|L_2|)$.

Let $n_0 = |L_0|$ be the number of chains in 0-depth subset. The probability that the honest players generate a new best chain of length $\ell + 1$ is $\mathsf{w}(n_0) \approx n_0 \cdot \alpha$ (see Figure 9). Recall that, if the honest players follow the single extension framework and only extend the best chain, then the probability that the honest players generate a new best chain of length $\ell + 1$ is $\mathsf{w}(1) = \alpha$. Here, the amplification ratio $\hat{\mathtt{A}}^{\bullet}_D$ can be estimated by the average number of chains in L_0. More concretely, let $\mathbb{E}[n_0]$ be the expected value of the number of chains n_0 in L_0 and $\mathbb{E}[\mathsf{w}(n_0)]$ be the expected value of the probability that the honest players generate a new best chain. We have, the amplification ratio is

$$\hat{\mathtt{A}}^{\bullet}_D = \frac{\mathbb{E}[\mathsf{w}(n_0)]}{\alpha} \approx \frac{\mathbb{E}[n_0 \cdot \alpha]}{\alpha} = \mathbb{E}[n_0].$$

To analyze the number of chains in L_0, we need to analyze the number of chains in L_1. Similarly, for all $i \in [D-1]$, to analyze the number of chains in L_i, we need to analyze the number of chains in L_{i+1}. Thus, to compute the amplification ratio, we develop a Markov chain that consists of multiple states. The states in the Markov chain represent the number of chains in all depth-based subsets in the set of best chains. Note that, as the subset L_D always has exactly one chain, we omit the representation of $|L_D|$ in the Markov chain state.

7.2.2 The simplified Markov chain for $D = 1$

We describe the simplified Markov chain with the state space S and a transition matrix $\mathbf{T}$ to analyze the chain growth for $D = 1$. Here, each state in S is in the format of $\langle n_0 \rangle$ that represents a protocol round in which a set of best chains in which the number of chains in 0-depth subset is n_0, where $n_0 \in \mathbb{N}$. The state space is given as $S = \{\langle n_0 \rangle\}_{n_0 \in \mathbb{N}}$. More concretely, consider the set of best chains that is partitioned into two subsets L_0 and L_1. Let ℓ be the length of the current best chain. Let $n_0 = |L_0|$ and $n_1 = |L_1|$ be the numbers of chains in L_0 and L_1, respectively. Note that, the number of chains in L_1 always equals 1, i.e., $n_1 = 1$. Thus, we omit the representation of $|L_1|$ in the states of the Markov chain.

The transition matrix $\mathbf{T}$ is an $|S| \times |S|$ matrix that contains information on the probability of transitioning between states. We construct the transition matrix $\mathbf{T}$ as follows. Initially, we set all the values in $\mathbf{T}$ to 0. Then, for each state $\langle n_0 \rangle$, we update the transition matrix based on the following cases of transitions (see Figure 10).

Case 1: *A new best chain of length $\ell + 1$ is generated.* In other words, a chain of length ℓ in L_0 is extended. The probability that the honest players generate at least one new best chain of length $\ell + 1$ is $\mathsf{w}(n_0)$, where n_0 is the number of chains in L_0. Recall that the extension of more than one chain in a round is negligible. In this case, the new 0-depth subset of the new set of best chains only consists of one chain, i.e., the new best 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 of length ℓ is generated (and Case 1 does not happen).* In other words, a chain of length $\ell - 1$ in L_1 is extended. As the number of chains in L_1 is one, the probability that the honest players generate a new chain of length ℓ 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 $\mathbf{T}_{\langle n_0 \rangle, \langle n_0+1 \rangle} := \mathsf{w}(1) \cdot (1 - \mathbf{T}_{\langle n_0 \rangle, \langle 1 \rangle}) \approx \mathsf{w}(1)$.

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 $\langle n_0 \rangle$. Similar to Equation 1, for all $s \in S$, we have, $\sum_{s \in S} q_s = 1$ and $q_s = \sum_{s' \in S} q_{s'} \cdot \mathbf{T}_{s',s}$. We define the chain growth function $\mathsf{growth} : S \to [0,1]$ such that $\mathsf{growth}(n_0) = \mathsf{w}(n_0) \approx n_0 \cdot \alpha$. The amplification ratio $\hat{\mathtt{A}}^{\bullet}_1$ is computed as the expected number of chains in L_0 in each round, i.e.,

$$\hat{\mathtt{A}}^{\bullet}_1 = \sum_{n_0=1}^{\infty} \left(q_{n_0} \cdot \frac{\mathsf{growth}(n_0)}{\alpha} \right) = \sum_{n_0=1}^{\infty} q_{n_0} \cdot n_0.$$

Based on the equation, we have $\hat{\mathtt{A}}^{\bullet}_1 = 1.39$.

7.2.3 The simplified Markov chain for a general D

We now describe the simplified Markov chain with the state space S and a transition matrix $\mathbf{T}$ to analyze the chain growth for arbitrary D. Consider a protocol round in which the set of best chains is $\mathbb{C}_{\text{best}}$. For $i \in [0..D]$, let L_i be the i-depth subset in $\mathbb{C}_{\text{best}}$ and $n_i = |L_i|$ be the number of chains in L_i. In the simplified Markov chain, we use the state $\langle n_0, \cdots, n_{D-1} \rangle$ to represent the protocol round with the set of best chains $\mathbb{C}_{\text{best}}$. Note that, since the subset L_D always has exactly one chain, i.e., $n_D = 1$, we do not include the number of chains in L_D in the representation of the states. The state space of the simplified Markov chain is given as $S = \{\langle n_0, \cdots, n_{D-1} \rangle\}_{n_i \in \mathbb{N}, \forall i \in [0..D-1]}$.

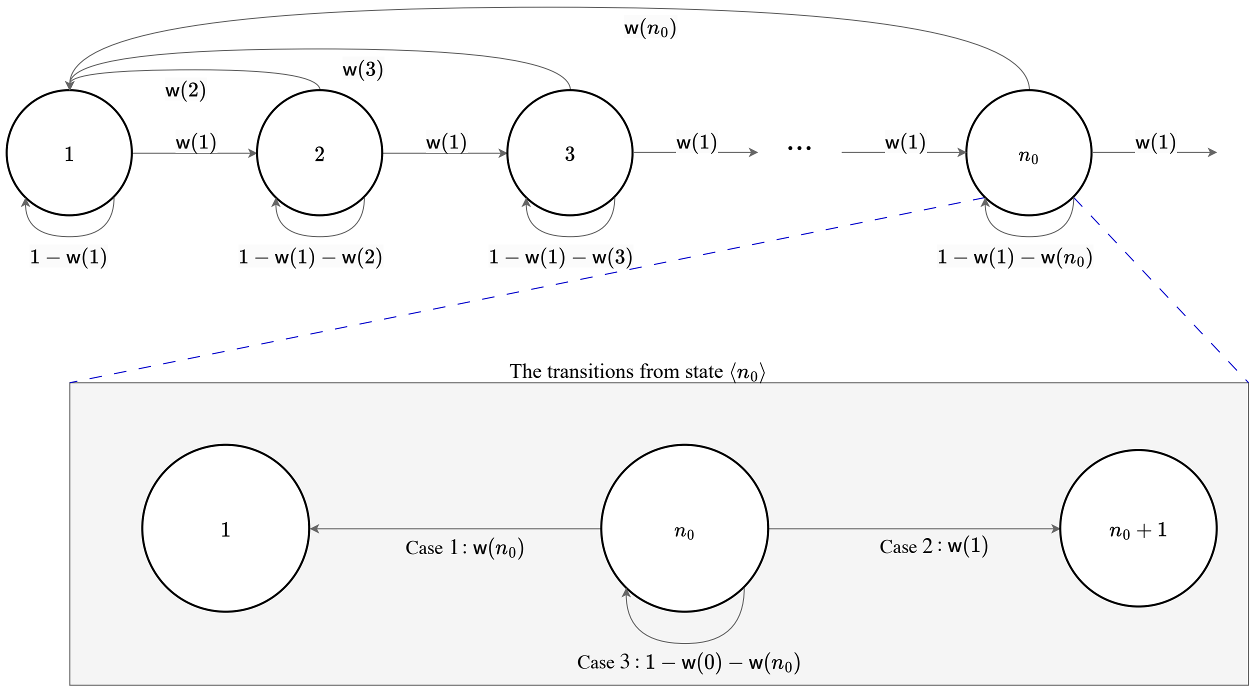
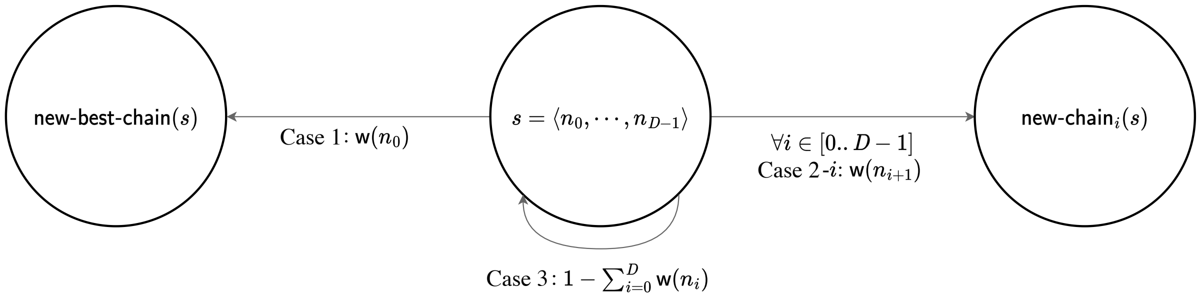


Figure 10: The complete state machine for the simplified Markov chain for $D = 1$. In the 1-distance-greedy strategy, the set of best chains can be partitioned into two depth-based subsets L_0 and L_1. Since there is only one chain in L_1, each state $\langle n_0 \rangle$ only indicates the number of chains in L_0, which is n_0. For each state $\langle n_0 \rangle$, we have the following cases of transitions. *Case 1: A new best chain of length $\ell + 1$ is generated.* In other words, a chain of length ℓ in L_0 is extended. The probability that the honest players generate a new best chain of length $\ell + 1$ is $\mathsf{w}(n_0)$, where n_0 is the number of chains in L_0. In this case, the new 0-depth subset only consists of one chain, i.e., the new best chain. The state machine moves from state $\langle n_0 \rangle$ to state $\langle 1 \rangle$. *Case 2: A new chain of length ℓ is generated (and Case 1 does not happen).* In other words, a chain of length $\ell - 1$ in L_1 is extended. As the number of chains in L_1 is one, the probability that the honest players generate a new chain of length ℓ is $\mathsf{w}(1)$. In this case, the state machine moves from state $\langle n_0 \rangle$ to state $\langle n_0 + 1 \rangle$. *Case 3: No new chain is generated.* The state machine remains at the current state n_0 with a probability of $1 - \mathsf{w}(1) - \mathsf{w}(n_0)$.

The state of the simplified Markov chain provides information about the set of best chains in each round. After each round, the state machine transitions to a new state depending on updates in the set of best chains. We categorize the transitions in the simplified Markov chain based on how the set of the best chains is updated as follows.

- *Case 1: A new best chain is generated, i.e., the length of the best chain increases by* 1. As a result, the new best chain is added to the set of best chains, while some existing chains may be removed. Based on the number of chains in the depth-based subsets, we cannot know how many chains are removed. Thus, we consider the worst-case scenario where the set of best chains only consists of the best chain and its prefixes. The state machine moves to the new state in which the number of each depth-based subset contains only one block.
- *Case 2: A new chain is generated but the length of the best chain remains unchanged.* In this case, a chain will be added to a depth-based subset. The remaining depth-based subsets will remain the same. The state machine moves to a new state in which the number of chains in the updated depth-based subset increases by one.
- *Case 3: No chain is generated.* The set of best chains remains unchanged. The state machine remains in the same state.



new-best-chain(s)
$s = \langle n_0, \cdots, n_{D-1}\rangle$
new-chain$_i(s)$
Case 1: $\mathsf{w}(n_0)$
$\forall i \in [0..D-1]$
Case 2-i: $\mathsf{w}(n_{i+1})$
Case 3: $1 - \sum_{i=0}^{D} \mathsf{w}(n_i)$

Figure 11: The transitions from state $s = \langle n_0, \cdots, n_{D-1}\rangle$ in the state machine of a simplified Markov chain for a general D. Here, the state represents the protocol round in which the set of best chains is partitioned into $D+1$ subsets $L_0, L_1, \cdots, L_D$ and $i \in [0..D-1]$, the number of chains in L_i is $|L_i| = n_i$. Note that, the number of chains in L_D always equals 1, i.e., $n_D = 1$. From state $\langle n_0, \cdots, n_{D-1}\rangle$, we have three cases of transitions as follows. *Case 1: A new best chain is generated, i.e., the length of the best chain increases by* 1. In this case, a chain in the subset L_0 is extended. The probability that the honest players generate a new best chain of length $\ell + 1$ is $\mathsf{w}(n_0)$, where n_0 is the number of chains in the subset L_0. This case is the generalization of Case 1 in Figure 10. In this case, we consider the worst-case scenario where the set of best chains only consists of the best chain and its prefixes, meaning each depth-based subset contains only one block. This is because based on the information on depth-based subsets, we cannot determine how many chains will be removed from the set of best chains if the distance from the new best chains to those chains is greater than D. We define the function new-best-chain : $S \to S$ that takes as input a state in S and outputs an updated state when a new best chain is generated. In other words, if a new best chain is generated, the state machine moves from state $\langle n_0, \cdots, n_{D-1}\rangle$ to state new-best-chain$(n_0, \cdots, n_{D-1})$. Next, we show the definition of function new-best-chain. For a state $\langle n_0, \cdots, n_{D-1}\rangle \in S$, let $\langle n'_0, \cdots, n'_{D-1}\rangle =$ new-best-chain$(n_0, \cdots, n_{D-1})$. For all $i' \in [0..D-1]$, we have, $n'_{i'} = 1$. For example, with $D = 1$, we have, new-best-chain$(n_0) = \langle 1\rangle$. *Case 2: A new chain is generated but the length of the best chain remains the same.* Here, a player generates a chain $\mathcal{C}$ such that len$(\mathcal{C}) \leq \ell$. The chain $\mathcal{C}$ can be added to the i-depth subset of $\mathbb{C}_{\text{best}}$, where $i \in [0..D-1]$. This case is the generalization of Case 2 in Figure 10. Based on the depth of the new chain, we divide this case into D sub-cases as follows. For any $i \in [0..D-1]$, we consider the sub-case 2-i in which the new chain is added to subset L_i. In other words, a chain in subset L_{i+1} is extended. The probability of such an event is $\mathsf{w}(n_{i+1})$, where n_{i+1} is the number of chains in the subset L_{i+1}. For each $i \in [0..D-1]$, we define the function new-chain$_i : S \to S$ that takes as input a state in S and outputs an updated state when a new chain is added to the i-depths subset L_i. In other words, if a new chain is added to L_i, the state machine moves from state $\langle n_0, \cdots, n_{D-1}\rangle$ to state new-chain$_i(n_0, \cdots, n_{D-1})$. Next, we show the definition of function new-chain$_i$. For a state $\langle n_0, \cdots, n_{D-1}\rangle \in S$, let $\langle n'_0, \cdots, n'_{D-1}\rangle =$ new-chain$_i(n_0, \cdots, n_{D-1})$. We have, $n'_{i'} = n_{i'}, \forall i' \neq i$ and $n'_{i'} = n_{i'} + 1$ if $i' = i$. For example, with $D = 1$, we have, new-chain$_0(n_0) = \langle n_0 + 1\rangle$. *Case 3: No chain is generated.* In this case, the state machine remains at the current state $\langle n_0, \cdots, n_{D-1}\rangle$ with a probability of $1 - \sum_{i=0}^{D} \mathsf{w}(n_i)$.

