Deep Selfish Proposing in Longest-Chain Proof-of-Stake Protocols

Roozbeh Sarenche, Svetla Nikova, and Bart Preneel

COSIC, KU Leuven, Belgium
{roozbeh.sarenche,svetla.nikova,bart.preneel}@esat.kuleuven.be

Abstract. It has been shown that the selfish mining attack enables a miner to achieve an unfair relative revenue, posing a threat to the progress of longest-chain blockchains. Although selfish mining is a well-studied attack in the context of Proof-of-Work blockchains, its impact on the longest-chain Proof-of-Stake (LC-PoS) protocols needs yet to be addressed. This paper involves both theoretical and implementation-based approaches to analyze selfish proposing[1] attack in the LC-PoS protocols. We discuss how factors such as the nothing-at-stake phenomenon and the proposer predictability in PoS protocols can make the selfish proposing attack in LC-PoS protocols more destructive compared to selfish mining in PoW. In the first part of the paper, we use combinatorial tools to theoretically assess the selfish proposer's block ratio in simplistic LC-PoS environments and under simplified network connection. However, these theoretical tools or classical MDP-based approaches cannot be applied to analyze the selfish proposing attack in real-world and more complicated LC-PoS environments. To overcome this issue, in the second part of the paper, we employ deep reinforcement learning techniques to find the near-optimal strategy of selfish proposing in more sophisticated protocols. The tool implemented in the paper can help us analyze the selfish proposing attack across diverse blockchain protocols with different reward mechanisms, predictability levels, and network conditions.

Keywords: Blockchain · Proof-of-Stake · Selfish proposing · Deep Q-learning.

1 Introduction

In Proof-of-Stake (PoS) blockchains, proposers—namely, users who have deposited a specific amount of stake in the blockchain ledger—are responsible for proposing new blocks and extending the blockchain ledger [4]. Once a proposer is selected to propose a block, it needs to follow a pre-determined fork choice rule to select a chain on top of which the new block will be proposed. Fork choice rules are essential for blockchains to achieve safety and liveness [21]. Bitcoin [20] uses the longest-chain fork choice rule, where miners choose the longest chain

[1] As there is no mining process in PoS blockchains, we refer to this attack as "selfish proposing".

to mine on top of it. Although the longest-chain rule was originally designed for Proof-of-Work (PoW), some PoS-based blockchain protocols have also implemented the longest-chain fork choice rule within their consensus layers, such as the Ouroboros protocol in Cardano [17,10], the Emmy protocol in Tezos [15], and Snow White [9]. In PoW, the longest chain refers to the chain with the most amount of work. However, in PoS, since there is no mining process, the longest chain refers to the chain that is literally the longest.

One of the most important attacks that can threaten longest-chain-based blockchain protocols is the selfish mining attack presented by Eyal and Sirer [12]. The authors of this paper have shown that a malicious miner in Bitcoin can increase his payoff by deviating from mining honestly. In the selfish mining attack, once the attacker mines a new block, he keeps his block secret rather than immediately publishing it to the other mining nodes. By keeping his blocks secret, the attacker can cause some part of the honest mining power to get wasted since the honest nodes continue mining on top of the public chain which is shorter than the attacker's secret chain. Due to the selfish mining attack, some of the honest blocks get orphaned, i.e., get excluded from the main chain. This results in an increase in the ratio of the number of attacker's blocks added to the main chain to the total number of main-chain blocks, which we refer to as the attacker's "block ratio". The selfish mining attack is applicable to any blockchain that is designed based on the longest-chain fork choice rule. Thus, not only the PoW-based blockchains but also PoS-based blockchains that use the longest-chain paradigm face the threat of selfish mining. Note that since no mining process is involved in PoS-based blockchains, we use the term "selfish proposing" rather than selfish mining to describe this attack in the PoS context.

There exist several reasons that make selfish proposing in PoS even more destructive compared to selfish mining in PoW. The first reason is the presence of the nothing-at-stake phenomenon in PoS-based blockchains. This phenomenon implies that since there is no mining process in PoS protocols, generating multiple blocks could be wasteless for proposers [7]. Therefore, a PoS proposer has the flexibility to modify the content of an already generated block as long as the block is not yet published. However, this is not the case in PoW-based blockchains. A PoW miner should specify the block content prior to the start of the mining process, and the block content cannot be changed once it is mined. The block content, in both PoW and PoS, includes a reference to its parent block. In PoS-based blockchains, the nothing-at-stake phenomenon allows a selfish proposer to postpone specifying the block's parent until just before the block needs to be published. This can provide a selfish proposer in PoS-based blockchains with a set of new actions to fortify the selfish proposing attack [13].