The transition matrix $\mathbf{T}$ is constructed as follows. First, all values in $\mathbf{T}$ are initially set to 0. Then, for each state $\langle n_0, \cdots, n_{D-1}\rangle \in \mathbf{T}$, the transition matrix is updated based on the three cases of transitions (see Figure 11).

Case 1: A new best chain is generated, i.e., the length of the best chain increases by 1. In this case, a chain in the subset L_0 is extended. The probability that the honest players generate a new best chain of length $\ell + 1$ is $\mathsf{w}(n_0)$, where n_0 is the number of chains in the subset L_0. This case is the generalization of Case 1 in Figure 10. In this case, we consider the worst-case scenario where the set of best chains only consists of the best chain and its prefixes, meaning each depth-based subset contains only one block. This is because based on the information on depth-based subsets, we cannot determine how many chains will be removed from the set of best chains if the distance from the new best chains to those chains is greater than D. We define the function $\mathsf{new\text{-}best\text{-}chain} : S \to S$ that takes as input a state in S and outputs an updated state when a new best chain is generated. In other words, if a new best chain is generated, the state machine moves from state $\langle n_0, \cdots, n_{D-1}\rangle$ to state $\mathsf{new\text{-}best\text{-}chain}(n_0, \cdots, n_{D-1})$. Next, we show the definition of function $\mathsf{new\text{-}best\text{-}chain}$. For a state $\langle n_0, \cdots, n_{D-1}\rangle \in S$, let $\langle n'_0, \cdots, n'_{D-1}\rangle = \mathsf{new\text{-}best\text{-}chain}(n_0, \cdots, n_{D-1})$. We have, $n'_{i'} = 1, \forall i' \in [0..D-1]$. For example, with $D = 1$, we have, $\mathsf{new\text{-}best\text{-}chain}(n_0) = \langle 1\rangle$. We set $\mathbf{T}_{\langle n_0,\cdots,n_{D-1}\rangle,\mathsf{new\text{-}best\text{-}chain}(n_0,\cdots,n_{D-1})} := \mathsf{w}(n_0)$.

<u>Case 2</u>: *A new chain is generated but the length of the best chain remains unchanged.* Here, a player generates a chain $\mathcal{C}$ such that $\mathsf{len}(\mathcal{C}) \leq \ell$. The chain $\mathcal{C}$ can be added to one of the i-depth subsets L_i in the set of best chains $\mathbb{C}_{\text{best}}$, where $i \in [0..D-1]$. This case is the generalization of Case 2 in Figure 10. Based on the depth of the new chain, we divide this case into D sub-cases as follows. For any $i \in [0..D-1]$, we consider the sub-case 2-i in which the new chain is added to the subset L_i. In other words, a chain in subset L_{i+1} is extended. The probability of such an event is $\mathsf{w}(n_{i+1})$, where n_{i+1} is the number of chains in the subset L_{i+1}. For each $i \in [0..D-1]$, we define the function $\mathsf{new\text{-}chain}_i : S \to S$ that takes as input a state in S and outputs an updated state when a new chain is added to the i-depths subset L_i. In other words, if a new chain is added to L_i, the state machine moves from state $\langle n_0, \cdots, n_{D-1}\rangle$ to state $\mathsf{new\text{-}chain}_i(n_0, \cdots, n_{D-1})$. Next, we show the definition of function $\mathsf{new\text{-}chain}_i$. For a state $\langle n_0, \cdots, n_{D-1}\rangle \in S$, let $\langle n'_0, \cdots, n'_{D-1}\rangle = \mathsf{new\text{-}chain}_i(n_0, \cdots, n_{D-1})$. We have, $n'_{i'} = n_{i'}, \forall i' \neq i$ and $n'_{i'} = n_{i'} + 1$ if $i' = i$. For example, with $D = 1$, we have, $\mathsf{new\text{-}chain}_0(n_0) = \langle n_0 + 1\rangle$. (Here, we do not consider the possibility of multiple sub-cases happening at the same time. Take the instance of sub-cases 2-i and 2-j, where $i \neq j$; the probability of both happening together in one round is $\mathsf{w}(n_{i+1}) \cdot \mathsf{w}(n_{j+1}) \approx n_{i+1} \cdot n_{j+1} \cdot \alpha^2$, and we can ignore higher-order terms involving α.) We set $\mathbf{T}_{\langle n_0,\cdots,n_{D-1}\rangle,\mathsf{new\text{-}chain}_i(n_0,\cdots,n_{D-1})} := \mathsf{w}(n_{i+1}) \cdot \prod_{i' \in [0..D], i' \neq i+1}(1 - \mathsf{w}(n_{i'})) \approx \mathsf{w}(n_{i+1})$.

<u>Case 3</u>: *No chain is generated*. In this case, the state machine remains at the current state $\langle n_0, \cdots, n_{D-1}\rangle$ with a probability of $1 - \sum_{i=0}^{D} \mathsf{w}(n_i)$. We set $\mathbf{T}_{\langle n_0,\cdots,n_{D-1}\rangle,\langle n_0,\cdots,n_{D-1}\rangle} := 1 - \sum_{i=0}^{D} \mathsf{w}(n_i)$.

Let $q_{n_0,\cdots,n_{D-1}}$ be the stationary probability of the state $\langle n_0, \cdots, n_{D-1}\rangle$. Similar to Equation 1, we have,

$$\begin{cases} \sum_{n_0=1}^{\infty} \cdots \sum_{n_{D-1}=1}^{\infty} q_{n_0,\cdots,n_{D-1}} = 1, \\ q_{n_0,\cdots,n_{D-1}} = \sum_{\langle n'_0,\cdots,n'_{D-1}\rangle \in S} \left(q_{n'_0,\cdots,n'_{D-1}} \cdot \mathbf{T}_{\langle n'_0,\cdots,n'_{D-1}\rangle,\langle n_0,\cdots,n_{D-1}\rangle} \right). \end{cases} \tag{2}$$

We define the chain growth function $\mathsf{growth} : S \to [0, 1]$ such that $\mathsf{growth}(n_0, \cdots, n_{D-1}) = \mathsf{w}(n_0) \approx n_0 \cdot \alpha$. The amplification ratio $\hat{\mathtt{A}}^{\bullet}_D$ equals the expected number of chains in L_0 in each round, i.e.,

$$\begin{aligned} \hat{\mathtt{A}}^{\bullet}_D &= \sum_{n_0=1}^{\infty} \cdots \sum_{n_{D-1}=1}^{\infty} \left(q_{n_0,\cdots,n_{D-1}} \cdot \frac{\mathsf{growth}(n_0, \cdots, n_{D-1})}{\alpha} \right) \\ &= \sum_{n_0=1}^{\infty} \cdots \sum_{n_{D-1}=1}^{\infty} \left(q_{n_0,\cdots,n_{D-1}} \cdot n_0 \right). \end{aligned}$$

Using the simplified Markov chain, we can find a lower bound of the amplification ratio as shown in Figure 7. For $D = 50$, we can find a lower bound $\hat{\mathtt{A}}^{\bullet}_{50} \geq 1.56$.

7.3 Analyzing the chain growth via an augmented Markov chain

In the simplified Markov chain, the state only provides the information on the number of chains in the depth-based subsets. Some critical information is omitted due to this simple representation. Indeed, when a new best chain is generated and some of the chains are removed from the set of best chains, we have to consider the worst-case scenario. Hence, we cannot establish a good lower bound of the amplification ratio, *even when D is big!* For example, with $D = 50$, using the simplified Markov chain, we can only guarantee an amplification of 1.56. Here, we present an **augmented** Markov chain in which *each state contains more information on the set of best chains*. With the augmented Markov chain, we can find a *better* lower bound of the amplification ratio.

Before presenting the augmented Markov chain, we introduce the notion of *depth-distance-based subsets* in Subsection 7.3.1. The chains in a depth-distance-based subset are selected based on *both* the length of those chains *and* the distance from the best chain to those chains. Each state in the augmented Markov chain contains information about the number of chains in each depth-distance-based subset. With this information, we can determine the number of chains removed from the set of best chains when a new longest chain is generated. This avoids having to consider the worst-case scenario where each depth-based subset contains only one chain, as in the simplified Markov chain.

We first present an augmented Markov chain to analyze the chain growth for $D = 2$ in Subsection 7.3.2 and then extend the augmented Markov chain for an arbitrary D int Subsection 7.3.3.

7.3.1 Depth-distance-based subsets in the set of best chains in the execution

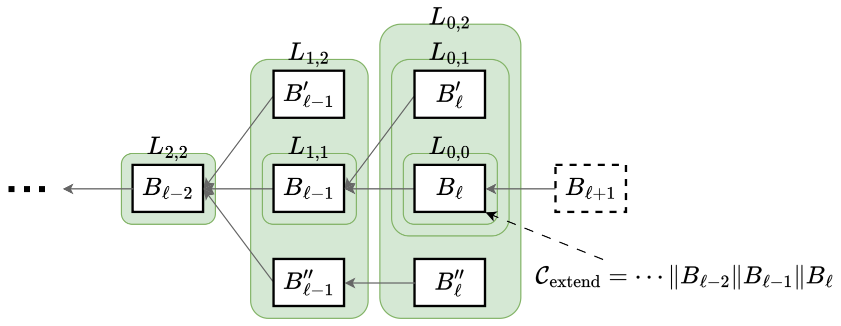


Figure 12: The depth-distance-based subsets of the set of best chains $\mathbb{C}_{\text{best}}$ for $D = 2$. Recall that, in Figure 9, the set of best chains $\mathbb{C}_{\text{best}}$ is partitioned into 3 disjoint depth-based subsets L_0, L_1, L_2. Let $\mathcal{C}_{\text{extend}}$ be the first chain in L_0 that is extended. In this figure, $\mathcal{C}_{\text{extend}} = \cdots \| B_{\ell-1} \| B_{\ell-1} \| B_{\ell}$. (In some future round, a new best chain is generated by adding a block $B_{\ell+1}$ to $\mathcal{C}_{\text{extend}}$.) Consider a i-depth subset L_i, where $i \in [0..D]$. We further define multiple subsets of chains based on the distance from $\mathcal{C}_{\text{extend}}$ to those chains in L_i. More concretely, for $i \in [0..D]$, $j \in [i..D]$, the "i-depth j-distance" subset $L_{i,j}$ consists of all the chains in the i-depth subset L_i such that the distance from the chain $\mathcal{C}_{\text{extend}}$ to those chains does not exceed j, i.e., $L_{i,j} = \{\mathcal{C} \in L_i : \mathsf{distance}(\mathcal{C}_{\text{extend}} \rightarrow \mathcal{C}) \leq j\}$. For example, the "0-depth 0-distance" $L_{0,0}$ consists of 1 chain that has the last block is B_ℓ. The "0-depth 1-distance" $L_{0,1}$ consists of 2 chains that have the last blocks are B_ℓ, B'_ℓ. The "0-depth 2-distance" $L_{0,2}$ consists of 3 chains that have the last blocks are $B_\ell, B'_\ell, B''_\ell$.

We introduce the notion of depth-distance-based subsets. Let $\mathcal{C}_{\text{extend}}$ be the first chain in L_0 that is extended. We remark that the chain $\mathcal{C}_{\text{extend}}$ can be changed when the set of best chain $\mathbb{C}_{\text{best}}$ is updated. Indeed, when a chain $\mathcal{C}'$ is added to subset L_0, the chain $\mathcal{C}'$ can be extended earlier than the current chain $\mathcal{C}_{\text{extend}}$. In this case, we update $\mathcal{C}_{\text{extend}} = \mathcal{C}'$. For $i \in [0..D]$, we define multiple subsets of the i-depth subset L_i as follows. For $j \in [0..D]$, the "i-depth j-distance" subset $L_{i,j}$ is the set of chains in the i-subset L_i such that the distance from $\mathcal{C}_{\text{extend}}$ to those chains does not exceed j, i.e., $L_{i,j} = \{\mathcal{C} \in L_i : \mathsf{distance}(\mathcal{C}_{\text{extend}} \rightarrow \mathcal{C}) \leq j\}$ (see Figure 12 for an example of depth-distance-based subsets). Here, for all $j < i$, $L_{i,j} = \emptyset$ and $L_{i,i}$ consists of exact 1 chain, i.e., the prefix of $\mathcal{C}_{\text{extend}}$ with the length $\ell - i$, where $\ell = \mathsf{len}(\mathcal{C}_{\text{extend}})$. Further, $L_{i,j}$ is a subset of $L_{i,j+1}$, i.e., $L_{i,j} \subseteq L_{i,j+1}$. In other words, $L_{i,i} \subseteq L_{i,i+1} \subseteq \cdots \subseteq L_{i,D}$.