Another bottleneck of PoS protocols that leads to a higher selfish proposing profit is the proposer predictability. In PoS protocols, there exists a lottery mechanism that specifies the block proposer(s) for each slot. To select proposers in a random manner, the lottery mechanism takes a pseudorandom seed as one of its inputs. This seed, in most PoS-based blockchains, is extracted from the blockchain ledger information. To ensure consistency and liveness, the pseudorandom seed for each slot needs to be determined when we are sufficiently far from that specific slot [3]. Knowing these pseudorandom seeds can help validators predict the block proposers in future slots, which is impossible in PoW. Based on how the PoS protocol is designed, the validators can have different levels of predictability. In some protocols, validators can only predict the slots in which they themselves are selected as proposers, while in other protocols, the validators can predict the block proposers for all the upcoming slots. In both of these protocols, a selfish proposer can use information regarding the future proposers to improve his selfish proposing strategy. During a selfish mining attack in PoW, some of the attacker's blocks may get orphaned. In PoS protocols having knowledge about future slot proposers can help a selfish proposer reduce the risk of losing his blocks during the attack and increase his block ratio.

In PoW blockchains, tools such as the Markov chain and the Markov Decision Process (MDP) can be used to obtain the selfish miner's optimal strategies [22,24]. These tools are useful for analyzing environments with a limited set of actions and states. However, to consider the effect of the nothing-at-stake phenomenon and predictability on the selfish proposer's strategy in longest-chain PoS blockchains, it is necessary to handle larger sets of actions and states. Therefore, the classical MDP-based tools cannot directly be used to analyze the selfish proposing attack in longest-chain PoS protocols. One approach to assess the environments with huge sets of actions and states is to use deep reinforcement learning techniques such as deep Q-learning. These techniques not only enable us to handle larger sets of actions and states but also facilitate the study of the selfish proposing profitability under more realistic and complicated environments. There exist research papers such as [5,6] that have used deep reinforcement learning techniques to analyze selfish mining in PoW protocols. However, developing a tool for analyzing the selfish proposing attack in longest-chain PoS has yet to be addressed.

In this paper, we assess the selfish proposing attack in the longest-chain PoS (LC-PoS) blockchains. In Appendix B, we introduce our model which is built upon the model presented in [7]. In our paper, we extend and modify the PoS blockchain model and definitions presented in [7] to propose a realistic model that fits better to the PoS protocols currently in use. In Appendix C, we define the terms "block ratio" and "time-averaged profit" and discuss the impact of the selfish proposing attack on the attacker's profitability in LC-PoS protocols, comparing it to the profitability of selfish mining in PoW. In fact, we explain how an increase in the attacker's block ratio due to selfish proposing can lead to a rise in the attacker's time-averaged profit. In the main body of the paper, we present:

Nothing-at-stake selfish proposing attack in LC-PoS In Section 2, we generalize the nothing-at-stake selfish proposing attack introduced in [13] by taking into account a set of new actions that can lead to a more profitable strategy. Besides, we prove interesting lower bounds on the minimum amount of stake share that can make the selfish proposing attack more profitable than honest proposing in LC-PoS protocols with perfect randomness.

Predictable selfish proposing attack in full-predictable LC-PoS In Section 3, we use combinatorial tools to theoretically assess the selfish proposing block ratio under multiple strategies in full-predictable LC-PoS protocols. Moreover, we present an optimal selfish proposing strategy for full-predictable LC-PoS protocols.

Deep Q-learning tool to analyze the selfish proposing attack In Section 4, we present a deep Q-learning tool to analyze the selfish proposing attack in realistic and more complicated LC-PoS environments such as semi-predictable protocols. We demonstrate how this tool can assist in discovering the near-optimal selfish proposing strategy for selfish proposers with varying stake shares and communication capabilities under different LC-PoS environments.

In the main text, terms such as "honest strategy", "communication capability", and "block ratio" are frequently used, which are defined in Definitions 20, 21, and 22 presented in Appendix C, respectively. Readers are encouraged to review these definitions before delving into the main text.

2 Nothing-at-stake selfish proposing in LC-PoS protocols

A fundamental difference between LC-PoS protocols and PoW protocols is the fact that generating a new valid block in LC-PoS protocol is effortless. This phenomenon which is known as nothing-at-stake can provide an attacker in LC-PoS protocols with a set of new actions and strategies that are impossible in PoW protocols. The authors in [13] have introduced the nothing-at-stake selfish proposing attack. However, the introduced attack is limited to an attacker with communication capability equal to 0 and can only benefit from a small set of new actions made possible by the nothing-at-stake phenomenon. In this paper, we generalize nothing-at-stake selfish proposing by considering a set of new actions that can lead to an increase in selfish proposing profitability.