When the chain $\mathcal{C}_{\text{extend}}$ is extended, a new best chain is added and some chains are removed from the set of best chain (see Figure 13 for an example). Let $\mathcal{C}'_{\text{best}}$ be the new best chain and $\mathbb{C}'_{\text{best}}$ be the new set of best chain. Let L'_i be the i-depth subset of the set of best chains $\mathbb{C}'_{\text{best}}$, i.e., $L'_i = \{\mathcal{C} \in \mathbb{C}'_{\text{best}} : \mathsf{len}(\mathcal{C}) = \ell + 1 - i\}$. The subset L'_0 only consists of one best chain $\mathcal{C}'_{\text{best}}$. For $i \in [1..D]$ and $j \in [i..D]$, the "i-depth j-distance" subset of $\mathbb{C}'_{\text{best}}$ can be obtained by the depth-distance-based subsets of $\mathbb{C}_{\text{best}}$ in Figure 12. Indeed, for any chain $\mathcal{C}$, we have, $\mathsf{distance}(\mathcal{C}_{\text{extend}} \rightarrow \mathcal{C}) = \mathsf{distance}(\mathcal{C}'_{\text{best}} \rightarrow \mathcal{C}) - 1$. Thus, $\mathbb{C}'_{\text{best}} = \{\mathcal{C} : \mathsf{distance}(\mathcal{C}'_{\text{best}} \rightarrow \mathcal{C}) \leq D\} = \{\mathcal{C} : \mathsf{distance}(\mathcal{C}_{\text{extend}} \rightarrow \mathcal{C}) \leq D - 1\}$. Hence, $L'_{i,j} = \{\mathcal{C} : \mathsf{len}(\mathcal{C}) = \ell + 1 - i \wedge \mathsf{distance}(\mathcal{C}_{\text{extend}} \rightarrow \mathcal{C}) \leq j - 1\} = L_{i-1,j-1}$.

7.3.2 The augmented Markov chain for $D = 2$

We will now design an augmented Markov chain to analyze chain growth. The augmented Markov chain keeps track of the number of chains in $L_{i,j}$ for different values of i and j, providing a way to analyze chain growth. The states in the augmented Markov chain capture information about the number of chains in each $L_{i,j}$, where $i \in [0..D]$ and $j \in [0..D]$. Note that for all $i \in [1..D]$, we have $|L_{i,j}| = 0$ for $j \in [0..i-1]$ and $|L_{i,i}| = 1$. Thus, for $i \in [0..D]$ and $j \in [0..i]$, we omit the representation of $L_{i,j}$.

For $D = 1$, the augmented Markov chain is equivalent to the simplified Markov chain. Therefore, we will first design an augmented Markov chain for $D = 2$. Then, we will extend the design for a general D.

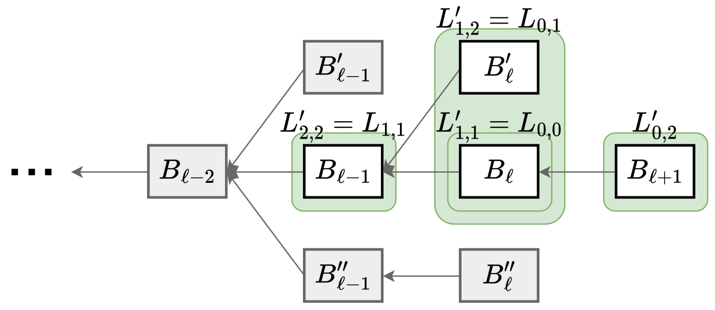


Figure 13: The new set of best chains $\mathbb{C}'_{\text{best}}$ when a new block is added on $\mathcal{C}_{\text{extend}}$. Here, the set of best chains $\mathbb{C}'_{\text{best}}$ consists of the chains in which the last blocks of those chains are $B_{\ell-1}, B_\ell, B'_\ell, B_{\ell+1}$. The chains in which the last block of those chains are $B_{\ell-2}, B'_{\ell-1}, B''_{\ell-1}, B''_\ell$ belongs to $\mathbb{C}_{\text{best}}$ but not $\mathbb{C}'_{\text{best}}$. For $i \in [0..D], j \in [i..D]$, let $L'_{i,j}$ be the "i-depth j-distance" subset of the set of best chains $\mathbb{C}'_{\text{best}}$, i.e., $L'_i = \{\mathcal{C} \in \mathbb{C}'_{\text{best}} : \mathsf{len}(\mathcal{C}) = \ell + 1 - i\}$. The subset L'_0 only consists of the best chain $\mathcal{C}'_{\text{best}} = \cdots \| B_{\ell-2} \| B_{\ell-1} \| B_\ell \| B_{\ell+1}$. For $i \in [1..D]$, the i-depth subset of $\mathbb{C}'_{\text{best}}$ can be obtained by the depth-distance-based subsets of $\mathbb{C}_{\text{best}}$ in Figure 12. Indeed, $L'_{i,j} = L_{i-1,j-1}$, where $L_{i-1,j-1}$ is the "$(i-1)$-depth $(j-1)$-distance" subset of $\mathbb{C}_{\text{best}}$.

We describe the augmented Markov chain with the state space $\hat{S}$ and a transition matrix $\hat{\mathbf{T}}$ to analyze the chain growth for $D = 2$. Consider a round with the set of best chains $\mathbb{C}_{\text{best}}$, let $\mathcal{C}_{\text{extend}}$ be the first chain in L_0 that is extended. For $i, j \in [0..2]$, let $L_{i,j}$ be the "i-depth j-distance" subset of the set of best chain $\mathbb{C}_{\text{best}}$, i.e., $L_{i,j} = \{\mathcal{C} \in L_i : \mathsf{distance}(\mathcal{C}_{\text{extend}} \to \mathcal{C}) \leq j\}$. Let $n_{i,j} = |L_{i,j}|$ be the numbers of chains in $L_{i,j}$. In the augmented Markov chain, the state $\langle (n_{0,1}, n_{0,2}), (n_{1,2}) \rangle$ represents the protocol round with the set of best chains $\mathbb{C}_{\text{best}}$. (Note that, for $i \in [0..D]$ and $j \in [0..i]$, we omit the representation of $L_{i,j}$ since the numbers of chains in those subsets are the same for every state.) The state space of the augmented Markov is given as $\hat{S} = \{\langle (n_{0,1}, n_{0,2}), (n_{1,2}) \rangle\}_{n_{0,1}, n_{0,2}, n_{1,2} \in \mathbb{N}}$. Here, each state in $\hat{S}$ is in the format of $\langle (n_{0,1}, n_{0,2}), (n_{1,2}) \rangle$ that represents a set of best chains such that the numbers of chains in "0-depth 1-distance", "0-depth 2-distance", and "1-depth 2-distance" subsets are $n_{0,1}, n_{0,2}, n_{1,2}$, respectively. The state space is given as $\hat{S} = \{\langle (n_{0,1}, n_{0,2}), (n_{1,2}) \rangle\}_{n_{0,1}, n_{0,2}, n_{1,2} \in \mathbb{N}}$.

Recall that, the transition matrix $\hat{\mathbf{T}}$ is an $|\hat{S}| \times |\hat{S}|$ matrix that contains information on the probability of transitioning between states. We construct the transition matrix $\hat{\mathbf{T}}$ as follows. Initially, we set all the values in $\hat{\mathbf{T}}$ to 0. Then, for each state $\langle (n_{0,1}, n_{0,2}), (n_{1,2}) \rangle$, we update the transition matrix based on the following cases of transitions (see Figure 14). Furthermore, we assume that in each round, state transitions belong to one and only one of the following cases. While situations where multiple cases occur at the same round may exist, similar to the analysis of the simplified Markov chain, we can overlook the probabilities of these situations.

<u>Case 1</u>: *A new best chain is generated, i.e., the length of the best chain increases by* 1. A new best chain is generated if a chain in the subset L_0 is extended. As $L_0 = L_{0,2}$, the number of chains in L_0 is $n_{0,2}$. Thus, with a probability of $\mathsf{w}(n_{0,2})$, a new best chain is generated. After the new chain is generated, the 0-depth subset only consists of the best chain. Thus, the number of chains in "0-depth 1-distance" subset and "0-depth 2-distance" subset are the same and equal 1. The number of chains in "1-depth 2-distance" subset is $n_{0,1}$. Thus, the state machine moves from state $\langle (n_{0,1}, n_{0,2}), (n_{1,2}) \rangle$ to state $\langle (1, 1), (n_{0,1}) \rangle$. We set $\hat{\mathbf{T}}_{\langle (n_{0,1}, n_{0,2}), (n_{1,2}) \rangle, \langle (1,1), (n_{0,1}) \rangle} := \mathsf{w}(n_{0,2})$.

<u>Case 2</u>: *A new chain $\mathcal{C}$ is added to subset $L_{1,2}$.* A chain $\mathcal{C}$ is added to subset $L_{1,2}$ if a chain in $L_{2,2}$ is extended. As the number of chains in the subset $L_{2,2}$ is 1, the probability of this case is $\mathsf{w}(1)$. In this case, the state machine moves from state $\langle (n_{0,1}, n_{0,2}), (n_{1,2}) \rangle$ to state $\langle (n_{0,1}, n_{0,2}), (n_{1,2} + 1) \rangle$. In fact, similar to the analysis of the simplified Markov chain, here we also need to consider the probability of Case 1 not occurring. However, this probability can be neglected. (In the subsequent case analyses, we will omit such explanations.) We set $\hat{\mathbf{T}}_{\langle (n_{0,1}, n_{0,2}), (n_{1,2}) \rangle, \langle (n_{0,1}, n_{0,2}), (n_{1,2}+1) \rangle} := \mathsf{w}(1)$.

<u>Case 3</u>: *A new chain $\mathcal{C}$ is added to subset $L_{0,1}$ and the chain $\mathcal{C}$ is the first chain that is extended in the subset L_0.* A new chain $\mathcal{C}$ is added to subset $L_{0,1}$ if a chain in subset $L_{1,1}$ is extended. As the number of chains in the subset $L_{1,1}$ is 1, the probability that a new chain $\mathcal{C}$ is added to subset $L_{0,1}$ is $\mathsf{w}(1)$. The probability that the chain $\mathcal{C}$ is the first chain that is extended in the subset L_0 is $\frac{1}{n_{0,2}+1}$. The probability of this event is $\frac{1}{n_{0,2}+1}$. Due to the change of $\mathcal{C}_{\text{extend}}$, we consider the worst-case scenario here, where the size of $L_{0,1}$ becomes 1. In this case, the state machine moves from state $\langle (n_{0,1}, n_{0,2}), (n_{1,2}) \rangle$ to state $\langle (1, n_{0,2} + 1), (n_{1,2}) \rangle$. We set $\hat{\mathbf{T}}_{\langle (n_{0,1}, n_{0,2}), (n_{1,2}) \rangle, \langle (1, n_{0,2}+1), (n_{1,2}) \rangle} := \frac{\mathsf{w}(1)}{n_{0,2}+1}$.

<u>Case 4</u>: *A new chain $\mathcal{C}$ is added to subset $L_{0,1}$ and the chain $\mathcal{C}$ is not the first chain that is extended in the subset L_0.* As

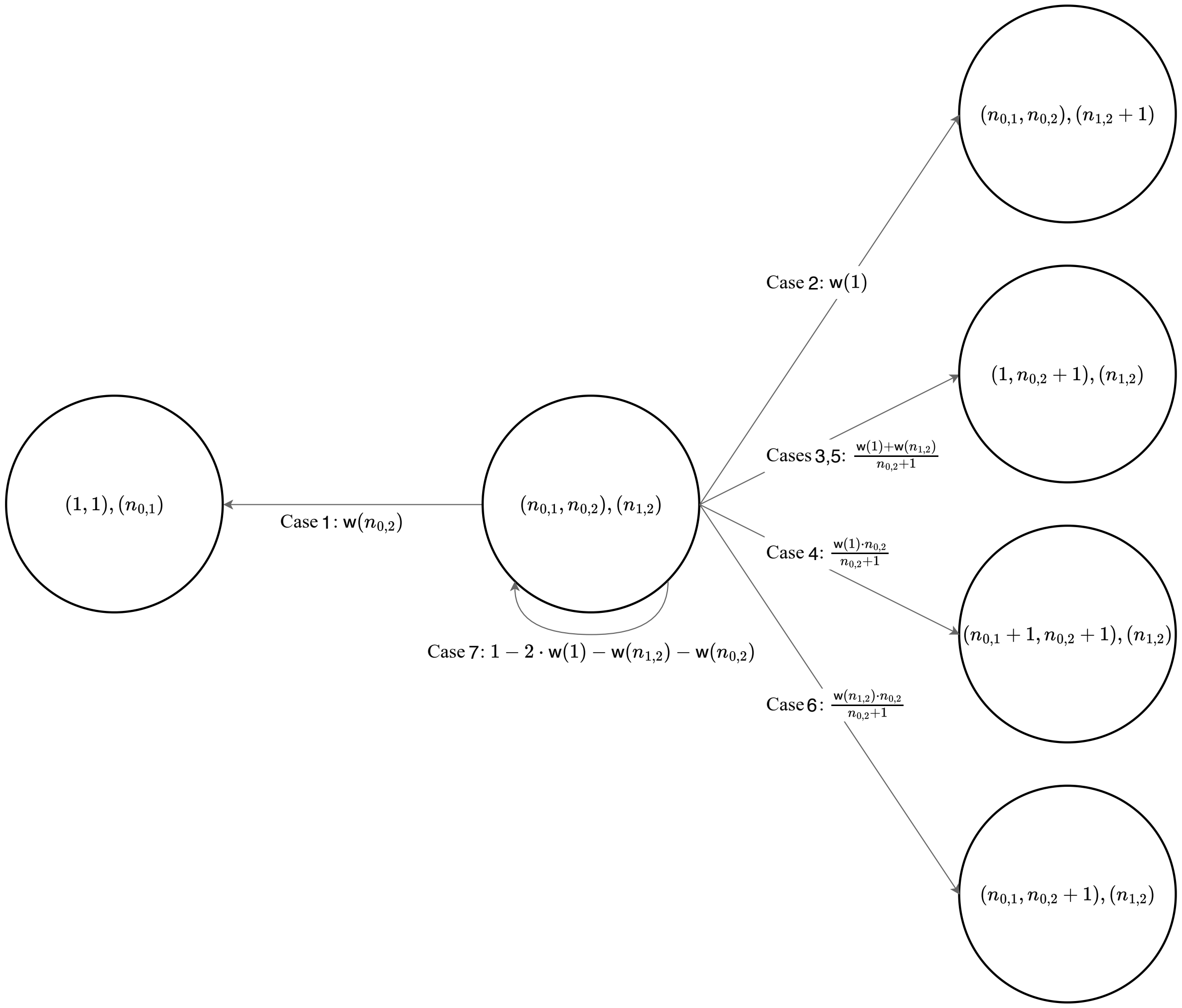

$(n_{0,1}, n_{0,2}), (n_{1,2}+1)$
Case 2: $\mathsf{w}(1)$
$(1, n_{0,2}+1), (n_{1,2})$
Cases 3,5: $\frac{\mathsf{w}(1)+\mathsf{w}(n_{1,2})}{n_{0,2}+1}$
$(1,1), (n_{0,1})$
Case 1 : $\mathsf{w}(n_{0,2})$
$(n_{0,1}, n_{0,2}), (n_{1,2})$
Case 4: $\frac{\mathsf{w}(1)\cdot n_{0,2}}{n_{0,2}+1}$
$(n_{0,1}+1, n_{0,2}+1), (n_{1,2})$
Case 7: $1 - 2\cdot \mathsf{w}(1) - \mathsf{w}(n_{1,2}) - \mathsf{w}(n_{0,2})$
Case 6 : $\frac{\mathsf{w}(n_{1,2})\cdot n_{0,2}}{n_{0,2}+1}$
$(n_{0,1}, n_{0,2}+1), (n_{1,2})$