Note that throughout this paper, we assume that a block proposer is either honest or adversarial. The honest proposers, denoted by $\mathcal{H}$, always follow the honest strategy. The adversarial proposers are under the control of an attacker denoted by $\mathcal{A}$ and can deviate from the honest strategy. We denote by α and η the attacker's stake share and communication capability, respectively.

2.1 Intuition behind nothing-at-stake selfish proposing

We first explain how nothing-at-stake can lead to a more profitable selfish proposing attack. In PoW protocols, a miner should specify the content of a block including a reference to its parent prior to the start of the mining process. After a block is successfully mined in PoW protocols, modifying its parent block requires a significant amount of computational work since the miner needs to mine an entirely new block to include a different parent. However, in PoS protocols, a proposer can still change the content of a block after it is proposed and before it is published. In fact, once a proposer is eligible to propose a block in a PoS

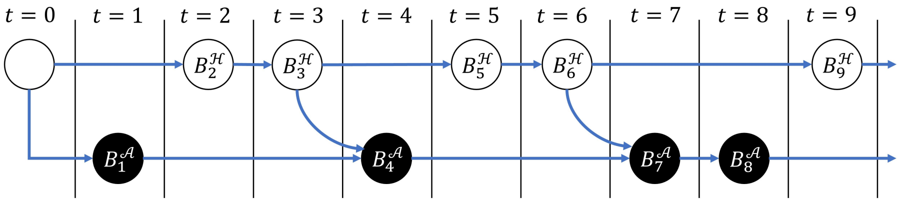


Fig. 1: Nothing-at-stake selfish proposing

protocol, he has the freedom to change the parent of the block as long as it is not published without any further effort.

To get an intuition that how the nothing-at-stake phenomenon can fortify selfish proposing, consider the picture depicted in Figure 1. In this figure, which is based on the single-proposer model introduced in Definition 11, the block proposed by proposer P at time slot t is denoted by B_t^P. The block proposer P can be either honest or adversarial, namely $P \in \{\mathcal{H}, \mathcal{A}\}$. As can be seen in Figure 1, the honest chain has a 2-block lead over the adversarial chain. We assume that none of the adversarial blocks in Figure 1 is published. In a similar situation in PoW protocols, the attacker should decide whether to give up on his fork and continue mining on top of the honest chain (block $B_9^{\mathcal{H}}$) or to continue mining on top of his secret chain. On the one hand, if the attacker chooses to give up, then his 4 adversarial blocks get orphaned. On the other hand, if the attacker chooses to continue mining on top of his secret chain that is 2 blocks behind, he has a relatively low chance of catching up the public chain and may risk losing his future blocks. In LC-PoS protocols, however, besides these two actions, the attacker can perform other reasonable actions. For the scenario depicted in Figure 1, we explain two actions that a selfish proposer can only take in LC-PoS protocols. Assume the attacker needs to decide on an action after that the honest block $B_9^{\mathcal{H}}$ is proposed at time slot $t = 9$. As the first reasonable action, the attacker can give up on block $B_1^{\mathcal{A}}$, change the parent of $B_4^{\mathcal{A}}$ to $B_3^{\mathcal{H}}$, and generate a new fork that includes 3 adversarial blocks. In this case, the length of both adversarial and honest chains gets equal to 3, and the attacker has a higher chance to orphan the honest chain. As the second reasonable action, the attacker can give up on blocks $B_1^{\mathcal{A}}$ and $B_4^{\mathcal{A}}$, change the parent of $B_7^{\mathcal{A}}$ to $B_6^{\mathcal{H}}$, and generate a new fork that includes 2 adversarial blocks. In this case, the adversarial chain has a 1-block lead over the honest chain. These two actions are examples of a broader set of actions, which we refer to as "`jump`".

2.2 Set of actions for nothing-at-stake selfish proposing

In this section, we introduce the set of actions for selfish proposing in LC-PoS protocols. We assume that the PoS protocol is accompanied by a slashing mechanism, i.e., a malicious proposer who publishes two or more contradicting blocks can get slashed by the other proposers. Therefore, a selfish proposer can publish at most one block in each slot. Note that when discussing the attacker's actions, it is important to distinguish between the terms "proposing" and "publishing" blocks. The term that the attacker proposes a block at time slot t indicates that the attacker is eligible to create a block at time slot t; however, the content of the block may not be finalized yet. The term that the attacker publishes a block at time slot t indicates that the content of the block proposed at time slot t is finalized and honest proposers are aware of that. Regarding honest proposers, the terms "proposing" and "publishing" can be used interchangeably since an honest proposer publishes its block immediately after it is proposed.