Figure 14: The transitions from a state $\langle (n_{0,1}, n_{0,2}), (n_{1,2})\rangle$ in the state machine for $D = 2$. We have the following cases of transitions. *Case 1: A new best chain is generated, i.e., the length of the best chain increases by* 1. A new best chain is generated if a chain in the subset L_0 is extended. As $L_0 = L_{0,2}$, the number of chains in L_0 is $n_{0,2}$. Thus, with a probability of $\mathsf{w}(n_{0,2})$, a new best chain is generated. After the chains are generated, the 0-depth subset only consists of the best chain. Thus, the number of chains in "0-depth 1-distance" subset and "0-depth 2-distance" subset are the same and equal 1. The number of chains in "1-depth 2-distance" subset is $n_{0,1}$. Thus, the state machine moves from state $\langle (n_{0,1}, n_{0,2}), (n_{1,2})\rangle$ to state $\langle (1,1), (n_{0,1})\rangle$. *Case 2: A new chain $\mathcal{C}$ is added to subset $L_{1,2}$.* A chain $\mathcal{C}$ is added to subset $L_{1,2}$ if a chain in $L_{2,2}$ is extended. As the number of chains in the subset $L_{2,2}$ is 1, the probability of this case is $\mathsf{w}(1)$. In this case, the state machine moves from state $\langle (n_{0,1}, n_{0,2}), (n_{1,2})\rangle$ to state $\langle (n_{0,1}, n_{0,2}), (n_{1,2}+1)\rangle$. *Case 3: A new chain $\mathcal{C}$ is added to subset $L_{0,1}$ and the chain $\mathcal{C}$ is the first chain that is extended in the subset L_0.* A new chain $\mathcal{C}$ is added to subset $L_{0,1}$ if a chain in subset $L_{1,1}$ is extended. As the number of chains in the subset $L_{1,1}$ is 1, the probability that a new chain $\mathcal{C}$ is added to subset $L_{0,1}$ is $\mathsf{w}(1)$. The probability that the chain $\mathcal{C}$ is the first chain that is extended in the subset L_0 is $\frac{1}{n_{0,2}+1}$. The probability of this event is $\frac{1}{n_{0,2}+1}$. In this case, the state machine moves from state $\langle (n_{0,1}, n_{0,2}), (n_{1,2})\rangle$ to state $\langle (1, n_{0,2}+1), (n_{1,2})\rangle$. *Case 4: A new chain $\mathcal{C}$ is added to subset $L_{0,1}$ and the chain $\mathcal{C}$ is not the first chain that is extended in the subset L_0.* As we show in Case 3, the probability that a new chain $\mathcal{C}$ is added to subset $L_{0,1}$ is $\mathsf{w}(1)$. Given that a new chain $\mathcal{C}$ is added to subset $L_{0,1}$, the probability that the chain $\mathcal{C}$ is not the first chain that is extended in the subset L_0 is $\frac{n_{0,2}}{n_{0,2}+1}$. In this case, the state machine moves from state $\langle (n_{0,1}, n_{0,2}), (n_{1,2})\rangle$ to state $\langle (n_{0,1}+1, n_{0,2}+1), (n_{1,2})\rangle$. *Case 5: A new chain is added to subset $L_{0,2}$ and the chain $\mathcal{C}$ is the first chain that is extended in the subset L_0.* A new chain is added to subset $L_{0,2}$ if a chain in the subset $L_{1,2}$ is extended. As the number of chains in the subset $L_{1,2}$ is $n_{1,2}$, the probability that a new chain $\mathcal{C}$ is added to subset $L_{0,2}$ is $\mathsf{w}(n_{1,2})$. The probability that the chain $\mathcal{C}$ is the first chain that is extended in the subset L_0 is $\frac{1}{n_{0,2}+1}$. In this case, the state machine moves from state $\langle (n_{0,1}, n_{0,2}), (n_{1,2})\rangle$ to state $\langle (1, n_{0,2}+1), (n_{1,2})\rangle$. Recall that, we already added a transition from state $\langle (n_{0,1}, n_{0,2}), (n_{1,2})\rangle$ to state $\langle (1, n_{0,2}+1), (n_{1,2})\rangle$ in Case 3. *Case 6: A new chain is added to subset $L_{0,2}$ and the chain $\mathcal{C}$ is not the first chain that is extended in the subset L_2.* As we show in Case 5, the probability that a new chain $\mathcal{C}$ is added to subset $L_{0,2}$ is $\mathsf{w}(n_{1,2})$. Given that a new chain $\mathcal{C}$ is added to subset $L_{0,1}$, the probability that the chain $\mathcal{C}$ is not the first chain that is extended in the subset L_0 is $\frac{n_{0,2}}{n_{0,2}+1}$. In this case, the state machine moves from state $\langle (n_{0,1}, n_{0,2}), (n_{1,2})\rangle$ to state $\langle (n_{0,1}, n_{0,2}+1), (n_{1,2})\rangle$. *Case 7: No new chain is generated.* With a probability of $1 - 2\cdot \mathsf{w}(1) - \mathsf{w}(n_{0,2}) - \mathsf{w}(n_{0,2})$, the state machine remains at the current state $\langle (n_{0,1}, n_{0,2}), (n_{1,2})\rangle$.

we show in Case 3, the probability that a new chain $\mathcal{C}$ is added to subset $L_{0,1}$ is $\mathsf{w}(1)$. Given that a new chain $\mathcal{C}$ is added to subset $L_{0,1}$, the probability that the chain $\mathcal{C}$ is not the first chain that is extended in the subset L_0 is $\frac{n_{0,2}}{n_{0,2}+1}$. In this case, the state machine moves from state $\langle(n_{0,1}, n_{0,2}), (n_{1,2})\rangle$ to state $\langle(n_{0,1}+1, n_{0,2}+1), (n_{1,2})\rangle$. We set $\hat{\mathbf{T}}_{\langle(n_{0,1},n_{0,2}),(n_{1,2})\rangle,\langle(n_{0,1}+1,n_{0,2}+1),(n_{1,2})\rangle} := \frac{n_{0,2}\cdot\mathsf{w}(1)}{n_{0,2}+1}$.

Case 5: *A new chain is added to subset $L_{0,2}$ and the chain $\mathcal{C}$ is the first chain that is extended in the subset L_0.* A new chain is added to subset $L_{0,2}$ if a chain in the subset $L_{1,2}$ is extended. As the number of chains in the subset $L_{1,2}$ is $n_{1,2}$, the probability that a new chain $\mathcal{C}$ is added to subset $L_{0,2}$ is $\mathsf{w}(n_{1,2})$. The probability that the chain $\mathcal{C}$ is the first chain that is extended in the subset L_0 is $\frac{1}{n_{0,2}+1}$. In this case, the state machine moves from state $\langle(n_{0,1}, n_{0,2}), (n_{1,2})\rangle$ to state $\langle(1, n_{0,2}+1), (n_{1,2})\rangle$. Recall that, we already added a transition from state $\langle(n_{0,1}, n_{0,2}), (n_{1,2})\rangle$ to state $\langle(1, n_{0,2}+1), (n_{1,2})\rangle$ in Case 3. Thus, we set $\hat{\mathbf{T}}_{\langle(n_{0,1},n_{0,2}),(n_{1,2})\rangle,\langle(1,n_{0,2}+1),(n_{1,2})\rangle} := \frac{\mathsf{w}(n_{1,2})}{n_{0,2}+1} + \frac{\mathsf{w}(1)}{n_{0,2}+1}$.

Case 6: *A new chain is added to subset $L_{0,2}$ and the chain $\mathcal{C}$ is not the first chain that is extended in the subset L_2.* As we show in Case 5, the probability that a new chain $\mathcal{C}$ is added to subset $L_{0,2}$ is $\mathsf{w}(n_{1,2})$. Given that a new chain $\mathcal{C}$ is added to subset $L_{0,1}$, the probability that the chain $\mathcal{C}$ is not the first chain that is extended in the subset L_0 is $\frac{n_{0,2}}{n_{0,2}+1}$. In this case, the state machine moves from state $\langle(n_{0,1}, n_{0,2}), (n_{1,2})\rangle$ to state $\langle(n_{0,1}, n_{0,2}+1), (n_{1,2})\rangle$. We set $\hat{\mathbf{T}}_{\langle(n_{0,1},n_{0,2}),(n_{1,2})\rangle,\langle(n_{0,1},n_{0,2}+1),(n_{1,2})\rangle} := \frac{n_{0,2}\cdot\mathsf{w}(n_{1,2})}{n_{0,2}+1}$.

Case 7: *No new chain is generated.* With a probability of $1 - 2\cdot\mathsf{w}(1) - \mathsf{w}(n_{0,2}) - \mathsf{w}(n_{0,2})$, the state machine remains at the current state $\langle(n_{0,1}, n_{0,2}), (n_{1,2})\rangle$. We set $\hat{\mathbf{T}}_{\langle(n_{0,1},n_{0,2}),(n_{1,2})\rangle,\langle(n_{0,1},n_{0,2}),(n_{1,2})\rangle} := 1 - 2\cdot\mathsf{w}(1) - \mathsf{w}(n_{0,2}) - \mathsf{w}(n_{0,2})$.

Let $q_{(n_{0,1},n_{0,2}),(n_{1,2})}$ be the stationary probability of the state $\langle(n_{0,1}, n_{0,2}), (n_{1,2})\rangle$. Similar to Equation 1, we have,

$$\begin{cases} \sum_{s\in\hat{S}} q_s = 1, \\ q_s = \sum_{s'\in\hat{S}} q_{s'}\cdot\hat{\mathbf{T}}_{s',s}, \forall s\in\hat{S}. \end{cases} \tag{3}$$

The equations in Equation 3 are equivalent to the following.

$$\begin{cases} \sum_{n_{0,1}=1}^{\infty}\sum_{n_{0,2}=1}^{\infty}\sum_{n_{1,2}=1}^{\infty} q_{(n_{0,1},n_{0,2}),(n_{1,2})} = 1, \\ 3\cdot\mathsf{w}(1)\cdot q_{(1,1),(n_{1,2})} = \sum_{n_{2,1}=1}^{\infty}\sum_{n_{2,2}=1}^{\infty} q_{(n_{0,1},n_{0,2}),(n_{1,2})}, \\ (2\cdot\mathsf{w}(1)+\mathsf{w}(n_{0,2}))\cdot q_{(n_{0,1},n_{0,2}),(n_{1,2})} = q_{(n_{0,1}-1,n_{0,2}-1),(n_{1,2})}\cdot\mathsf{w}(1), \\ (\mathsf{w}(1)+\mathsf{w}(n_{1,2})+\mathsf{w}(n_{0,1}))\cdot q_{(1,n_{0,2}),(n_{1,2})} = q_{(n_{0,1},n_{0,2})-1,(n_{1,2})}\cdot\mathsf{w}(1) + q_{(n_{0,1},n_{0,2}),(n_{1,2})}\cdot\mathsf{w}(n_1). \end{cases} \tag{4}$$

We define the chain growth function $\mathsf{growth} : \hat{S} \to [0,1]$ such that $\mathsf{growth}((n_{0,1}, n_{0,2}), (n_{1,2})) = \mathsf{w}(n_0) \approx n_0\cdot\alpha$. The amplification ratio $\hat{\mathtt{A}}_2^\bullet$ equals the expected number of chains in L_0 in each round, i.e.,

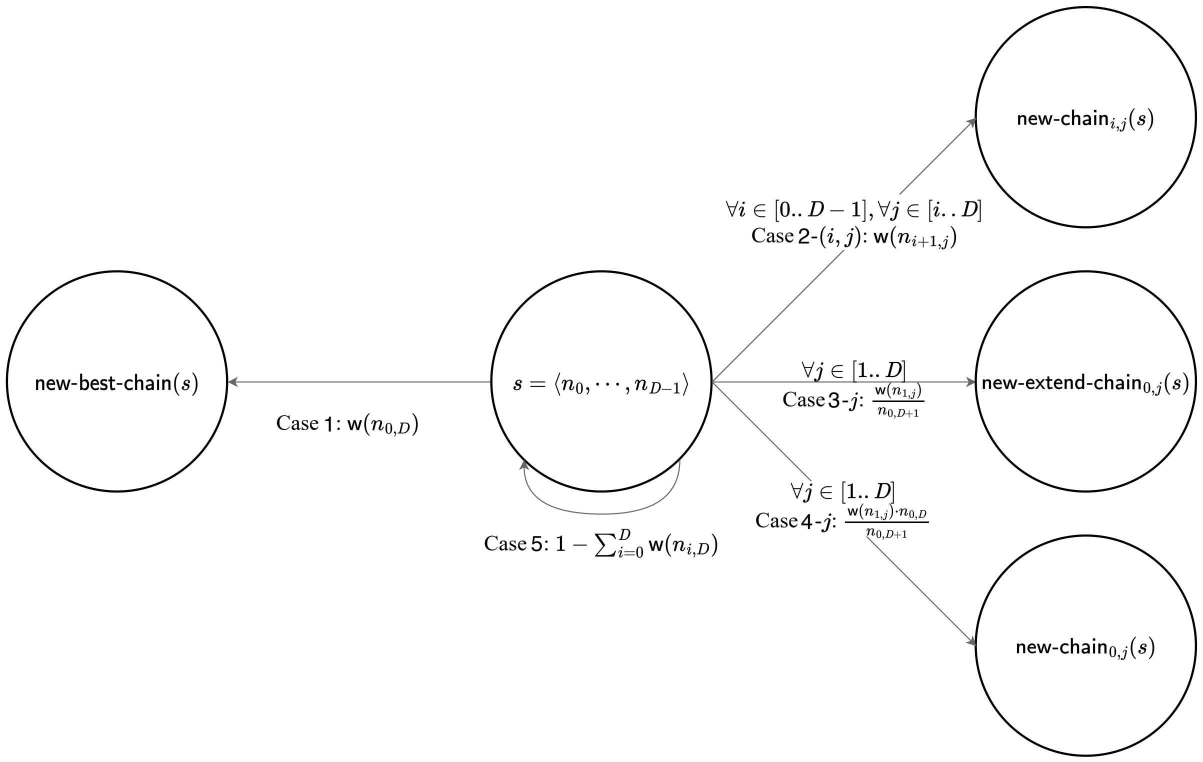
$$\begin{aligned} \hat{\mathtt{A}}_2^\bullet &= \sum_{n_{0,1}=1}^{\infty}\sum_{n_{0,2}=1}^{\infty}\sum_{n_{1,2}=1}^{\infty}\left(q_{(n_{0,1},n_{0,2}),(n_{1,2})}\cdot\frac{\mathsf{growth}((n_{0,1},n_{0,2}),(n_{1,2}))}{\alpha}\right) \\ &= \sum_{n_{0,1}=1}^{\infty}\sum_{n_{0,2}=1}^{\infty}\sum_{n_{1,2}=1}^{\infty}\left(q_{(n_{0,1},n_{0,2}),(n_{1,2})}\cdot n_{0,2}\right). \end{aligned}$$

Combining with Equation 4, we have, $\hat{\mathtt{A}}_2^\bullet \approx 1.51$.

7.3.3 The augmented Markov chain for a general D

We now describe the augmented Markov chain with the state space $\hat{S}$ and a transition matrix $\hat{\mathbf{T}}$ to analyze the chain growth for a general D. Consider a round with the set of best chains $\mathbb{C}_{\mathsf{best}}$. Let L_i be the i-depth subset in $\mathbb{C}_{\mathsf{best}}$, where $i \in [0..D]$. Let $\mathcal{C}_{\mathsf{extend}}$ be the first chain in L_0 that is extended. For $i \in [0..D]$ and $j \in [0..D]$, let $L_{i,j}$ be the "i-depth j-distance" subset of the set of best chains $\mathbb{C}_{\mathsf{best}}$, i.e., $L_{i,j} = \{\mathcal{C} \in L_i : \mathsf{distance}(\mathcal{C}_{\mathsf{extend}} \to \mathcal{C}) \le j\}$. Let $n_{i,j} = |L_{i,j}|$ be the number of chains in $L_{i,j}$. In the augmented Markov chain, the state $\langle(n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D})\rangle$ represents the protocol round with the set of best chains $\mathbb{C}_{\mathsf{best}}$. (We remark that, for $i \in [0..D]$ and $j \in [0..i]$, we do not include the number of chains in $L_{i,j}$ since they are the same for every state.) Hence, the state space is given as $\hat{S} = \{\langle(n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D})\rangle\}_{n_{i,j}\in\mathbb{N},\forall i\in[0..D-1],j\in[i+1..D]}$.

Similar to the simplified Markov chain, the state of the augmented Markov chain represents the information about the set of best chains in each round. We categorize the transitions in the simplified Markov chain based on how the set of the best chains is updated as follows.



new-chain$_{i,j}(s)$
$\forall i \in [0..D-1], \forall j \in [i..D]$
Case 2-(i,j): $\mathsf{w}(n_{i+1,j})$
new-best-chain(s)
$s = \langle n_0, \cdots, n_{D-1}\rangle$
$\forall j \in [1..D]$
Case 3-j: $\frac{\mathsf{w}(n_{1,j})}{n_{0,D+1}}$
new-extend-chain$_{0,j}(s)$
Case 1: $\mathsf{w}(n_{0,D})$
$\forall j \in [1..D]$
Case 4-j: $\frac{\mathsf{w}(n_{1,j})\cdot n_{0,D}}{n_{0,D+1}}$
Case 5: $1 - \sum_{i=0}^{D} \mathsf{w}(n_{i,D})$
new-chain$_{0,j}(s)$

Figure 15: The transition from state $s = \langle (n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D})\rangle$ in the state machine for a general D. We have the following cases of transitions. *Case 1: A new best chain is generated, i.e., the length of the best chain increases by* 1. This case is generalized from Case 1 in Figure 14. A new best chain is generated if a chain in the 0-depth subset L_0 is extended. As the number of chains in the subset L_0 is $n_0 = n_{0,D}$, the probability of this event is $\mathsf{w}(n_{0,D})$. We define the function new-best-chain : $\hat{S} \to \hat{S}$ that takes as input a state in $\hat{S}$ and outputs an updated state when a new best chain is generated. In other words, if a new best chain is generated, the state machine moves from state $\langle (n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D})\rangle$ to state new-best-chain$((n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D}))$. *Case 2: A new chain that is shorter than the current best chain is generated.* Based on the depth and the distance of the new chain $\mathcal{C}$ to the chain $\mathcal{C}_{\text{extend}}$, we consider the following sub-cases. For $i \in [1..D-1], j \in [i+1..D]$, we consider the sub-case 2-(i,j) where the new chain $\mathcal{C}$ is added to "i-depth j-distance" subset $L_{i,j}$. In other words, a player extends a chain in "$(i+1)$-depth j-distance" subset $L_{i+1,j}$. As the number of chains in $L_{i+1,j}$ is $n_{i+1,j}$, the probability of such event is $\mathsf{w}(n_{i+1,j})$. This case is generalized from Case 2 in Figure 14. We define a function new-chain$_{i,j}$: $\hat{S} \to \hat{S}$ that takes as input a state in $\hat{S}$ and outputs an updated state when a new chain is added to the subset $L_{i,j}$. In other words, if a new chain is added to $L_{i,j}$, the state machine moves from state $\langle (n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D})\rangle$ to state new-chain$_{i,j}((n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D}))$. *Case 3: A new chain $\mathcal{C}$, that has the same length as the current best chain, is generated, and the new chain $\mathcal{C}$ is not the first chain in subset L_0 that is extended.* Based on the distance of the new chain $\mathcal{C}$ to the chain $\mathcal{C}_{\text{extend}}$, we consider D sub-cases as follows. For $j \in [1..D]$, we consider a sub-case 3-j in which the new chain $\mathcal{C}$ is added to "0-depth j-distance" subset $L_{0,j}$. In other words, a player extends a chain in "1-depth j-distance" subset $L_{1,j}$. As the number of chains in $L_{1,j}$ is $n_{1,j}$, the probability of such event is $\mathsf{w}(n_{1,j})$. Given that a new chain $\mathcal{C}$ is added to "0-depth j-distance" subset $L_{0,j}$, the probability that $\mathcal{C}$ is not the first chain in subset L_0 that is extended is $\frac{n_{0,D}}{n_{0,D}+1}$. In this case, the state machine moves from state $\langle (n_{0,1}, \cdots, n_{0,D}), \cdots, n_{i,D}), \cdots, (n_{D-1,D})\rangle$ to state new-chain$_{0,j}((n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D}))$. Here the function new-chain$_{0,j}$ is defined as in Case 2. *Case 4: A new chain $\mathcal{C}$, that has the same length as the current best chain, is generated, and the new chain $\mathcal{C}$ is the first chain in subset L_0 that is extended.* Similar to Case 3, for $j \in [1..D]$, we consider a sub-case 4-j in which the new chain $\mathcal{C}$ is added to "0-depth j-distance" subset $L_{0,j}$. Given that a new chain $\mathcal{C}$ is added to "0-depth j-distance" subset $L_{0,j}$, the probability that $\mathcal{C}$ is the first chain in subset L_0 that is extended is $\frac{1}{n_{0,D}+1}$. We define a function new-extend-chain$_{0,j}$: $\hat{S} \to \hat{S}$ that takes as input a state in $\hat{S}$ and outputs an updated state when a new chain $\mathcal{C}$ is added to the "0-depth j-distance" subset $L_{i,j}$ and the chain $\mathcal{C}$ is the first chain in subset L_0 that is extended. If such event happens, the state machine moves from state $\langle (n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D})\rangle$ to state new-extend-chain$_{0,j}((n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D}))$. *Case 5: No new chain is generated.* This case is generalized from Case 7 in Figure 14. The probability of this event is $1 - \sum_{i=0}^{D} \mathsf{w}(n_{i,D})$. The state machine remains at the same state $\langle (n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D})\rangle$.

- *Case 1: A new best chain is generated, i.e., the length of the best chain increases by* 1. The new best chain has been added and some of the chains have been removed from the set of best chains. The chain $\mathcal{C}_{\text{extend}}$ has been updated to become the new best chain. The state machine has moved to a new state where the number of chains in distance-depth-based subsets is determined based on the number of chains in distance-depth-based subsets in the current state.
- *Case 2: A new chain that is shorter than the current best chain is generated.* In this case, a chain will be added to a distance-depth-based subset and all its supersets. The remaining distance-depth-based subsets will remain the same. As a result, the state machine will transition to a new state in which the number of chains in the updated depth-based subsets has increased by one.
- *Case 3: A new chain that has the same length as the current best chain is generated and added to L_0, and the new chain is not the first chain in subset L_0 that is extended.* In this case, the updates on the set of best chains are similar to Case 2. A chain will be added to a distance-depth-based subset and its supersets that are subsets of the 0-depth subset. The state machine moves to the new state in which the number of chains in the updated depth-based subsets increases by one.
- *Case 4: A new chain that has the same length as the current best chain is generated and added to L_0, and the new chain is the first chain in subset L_0 that is extended.* In this case, the new chain is added to the set of best chains without removing any chains. Additionally, the chain $\mathcal{C}_{\text{extend}}$ is updated as the new chain. The state machine transitions to a new state in which the number of chains in distance-depth-based subsets is updated based on the new value of $\mathcal{C}_{\text{extend}}$.
- *Case 5: No chain is generated.* The set of best chains remains unchanged. The state machine remains at the same state.

Let $\hat{\mathbf{T}}$ be an $|\hat{S}| \times |\hat{S}|$ transition matrix that contains information on the probability of transitioning between states. We construct the transition matrix $\hat{\mathbf{T}}$ as follows. Initially, we set all the values in $\hat{\mathbf{T}}$ to 0. Then, for each state $s = \langle (n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D}) \rangle$, we update the transition matrix based on the following cases of transitions (see Figure 15).

Case 1: A new best chain is generated, i.e., the length of the best chain increases by 1. This case is a generalization of Case 1 in Figure 14. A new best chain is generated if a chain in the 0-depth subset L_0 is extended. As the number of chains in the subset L_0 is $n_{0,D}$, the probability of this event is $\mathsf{w}(n_{0,D})$. We define the function $\mathsf{new\text{-}best\text{-}chain} : \hat{S} \to \hat{S}$ that takes a state in $\hat{S}$ as input and outputs an updated state when a new best chain is generated. In other words, if a new best chain is generated, the state machine moves from state $s = \langle (n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D}) \rangle$ to state $\mathsf{new\text{-}best\text{-}chain}(s)$. Let $\langle (n'_{0,1}, \cdots, n'_{0,D}), \cdots, (n'_{D-1,D}) \rangle = \mathsf{new\text{-}best\text{-}chain}(s)$. We have, $n'_{0,j'} := 1$ for all $j' \in [1..D]$; and $n'_{i',j'} := n_{i'-1,j'-1}$ for all $i \in [1..D-1], j' \in [i'+1..D]$. We set $\hat{\mathbf{T}}_{s,\mathsf{new\text{-}best\text{-}chain}(s)} := \mathsf{w}(n_{0,D})$.

Case 2: A new chain $\mathcal{C}$ that is shorter than the current best chain is generated. Based on the depth and the distance of the new chain $\mathcal{C}$ to the chain $\mathcal{C}_{\text{extend}}$, we consider the following sub-cases. For $i \in [1..D-1], j \in [i+1..D]$, we consider the sub-case 2-(i,j) where the new chain $\mathcal{C}$ is added to "i-depth j-distance" subset $L_{i,j}$. In other words, a player extends a chain in "$(i+1)$-depth j-distance" subset $L_{i+1,j}$. As the number of chains in $L_{i+1,j}$ is $n_{i+1,j}$, the probability of such event is $\mathsf{w}(n_{i+1,j})$. This case is generalized from Case 2 in Figure 14. We define a function $\mathsf{new\text{-}chain}_{i,j} : \hat{S} \to \hat{S}$ that takes as input a state in $\hat{S}$ and outputs an updated state when a new chain is added to the subset $L_{i,j}$. In other words, if a new chain is added to $L_{i,j}$, the state machine moves from state $\langle (n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D}) \rangle$ to state $\mathsf{new\text{-}chain}_{i,j}((n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D}))$.
Let $\langle (n'_{0,1}, \cdots, n'_{0,D}), \cdots, (n'_{D-1,D}) \rangle = \mathsf{new\text{-}chain}_{i,j}((n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D}))$. We have, $n'_{i',j'} := n_{i',j'}$ for all $i' \neq i, j' \in [i'+1..D]$ or $i' = i, j' \in [i+1..j-1]$; and $n'_{i,j'} := n_{i,j'} + 1$ for all $j' \in [j..D]$. We set $\hat{\mathbf{T}}_{s,\mathsf{new\text{-}chain}_{i,j}(s)} := \mathsf{w}(n_{i+1,j})$.

Case 3: A new chain $\mathcal{C}$, that has the same length as the current best chain, is generated, and the new chain $\mathcal{C}$ is not the first chain in subset L_0 that is extended. Based on the distance of the new chain $\mathcal{C}$ to the chain $\mathcal{C}_{\text{extend}}$, we consider D sub-cases as follows. For $j \in [1..D]$, we consider a sub-case 3-j in which the new chain $\mathcal{C}$ is added to "0-depth j-distance" subset $L_{0,j}$. In other words, a player extends a chain in "1-depth j-distance" subset $L_{1,j}$. As the number of chains in the subset $L_{1,j}$ is $n_{1,j}$, the probability of such event is $\mathsf{w}(n_{1,j})$. Given that a new chain $\mathcal{C}$ is added to "0-depth j-distance" subset $L_{0,j}$, the probability that $\mathcal{C}$ is not the first chain in subset L_0 that is extended is $\frac{n_{0,D}}{n_{0,D}+1}$. In this case, the state machine moves from state $\langle (n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D}) \rangle$ to state $\mathsf{new\text{-}chain}_{0,j}((n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D}))$. Here, function $\mathsf{new\text{-}chain}_{0,j}$ is defined as in Case 2. Let $\langle (n'_{0,1}, \cdots, n'_{0,D}), \cdots, (n'_{D-1,D}) \rangle = \mathsf{new\text{-}chain}_{0,j}$ $(\langle (n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D}) \rangle)$. For all $i' \neq 0, j' \in [i'+1..D]$

or $i' = 0, j' \in [i+1..j-1]$, we have, $n'_{i',j'} := n_{i',j'}$; and for all $j' \in [j..D]$, we have, $n'_{0,j'} := n_{0,j'} + 1$. We set $\hat{\mathbf{T}}_{s,\mathsf{new\text{-}chain}_{0,j}(s)} := \frac{\mathsf{w}(n_{1,j})\cdot n_{0,D}}{n_{0,D}+1}$.

<u>Case 4</u>: *A new chain $\mathcal{C}$, that has the same length as the current best chain, is generated, and the new chain $\mathcal{C}$ is the first chain in subset L_0 that is extended.* Similar to Case 3, for $j \in [1..D]$, we consider a sub-case 4-j in which the new chain $\mathcal{C}$ is added to "0-depth j-distance" subset $L_{0,j}$. Given that a new chain $\mathcal{C}$ is added to "0-depth j-distance" subset $L_{0,j}$, the probability that $\mathcal{C}$ is the first chain in subset L_0 that is extended is $\frac{1}{n_{0,D}+1}$. We define a function $\mathsf{new\text{-}extend\text{-}chain}_{0,j} : \hat{S} \to \hat{S}$ that takes as input a state in $\hat{S}$ and outputs an updated state when a new chain $\mathcal{C}$ is added to the "0-depth j-distance" subset $L_{i,j}$ and the chain $\mathcal{C}$ is the first chain in subset L_0 that is extended. If such event happens, the state machine moves from state $\langle (n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D}) \rangle$ to state $\mathsf{new\text{-}extend\text{-}chain}_{0,j}((n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D}))$. Let $\langle (n'_{0,1}, \cdots, n'_{0,D}), \cdots, (n'_{D-1,D}) \rangle = \mathsf{new\text{-}extend\text{-}chain}_{0,j}$ $((n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D}))$. We have, $n'_{i',j'} := 1$ for all $i' \in [0..D], j' \in [i'+1..D-1]$; $n'_{i',D} := n_{i',D}$ for all $i' \in [0..D]$, and $n'_{0,D} := n_{0,D} + 1$. We set $\hat{\mathbf{T}}_{s,\mathsf{new\text{-}extend\text{-}chain}_{0,j}(s)} := \frac{\mathsf{w}(n_{1,j})}{n_{0,D}+1}$.

<u>Case 5</u>: *No new chain is generated.* This case is generalized from Case 7 in Figure 14. The probability of this event is $1 - \sum_{i=0}^{D} \mathsf{w}(n_{i,D})$. Here, the state machine remains at the same state $\langle (n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D}) \rangle$. We set $\hat{\mathbf{T}}_{\langle (n_{0,1},\cdots,n_{0,D}),\cdots,(n_{D-1,D})\rangle, \langle (n_{0,1},\cdots,n_{0,D}),\cdots,(n_{D-1,D})\rangle} := 1 - \sum_{i=0}^{D} \mathsf{w}(n_{i,D})$.

Let $q_{(n_{0,1},\cdots,n_{0,D}),\cdots,(n_{D-1,D})}$ be the stationary probability of the state $\langle (n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D}) \rangle$. Based on Equation 1, we have,

$$\begin{cases} \sum_{n_{0,1}=1}^{\infty} \cdots \left(\sum_{n_{0,D}=1}^{\infty} \cdots \left(\sum_{n_{D-1,D}=1}^{\infty} q_{(n_{0,1},\cdots,n_{0,D}),\cdots,(n_{D-1,D})} \right) \right) = 1, \\ q_{(n_{0,1},\cdots,n_{0,D}),\cdots,(n_{D-1,D})} = \sum_{\langle (n'_{0,1},\cdots,n'_{0,D}),\cdots,(n'_{D-1,D})\rangle \in \hat{S}} \Big(q_{(n'_{0,1},\cdots,n'_{0,D}),\cdots,(n'_{D-1,D})} \cdot \\ \qquad \hat{\mathbf{T}}_{\langle (n'_{0,1},\cdots,n'_{0,D}),\cdots,(n'_{D-1,D})\rangle, \langle (n_{0,1},\cdots,n_{0,D}),\cdots,(n_{D-1,D})\rangle} \Big). \end{cases} \tag{5}$$

We define the chain growth function $\mathsf{growth} : \hat{S} \to [0,1]$ such that $\mathsf{growth}((n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D})) = \mathsf{w}(n_0) \approx n_0 \cdot \alpha$. The amplification ratio $\hat{\mathtt{A}}^{\bullet}_{D}$ equals the expected number of chains in L_0, i.e.,

$$\begin{aligned} \hat{\mathtt{A}}^{\bullet}_{D} &= \sum_{n_{0,1}=1}^{\infty} \cdots \left(\sum_{n_{0,D}=1}^{\infty} \cdots \left(\sum_{n_{D-1,D}=1}^{\infty} q_{(n_{0,1},\cdots,n_{0,D}),\cdots,(n_{D-1,D})} \cdot \frac{\mathsf{growth}((n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D}))}{\alpha} \right) \right) \\ &= \sum_{n_{0,1}=1}^{\infty} \cdots \left(\sum_{n_{0,D}=1}^{\infty} \cdots \left(\sum_{n_{D-1,D}=1}^{\infty} q_{(n_{0,1},\cdots,n_{0,D}),\cdots,(n_{D-1,D})} \cdot n_{0,D} \right) \right). \end{aligned}$$

Using the augmented Markov chain, we can find a lower bound of the amplification ratio as shown in Figure 7. For $D = 50$, we can find a lower bound $\hat{\mathtt{A}}^{\bullet}_{50} \geq 2.04$.

7.4 Achieving chain growth

Now, we show our protocol $\Pi^{\bullet}$ can achieve the chain growth property, based on the augmented Markov chain. As mentioned, we consider a hybrid experiment where all messages sent by the adversary are removed. We show that, in the hybrid experiment, the adversary cannot slow down the chain growth of the honest players. Then, we use the Chernoff bound on the augmented Markov chain to bound the chain growth of protocol $\Pi^{\bullet}$.

Lemma 7.3 (Chain growth)**.** *Consider protocol $\Pi^{\bullet}$ in the real execution $\mathsf{REAL}(\omega)$. Consider an honest player P with the best local chain $\mathcal{C}$ in round r, and an honest player P_1 with the best local chain $\mathcal{C}_1$ in round r_1, where $r_1 > r$. Then we have $\Pr\left[\mathsf{len}(\mathcal{C}_1) - \mathsf{len}(\mathcal{C}) \geq g \cdot t\right] \geq 1 - e^{-\Omega(t\cdot\alpha)}$, where $t = r_1 - r = \Omega(\kappa)$, $g = (1-\delta)\cdot \alpha_0^{\bullet}$, $\alpha_0^{\bullet} = \hat{\mathtt{A}}^{\bullet}_{D} \cdot \alpha$, and $\delta > 0$.*

Proof. In order to analyze the growth of the best chain from round r to round r_1 in the real execution $\mathsf{REAL}(\omega)$, we consider a hybrid execution $\mathsf{HYB}^{r}(\omega)$. Since the first r rounds of both the real execution $\mathsf{REAL}(\omega)$ and the hybrid execution $\mathsf{HYB}^{r}(\omega)$, the best chain at round r of the two executions is the same. Furthermore, based on Lemma 7.2, at round r_1, the best chain in the real execution $\mathsf{REAL}(\omega)$ is longer than the best chain in the hybrid execution $\mathsf{HYB}^{r}(\omega)$. Thus, from round r to round r_1, the chain growth in the real execution $\mathsf{REAL}(\omega)$ is greater than the chain growth in the hybrid execution $\mathsf{HYB}^{r}(\omega)$.

As defined in Subsection 6.1, let $Q = [q_s]_{s \in \hat{S}}$ be the stationary distribution over $\hat{S}$. Here, for each state $s \in \hat{S}$, the probability that the state s occurs in the random walk is $\Pr_{s' \sim Q}[s' = s] = q_s$, where the probability q_s is computed as in Equation 5. (Here, the state s is in the format $\langle (n_{0,1}, \cdots, n_{0,D}), \cdots, (n_{D-1,D}) \rangle$.)

We have,

$$\begin{aligned}
\mathbb{E}_{s \sim Q}\left[\mathsf{growth}(s)\right] &= \mathbb{E}_{\langle (n_{0,1},\cdots,n_{0,D}),\cdots,(n_{D-1,D})\rangle \sim Q}[\mathsf{w}(n_{0,D})] \\
&= \sum_{\langle (n_{0,1},\cdots,n_{0,D}),\cdots,(n_{D-1,D})\rangle \in \hat{S}} q_{\langle (n_{0,1},\cdots,n_{0,D}),\cdots,(n_{D-1,D})\rangle} \cdot \mathsf{w}(n_{0,D}) \\
&= \hat{\mathtt{A}}^{\bullet}_{D} \cdot \alpha.
\end{aligned}$$

Recall that each state in the augmented Markov chain reflects the number of chains in the depth-distance-based subsets. The chain growth function growth returns the probability of successfully extending a chain in the 0-depth subset L_0. Thus, the chain growth function provides an expectation for the probability that the honest players generate a new best chain. Therefore, the augmented Markov chain $(\hat{S}, \hat{\mathbf{T}})$ and the chain growth function growth are compatible to protocol $\Pi^{\bullet}$. Hence, based on Lemma 6.2 (using Chernoff bound), we have,

$$\Pr\left[\mathsf{len}(\mathcal{C}_1) - \mathsf{len}(\mathcal{C}) < (1-\delta) \cdot \hat{\mathtt{A}}^{\bullet}_{D} \cdot \alpha \cdot t\right] < e^{-\Omega(t \cdot \alpha)}.$$

□

8 Common Prefix in Multi-Extension: A New Analysis Framework

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.

8.1 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 $\mathbb{C}_{\text{best}}$ be the set of the player's best chains. For any two chains $\mathcal{C}, \mathcal{C}' \in \mathbb{C}_{\text{best}}$, 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.

Figure 16 illustrates 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 8.1 (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 $B_{\ell+1}$*[5]*. 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 Figure 16, $V_0 \| V_1 \| V_2 \| V_3'$ and $V_0 \| V_1 \| V_2 \| V_3 \| V_4 \| V_5$ are two virtual chains.*

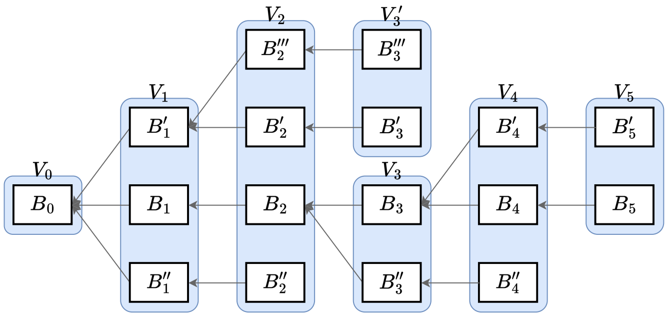


Figure 16: A toy example for the virtual block-sets and virtual chains with $D = 2$. Each block is represented by a solid rectangle and each virtual block-set is represented by a blue area that consists of multiple blocks. Here $V_0 = \{B_0\}$, $V_1 = \{B_1, B_1', B_1''\}$, $V_2 = \{B_2, B_2', B_2'', B_2'''\}$, $V_3 = \{B_3, B_3''\}$, $V_3' = \{B_3', B_3'''\}$, $V_4 = \{B_4, B_4', B_4''\}$, $V_5 = \{B_5, B_5'\}$, 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 $\mathcal{V}_{\text{best}} = V_0 \| V_1 \| V_2 \| V_3 \| V_4 \| V_5$.

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 Figure 16, 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 8.2 (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 $\mathcal{V}[\neg\kappa]$.

8.2 Common prefix property w.r.t. virtual chains

We are now ready to define the common prefix property w.r.t. virtual chains. The property states that all honest players share the same common prefix of virtual chains after removing the last κ virtual blocks.

Definition 8.3 (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. This, in turn, requires the adversary to have control over the majority of the stake. Therefore, our protocol achieves the common prefix property under the assumption that the majority of stake is controlled by honest players.

Definition 8.4 (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.*

[5]In protocol $\Pi^\bullet$, the block $B_{\ell+1}$ is in the form of $\langle \eta, r, \text{PK}, \sigma \rangle$, where η is the hash value of the previous block, r is the current round number, PK is the public key of the player, and σ is the signature of the player over $\langle \eta, r \rangle$. We say B_ℓ is linked by $B_{\ell+1}$ if $\eta = h(B_\ell)$.

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 8.5 (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 8.5, we have, $V_i' = V_i$.

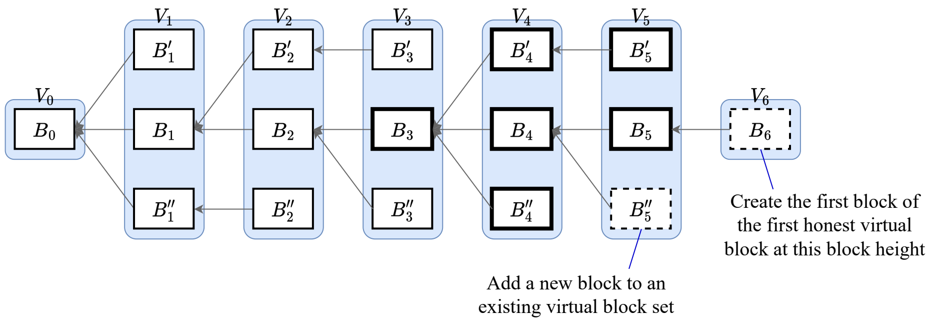


Figure 17: A toy example for illustrating an extension of honest players. Honest players extend the set of best chains from Figure 16, using 2-distance-greedy strategy. The blue blocks denote the new blocks. Here, the players generate either a new block to create a new longest chain (that is longer than the current longest chain) or a new block that is added to an existing virtual block-set.

We will demonstrate that there is at most one honest virtual block-set at each block height. The definition of virtual block-sets states that honest players only extend blocks that belong to the same virtual block-set. We consider two cases as follows (see Figure 17 for an example).

- If an honest player creates a new longest chain, a new honest virtual block-set is created at the new block height, as there was no honest virtual block-set at this height previously.
- If honest players do not create a new longest chain, they can only create a new block in an existing virtual block-set.

Lemma 8.6. *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).*

Proof. Let $\mathcal{C}_{\text{best}}$ be the best chain in the local state of player P at the beginning of round r. Let $\mathcal{C}' = \mathcal{C}[0, \ell' - 1]$. At round r, player P extend the chain $\mathcal{C}'$ by adding the block $B_{\ell'}$ to generate the new chain $\mathcal{C}$. Since the honest players only extend the chains in the set of best chains, we have, $\mathcal{C}' \in \mathbb{C}_{\text{best}}$. Recall from procedure D-$\mathsf{BestChainSet}^{\bullet}$, the distance from the best chain $\mathbb{C}_{\text{best}}$ to $\mathcal{C}'$ is smaller than D, i.e., $\mathsf{distance}(\mathcal{C}_{\text{best}} \to \mathcal{C}') \leq D$. We consider two cases of $\mathcal{C}'$ as follows.

- The length of the chain $\mathcal{C}'$ equals ℓ, i.e., $\mathsf{len}(\mathcal{C}') = \ell$. In this case, we have, $\ell' = \mathsf{len}(\mathcal{C}) = \mathsf{len}(\mathcal{C}' \| B) = \ell + 1$. In other words, a new best chain of length $\ell + 1$ is generated.
- The length of the chain $\mathcal{C}'$ is smaller than ℓ, i.e., $\mathsf{len}(\mathcal{C}') < \ell$. Since $\mathsf{distance}(\mathcal{C}_{\text{best}} \to \mathcal{C}') \leq D$, from Definition 4.1, we have, $\mathcal{C}_{\text{best}}[0, \ell - D] \preceq \mathcal{C}'$. Thus, $\mathcal{C}_{\text{best}}[0, \ell + 1 - D] \preceq \mathcal{C}$ (as $\mathcal{C} = \mathcal{C}' \| B$). Therefore, $\mathsf{distance}(\mathcal{C}_{\text{best}}[0, \ell'] \to \mathcal{C}) \leq D$. Plus, since $\mathcal{C}_{\text{best}}$ belongs to $\mathcal{V}_{\text{best}}$, we have, $\mathcal{C}_{\text{best}}[\ell'] \in V_{\ell'}$. Thus, based on the definition of the virtual block-set, we have, $B_{\ell'} \in V_{\ell'}$.

$\square$

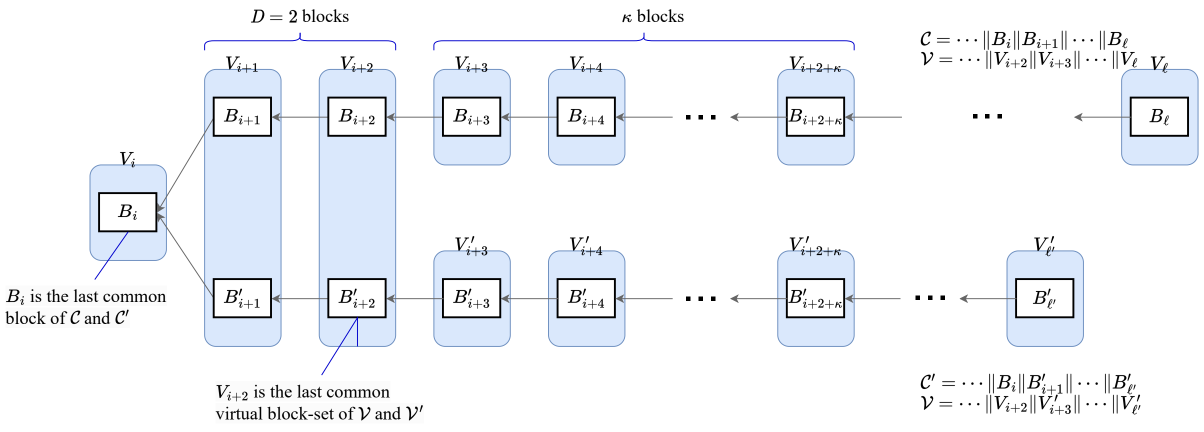
We assume $\alpha^\bullet = \lambda \cdot \beta^\bullet$, $\lambda > 1$. If $r_1 - r$ is big enough, the adversary cannot extend the virtual chain as fast as the honest players. Given that $\Delta \cdot \alpha^\bullet \ll 1$, most of the time, there is at most one honest virtual block-set at a block height. Now, we are ready to prove the common prefix property on virtual chains.

Lemma 8.7 (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,*

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

Proof. Assuming towards a contradiction that the virtual chain $\mathcal{V}$ does not share a common prefix with the virtual chain $\mathcal{V}'$ after removing the last κ virtual block-sets, i.e., $\mathcal{V}[\neg\kappa] \npreceq \mathcal{V}'$. Let us consider the last common virtual block-set of $\mathcal{V}$ and $\mathcal{V}'$ generated at round r_0. By the chain growth property in Lemma 7.3, from round r_0 to round r, the virtual chain $\mathcal{V}$ must increase in length by at least $\alpha^\bullet \cdot t$, where $t = r - r_0$. However, from Lemma 8.6, there can only be at most one honest virtual block-set at any given block height. Thus, the adversary must generate at least $\alpha^\bullet \cdot t$ virtual block-sets from round r_0 to round r, which occurs with probability less than $e^{-\Omega(\kappa)}$. □

8.3 From common prefix w.r.t. virtual chains, to the standard common prefix property



$D = 2$ blocks
κ blocks
$\mathcal{C} = \cdots \| B_i \| B_{i+1} \| \cdots \| B_\ell$
$\mathcal{V} = \cdots \| V_{i+2} \| V_{i+3} \| \cdots \| V_\ell$
V_{i+1}
V_{i+2}
V_{i+3}
V_{i+4}
$V_{i+2+\kappa}$
V_ℓ
B_{i+1}
B_{i+2}
B_{i+3}
B_{i+4}
$B_{i+2+\kappa}$
B_ℓ
V_i
B_i
V'_{i+3}
V'_{i+4}
$V'_{i+2+\kappa}$
$V'_{\ell'}$
B'_{i+1}
B'_{i+2}
B'_{i+3}
B'_{i+4}
$B'_{i+2+\kappa}$
$B'_{\ell'}$
B_i is the last common block of $\mathcal{C}$ and $\mathcal{C}'$
V_{i+2} is the last common virtual block-set of $\mathcal{V}$ and $\mathcal{V}'$
$\mathcal{C}' = \cdots \| B_i \| B'_{i+1} \| \cdots \| B'_{\ell'}$
$\mathcal{V} = \cdots \| V_{i+2} \| V'_{i+3} \| \cdots \| V'_{\ell'}$

Figure 18: From common prefix w.r.t. virtual chains, to the standard common prefix property. If common prefix property does not hold, i.e., $\mathcal{C}[\neg(\kappa + D)] \preceq \mathcal{C}'$, then common prefix w.r.t. virtual chain property does not hold, i.e., $\mathcal{V}[\neg\kappa] \preceq \mathcal{V}'$. Here, $\mathcal{C}$ belongs to $\mathcal{V}$ and $\mathcal{C}'$ belongs to $\mathcal{V}'$.

We prove the common prefix property w.r.t. virtual chains. Lemma 8.7 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 (see Figure 18). 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 8.8 (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,*

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

Proof. Assuming toward a contradiction that $\mathcal{C}[\neg(\kappa + D)] \npreceq \mathcal{C}'$. Let $\mathcal{V}$ and $\mathcal{V}'$ be the virtual chains of $\mathcal{C}$ and $\mathcal{C}'$, respectively. Let $\ell = \mathsf{len}(\mathcal{C})$ be the length of the chain $\mathcal{C}$ and $\ell' = \ell - (\kappa + D)$. Since $\mathcal{C}[\neg(\kappa + D)] \npreceq \mathcal{C}'$, the blocks at block height ℓ' of $\mathcal{C}$ and $\mathcal{C}'$ are different, i.e., $\mathcal{C}[\ell] \neq \mathcal{C}'[\ell]$. Thus, we have, $\mathsf{distance}(\mathcal{C}[0, \ell' + D] \to \mathcal{C}'[0, \ell' + D]) > D$.

Let $\mathcal{V}[\ell' + D]$, $\mathcal{V}'[\ell' + D]$ be the virtual block-sets at block height $\ell' + D$ of the virtual chains $\mathcal{V}$ and $\mathcal{V}'$, respectively. We have, $\mathcal{C}[\ell' + D] \in \mathcal{V}[\ell' + D]$ and $\mathcal{C}'[\ell' + D] \in \mathcal{V}'[\ell' + D]$. As $\mathsf{distance}(\mathcal{C}[0, \ell' + D] \to \mathcal{C}'[0, \ell' + D]) > D$, we have,

$\mathcal{V}[\ell' + D] \neq \mathcal{V}'[\ell' + D]$. In other words, $\mathcal{V}[0, \ell' + D] \neq \mathcal{V}'[0, \ell' + D]$ or $\mathcal{V}[\neg\kappa] \npreceq \mathcal{V}'$. This contradicts the common prefix property w.r.t. virtual chains in Lemma 8.7.

□

9 Chain quality and best possible unpredictability

We show that our protocol can achieve the chain quality and the best possible unpredictability properties.

9.1 Chain quality

After proving the chain growth and common prefix properties, the proof of chain quality will be very similar to the proof in [PSs17]. 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 9.1 (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 $\mu = 1 - (1+\delta) \cdot \frac{1}{\lambda}$.*

Proof. Assuming toward contradiction that all blocks from round r to round r_1 are generated by malicious players. From Lemma 7.3, we have, the length of the best chain from round r to round r_1 increase by at least $(1-\delta) \cdot \alpha^\bullet \cdot t$, where $t = r_1 - r$. As all blocks from round r' to round r'' are generated by malicious players, the adversary can grow the chain with the rate $(1-\delta) \cdot \alpha^\bullet$. Recall that, the adversary can grow the chain with the rate at most $\beta^\bullet$. Thus, we have, $\beta^\bullet > (1-\delta) \cdot \alpha^\bullet$. This contradicts the assumption that $\beta^\bullet < (1-\delta) \cdot \alpha^\bullet$. □

9.2 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 9.2. *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,*

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

Proof. Assuming toward contradiction that the player P is 2-predictable at round r. Hence, there exists a round $r' > r$ such that the adversary $\mathcal{A}$ can make an accurate prediction $z_P^{r'}$ at round r and $\mathsf{len}(\mathcal{C}^{r'}) = \mathsf{len}(\mathcal{C}^r) + 1$, where $\mathcal{C}^r$ and $\mathcal{C}^{r'}$ are the best chains at round r and r', respectively. Let $(\mathrm{SK}, \mathrm{PK})$ be the key pair of player P. In order to predict whether or not the player P can extend the chain $\mathcal{C}_P^{r'}$, the adversary must be able to compute the context η. Since the context η is computed as the hash value of the last block in the chain $\mathcal{C}^{r'}$, the adversary must know the chain $\mathcal{C}^{r'}$ to make a correct prediction on whether or not the player P can extend the chain $\mathcal{C}^{r'}$. As the chain $\mathcal{C}^{r'}$ is not generated at round r, the adversary cannot provide an accurate prediction. □

10 Extensions

We provide the extensions for our protocol to make it more practical. In Subsection 10.1, we will "upgrade" our protocol to a regular blockchain protocol so that payload (e.g., the transactions) can be included. Then, in Subsection 10.2, we further extend our protocol in a more realistic "non-flat" model. Finally, in Subsection 10.3, we follow a similar strategy in [BGK+18] to allow new players to join the system and participate in the process of extending the chains if they have their stake registered a specified number of rounds earlier.

10.1 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, (SK, PK), from a unique signature scheme $(\mathsf{uKeyGen}, \mathsf{uSign}, \mathsf{uVerify})$ and a pair of keys, $(\tilde{\text{SK}}, \tilde{\text{PK}})$, 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 [BM99].

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 $\tilde{B}_{\ell+1} = \langle \tilde{h}_\ell, B_{\ell+1}, X_{\ell+1}, \tilde{\text{PK}}, \tilde{\sigma}\rangle$ where $\tilde{\sigma} \leftarrow \mathsf{Sign}_{\tilde{\text{SK}}}(\tilde{h}_\ell, B_{\ell+1}, X_{\ell+1})$, $\tilde{h}_\ell := \mathsf{hash}(\tilde{B}_\ell)$, $X_{\ell+1}$ 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.

10.2 Blockchain in the non-flat model

In our previous sections, 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. In this section, 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, $B_0 = \langle(\langle \text{PK}_1, s_1\rangle, \langle \text{PK}_2, s_2\rangle, \cdots, \langle \text{PK}_N, s_N\rangle), \mathsf{rand}\rangle$. The number of stakes held by each player can vary in this model. For a PoS player with the key pair (PK, SK) holding s stake at round r, the hash inequality is changed as follows:

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

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 $1 - (1-p)^s \approx s \cdot p$.

10.3 Defending against adaptive registration

Our design can be further improved to allow for the dynamic registration and deregistration of players during the protocol's execution. As a reminder, the chain extension process relies on the hash inequality $\mathsf{H}(\textit{context}, \textit{solution}) < \mathtt{T}$, where $\textit{solution}$ 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 $\textit{context}$. In this strategy, a malicious player generates a key-pair (PK, 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. Similar to the approach in [BGK+18], we defend against rejection re-sampling attacks by requiring new players to have their stake registered for a specified number of rounds before being allowed to extend the chains. To join the protocol, player P generates two key pairs: $(\text{SK}, \text{PK}) \leftarrow \mathsf{uKeyGen}(1^\kappa)$ and $(\tilde{\text{SK}}, \tilde{\text{PK}}) \leftarrow \mathsf{KeyGen}(1^\kappa)$. P keeps SK and $\tilde{\text{SK}}$ secret and broadcasts a registration transaction. After a player has registered, they are eligible to extend the chain after η blocks have been added to the blockchain. It is important to note that players who registered prior to the start of the protocol (i.e. in the genesis block) do not have to wait η blocks before they can extend the chain.

By implementing this requirement, malicious players cannot register key pairs to extend the chains immediately. However, they can still attempt to register biased key pairs and then extend the chains many rounds later. However since the adversary cannot accurately predict future events, they cannot choose a biased key pair that will increase their chances of extending the chain many rounds in the future.

11 Related Work

We summarize the existing results for the designs and analysis of PoS protocols.

11.1 Proof-of-stake protocols

The idea of using coins/stakes to construct cryptocurrency has been intensively considered. Since the inception of the idea in an online forum [Bit11], several proof-of-stake proposals have been introduced or implemented (e.g., [NXT14, Kwo14, Vas14, But15, BGM16]). 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 [DPS19], Ouroboros Praos [DGKR18] and Genesis [BGK+18], and a protocol by Bagaria et al. [BDK+19]. Note that, all of the above protocols are single-extension protocols (thus suffering from the impossibility result in Section 3).

In Snow White [DPS19], 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 [PS17] (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 [DGKR18], 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) [MRV99] 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 [BGK+18], new players are allowed to join the protocol execution securely.

In Bagaria et al. [BDK+19], 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 [BGK+18] to allow new players to join the system. We remark that *the work in [BDK+19] is independent and concurrent from our effort in this paper.*

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 [CM19, GHM+17], EOS [EOS18], Dfinity [HMW18], Casper [BG17]) have been constructed in an interactive fashion.

In Algorand [CM19, GHM+17], 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 [EOS18] 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 [HMW18] 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[BG17], 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 [CM19, GHM+17], EOS [EOS18], Dfinity [HMW18], Casper [BG17]) require quadratic communication complexity to generate new blocks. HotStuff [YMR+19] 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). Deb et al. [DKT20a] 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 [DKT20a] 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.

11.2 Security analysis for Bitcoin-like PoS protocols

Bagaria et al. [BDK+19] 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 [DPS19, DGKR18], common prefix property is guaranteed with error $e^{-\Omega(\kappa)}$ by removing the last $O(\kappa^2)$ blocks. While in Bitcoin, the consistency is guaranteed with error $e^{-\Omega(\kappa)}$ by removing only the last $O(\kappa)$ blocks. Blum et al. [BKM+20] 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 [BKM+20], 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). Kiayias et al. [KQR20] extends the result from [BKM+20]. Here, the "multiple honest" rounds are treated as "unique honest" rounds (the rounds that have exactly one honest player that can generate a new block). Dembo et al. [DKT+20b] 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.

Unpredictability. 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. Brown-Cohen et al. [BCNPW19] 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.

References

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

[BCNPW19] 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.

[BDK$^+$19] 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.

[BG17] Vitalik Buterin and Virgil Griffith. Casper the friendly finality gadget. *CoRR*, abs/1710.09437, 2017.

[BGK$^+$18] 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.

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

[Bit11] Bitcointalk. Proof of stake instead of proof of work. July 2011. Online post by QuantumMechanic, available at https://bitcointalk.org/index.php?topic=27787.0.

[BKM$^+$20] 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.

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

[BM99] 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.

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

[Can00] Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, January 2000.

[Can17] Clément L Canonne. A short note on poisson tail bounds. 2017. http://www.cs.columbia.edu/~ccanonne/files/misc/2017-poissonconcentration.pdf.

[CLLM12] 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.

[CM19] 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.

[DGKR18] 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.

[DKT20a] Soubhik Deb, Sreeram Kannan, and David Tse. Posat: Proof-of-work availability and unpredictability, without the work. *FC 2021,*, 2020.

[DKT$^+$20b] 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.

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

[DPS19] 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.

[DY05] 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.

[EOS18] EOS whitepaper. 2018. https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md.

[GHM$^+$17] 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.

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

[GO93] 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.

[Goe15] Michel Goemans. Chernoff bounds, and some applications. 2015. https://math.mit.edu/~goemans/18310S15/chernoff-notes.pdf.

[Gol04] Oded Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, Cambridge, UK, 2004.

[HMW18] Timo Hanke, Mahnush Movahedi, and Dominic Williams. Dfinity technology overview series, consensus system. *arXiv preprint arXiv:1805.04548*, 2018.

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

[KQR20] Aggelos Kiayias, Saad Quader, and Alexander Russell. Consistency of proof-of-stake blockchains with concurrent honest slot leaders. *ICDCS*, 2020.

[KRs18] Lucianna Kiffer, Rajmohan Rajaraman, and abhi shelat. A better method to analyze blockchain consistency. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 729–744. ACM Press, October 2018.

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

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

[MRV99] Silvio Micali, Michael O. Rabin, and Salil P. Vadhan. Verifiable random functions. In *40th FOCS*, pages 120–130. IEEE Computer Society Press, October 1999.

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

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

[PS17] 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.

[PSs17] 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.

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

[YMR+19] 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.

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. [BCNPW19] 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.

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.

B Supplemental materials for Section 3

B.1 Existing single-extension proof-of-stake protocols

We now describe the existing state-of-the-art PoS protocols in [DPS19, DGKR18, BDK+19] as single-extension PoS protocols.

Snow White [DPS19]. The Snow White protocol [DPS19] 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 $\mathsf{Validate}$, $\mathsf{BestChain}$, $\mathsf{Context}$, and Extend are constructed as follows:

- The algorithm $\mathsf{Context}$ 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 $\texttt{PK}$. 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 [DGKR18]. The Ouroboros Praos protocol is constructed using a *Verifiable Random Function* [DY05] (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 $(\texttt{SK}, \texttt{PK})$. The algorithm Prove takes the secret key $\texttt{SK}$ and a message msg as input and returns a pseudorandom output σ along with a proof π. We write $(\sigma, \pi) := \mathsf{Prove}_{\texttt{SK}}(msg)$. The algorithm Ver takes a public key $\texttt{PK}$, 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 [DPS19], the Ouroboros Praos protocol [DGKR18] 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 $\texttt{SK}$. The algorithm computes $(\sigma, \pi) := \mathsf{Prove}_{\texttt{SK}}(\eta, r)$ and returns a new block if it holds that $\sigma < \texttt{T}$, where $\texttt{T}$ 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 $\texttt{T}$, 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. [BDK$^+$19]. The protocol described in Bagaria et al. [BDK$^+$19] 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 $\texttt{SK}$. The algorithm computes $(\sigma, \pi) := \mathsf{Prove}_{\texttt{SK}}(\eta, r)$ and returns a new block if it holds that $\sigma < \texttt{T}$, where $\texttt{T}$ 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 $\texttt{T}$, 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 $\mathbb{C}$.

C Supplemental materials for Section 4

C.1 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 [Gol04] 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 C.1. *We say* (uKeyGen, uKeyVer, uSign, 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

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

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*

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

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

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

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

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

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

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

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

Unique signature schemes and related notions have been investigated in literature (e.g., [GO93, MRV99, Lys02]). Please see Section 6.5.1 of Goldreich's textbook [Gol04] for detailed discussions about the constructions. Several efficient constructions can be found in the literature. For example, the well-known BLS signature [BLS01] can be a good candidate.

C.2 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 C.2 (Multi-extension framework for PoS protocols)**.** *A multi-extension PoS protocol Π° is parameterized by 4 deterministic algorithms* ($\mathsf{Validate}^\circ$, $\mathsf{BestChainSet}^\circ$, $\mathsf{Context}^\circ$, Extend°) *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* $\bot$*.*
- *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* $\textsc{sk}$*, and returns a new block* B *or* $\bot$ *(if no new block is generated). Here, the secret key* $\textsc{sk}$ *is generated by a player* P *in the blockchain initialization phase, and the corresponding public key of* $\textsc{sk}$ *will be stored in the genesis block. The function* $\mathsf{Extend}^\circ(\eta, r, \textsc{sk})$ *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* $\mathbb{C}_{\text{best}}$*. Here, the honest players will* ***extend multiple chains****, i.e., all the chains in the set of the best chains* $\mathbb{C}_{\text{best}}$*. 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* $\mathsf{Validate}$*,* $\mathsf{Context}$*, and* Extend *in the single-extension PoS protocol, defined in Definition 3.1. The difference between the single-extension and multi-extension protocols is the utilization of different algorithms, namely* $\mathsf{BestChain}$ *and* $\mathsf{BestChainSet}^\circ$*, for determining the chains to be extended. The single-extension protocol uses the best chain algorithm* $\mathsf{BestChain}$ *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* $\textsc{pk}_j$ *and private key* $\textsc{sk}_j$*. The public keys of all players are stored in the genesis block of the blockchain system. We let* B_0 *denote the genesis block. .*

Algorithm 5: A multi-extension proof-of-stake 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 $(\textsc{sk}, \textsc{pk})$ and local chain set $\mathbb{C}$, proceeds as follows.
1 Upon receiving a chain $\mathcal{C}'$, verify $\mathsf{Validate}^\circ(\mathcal{C}', r) = 1$ and set $\mathbb{C} := \mathbb{C} \cup \{\mathcal{C}'\}$;
2 Set $\mathbb{C}_{\text{best}} := \mathsf{BestChainSet}^\circ(\mathbb{C})$;
3 for $\mathcal{C} \in \mathbb{C}_{\text{best}}$ do
4     $\eta := \mathsf{Context}^\circ(\mathcal{C})$; $B := \mathsf{Extend}^\circ(\eta, r, \textsc{sk})$;
5     if $B \neq \bot$ then
6         $\mathcal{C}' := \mathcal{C} \| B$; Add $\mathcal{C}'$ to $\mathbb{C}$; Broadcast $\mathcal{C}'$;
```

Blockchain extension phase. *A multi-extension proof-of-stake protocol* Π *is described in Algorithm 5. In each round* r*, a player* P *with the secret key* $\textsc{sk}$ *proceeds as follows. First, the player* P *set* $\mathbb{C}_{\text{best}} := \mathsf{BestChainSet}^\circ(\mathbb{C})$*. Here the local set of chains* $\mathbb{C}$ *consists of all valid chains that are received (or generated) by* P*. Then, for each chain* $\mathcal{C} \in \mathbb{C}_{\text{best}}$*, 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* $\textsc{sk}$*, 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 C.3. *We remark that there are other ways to design multi-extension protocols. However, to simplify the presentation, we focus on the design in Definition C.2. We used this design of multi-extension protocols for our protocol in Section 4.*

D Supplemental materials for Section 8

D.1 An upper bound for the chain growth of the adversary

We now analyze the chain growth of an adversary that extends the chain by itself. First, we show that the probability that an adversary can generate a new block from a chain in a round is at most β. Then, we show that an arbitrary strategy adversary can amplify its chain growth by a factor of at most e. Therefore, the probability that an adversary can extend the best chain in a round is at most $e \cdot \beta$.

Lemma D.1. *Consider protocol* $\Pi^\bullet$*. Consider a valid chain* $\mathcal{C}$ *at round* r*. The probability that the chain* $\mathcal{C}$ *is extend by a malicious player at round* r *is* $\beta = 1 - (1-p)^{N \cdot \rho}$*.*

Proof. The uniqueness of the signature generation property of the unique signature scheme ensures that in each round, each malicious player can generate one solution from the context of the chain $\mathcal{C}$. In other words, in each round, each malicious player can make at most one attempt to generate a new block by making one query to the random oracle. The event where a malicious player successfully extends the chain $\mathcal{C}$ can be modeled as an (independent) Bernoulli random variable which takes the value 1 with probability p. Plus, the number of malicious players is at most $N \cdot \rho$. Therefore, the probability that the chain is extended by a malicious player at round r is at most $\beta = 1 - (1-p)^{N\cdot\rho}$. □

Lemma D.2. *Consider protocol $\Pi^\bullet$. Assume that the malicious players could follow any arbitrary strategy to extend a chain $\mathcal{C}_1$ at round r_1 into $\mathcal{C}_2$ at round r_2, where $r_2 = r_1 + 1$, and $t = \Omega(\kappa)$. For some $\epsilon > 0$, we have $\Pr\left[\mathsf{len}(\mathcal{C}_2) - \mathsf{len}(\mathcal{C}_1) < (1+\epsilon)\cdot\beta^\bullet\cdot t\right] \geq 1 - e^{-\Omega(t)}$, where $\beta^\bullet = e \cdot \beta$, and $e = 2.72$.*

Proof. Similar to the proof of Lemma 3.10, we model the chain extension of the adversary as a random tree in which the extension in the branches is independent. We describe the chain extension of the adversary as a branching process as follows. At the beginning, there is only one branch of length 0, i.e., $Z_0 = \{0\}$. Let Z_j be the set of all branches at round j, where $j \in [t]$ and G_j be the number of branches in Z_j. Let $X_{j,i}$ denote the random variable corresponding to the random process in the i-th branch in Z_j. Here $X_{j,i}$ are independent and identically distributed random variables of X, where X is a Poisson random variable that has the expected value of β. Let $\ell_{j,i}$ be the length of the i-th branch in Z_j. We will add $X_{j,i}+1$ branches with the length $\ell_{j,i}, \ell_{j,i}+1, \cdots, \ell_{j,i}+X_{j,i}$ into Z_{j+1}. We denote T_j as the maximum length of all branch in Z_j, i.e.,$T_j = \max_{i\in\{1,2,\cdots,G_j\}} \ell_{j,i}$. The length of a branch set is equivalent to the increasing length of the longest chain.

Consider the set of branches Z_t at time $r_1 + t$. For any $\epsilon'' > 0$, we have $\Pr[G_t > (\beta+1)^{(1+\epsilon'')\cdot k}] < e^{-\Omega(\kappa)}$. Let $Y = \sum_{j=1}^{t} X_j$ be a Poisson random variable with the expected value of $t\cdot\beta$. We have,

$$\begin{aligned}
\Pr\left[T_t > (1+\epsilon)\cdot t\cdot\beta\cdot e\right] \leq & \sum_{i\in\{1,2,\cdots,G_t\}} \Pr\left[\ell_{t,i} > (1+\epsilon)\cdot t\cdot\beta\cdot e\right] \\
\leq & G_t \cdot \Pr\left[Y > (1+\epsilon)\cdot t\cdot\beta\cdot e\right] \\
\leq & (\beta+1)^{(1+\epsilon'')\cdot t}\cdot\Pr\left[Y > (1+\epsilon)\cdot t\cdot\beta\cdot e\right] + e^{-\Omega(\kappa)} \\
\leq & \Pr\left[Y > (1+\epsilon')\cdot t\cdot\beta\right] + e^{-\Omega(\kappa)} = e^{-\Omega(k)}.
\end{aligned}$$

□