A brief overview of selfish mining in PoW protocols is presented in Appendix A.1. In our model, we assume that the attacker works on a secret chain while the honest proposers work on a single public chain. Let t_0 be the time slot up to which both the attacker's chain and the public chain share the same subchain. This indicates that after time slot t_0, the honest and adversarial chains have diverged. We refer to the attacker's chain (honest chain) proposed after t_0 as the adversarial fork (honest fork). Let $l^{\mathcal{A}}$ and $l^{\mathcal{H}}$ denote the length of the adversarial fork and the length of the honest fork, respectively. Additionally, let t_i for $i \geq 1$ represent the time slot at which the i^{th} block is proposed in the honest fork. The actions are a combination of two subactions and have the format $(\texttt{subaction}_1, \texttt{subaction}_2)$, where

$$\begin{aligned} &\texttt{subaction}_1 \in \left\{\texttt{jump}_i \;\middle|\; i \in \{0, 1, 2, \cdots, l^{\mathcal{H}}\}\right\} \ , \\ &\texttt{subaction}_2 \in \{\texttt{override}, \texttt{match}, \texttt{wait}\} \ . \end{aligned} \tag{1}$$

$\texttt{jump}_i$: The subaction represents that the attacker specifies the starting point of the fork. If $i = 0$, the attacker continues working on top of the current fork. For $i \geq 1$, the attacker gives up on the blocks in the adversarial fork proposed before including time slot t_i, adopts the honest fork up to including time slot t_i, and generates a new fork on top of the honest block proposed at time slot t_i. In this case, the new adversarial (honest) fork includes all the adversarial (honest) blocks proposed after the time slot t_i. Note that:

- For $i = 0$, the subaction is always feasible.
- For $i \geq 1$, the subaction is feasible if all the adversarial blocks proposed after t_i are unpublished.

If no adversarial block is proposed after the last block in the honest fork, one can consider $\texttt{jump}_{l^{\mathcal{H}}}$ as the action `adopt` employed in PoW blockchains, where the attacker gives up on his private chain and continues mining on top of the longest public chain. Once the starting point of the fork is specified using action $\texttt{jump}_i$, the attacker decides on which of the following subactions to perform on the new fork. Note that values of $l^{\mathcal{H}}$ and $l^{\mathcal{A}}$ may get updated after applying action $\texttt{jump}_i$.

`override`: The subaction represents that the attacker publishes his secret fork whose length is one block longer than the honest fork. This subaction is feasible if $l^{\mathcal{A}} > l^{\mathcal{H}}$.

`match`: The subaction represents that the attacker publishes his secret fork whose length is equal to the honest fork immediately after an honest block is published. This subaction is feasible if $l^{\mathcal{A}} \geq l^{\mathcal{H}}$ and $l^{\mathcal{A}} \neq 0$.
`wait`: The subaction represents that the attacker continues proposing new blocks in the specified fork. This subaction is always feasible.

2.3 Nothing-at-stake selfish proposing with perfect randomness

In this section, we introduce a selfish proposing strategy suitable for longest-chain PoS protocols with perfect randomness, denoted by $\pi^{\text{S-PR}}$. Here, the perfect randomness implies that the random seed of each time slot is revealed at the start of the time slot, and consequently, no proposer can predict the slot seed prior to the start of the slot. This indicates that similar to selfish mining in PoW, a selfish proposer in an LC-PoS protocol with perfect randomness does not enjoy predictability. Thus, the only difference between selfish mining in PoW and selfish proposing in LC-PoS with perfect randomness is the nothing-at-stake phenomenon. The strategy $\pi^{\text{S-PR}}$ and a comprehensive theoretical analysis of the block ratio gained by following this strategy are presented in Appendix D. Note that strategy $\pi^{\text{S-PR}}$ is not the optimal strategy that a selfish proposer can follow in an LC-PoS protocol with perfect randomness. However, we prove in Appendix D that for some range of the stake share, the introduced strategy can dominate the optimal selfish mining strategy in PoW [22] and the nothing-at-stake selfish mining strategy introduced in [13]. Later in Section 4.2, we introduce a deep Q learning-based method to obtain the near-optimal selfish proposing strategy in an LC-PoS protocol with perfect randomness.

In the following, we only review the amount of profitable stake share threshold (introduced in Definition 23) for a selfish proposer following the strategy $\pi^{\text{S-PR}}$. In PoW protocols, there is a famous bound that indicates the profitable mining share threshold of an attacker with communication capability $\eta = 0.5$ is equal to 0.25 [12,22]. According to the analysis presented in Appendix D, the profitable stake share threshold of a selfish proposer with communication capability $\eta = 0.5$ in an LC-PoS protocol with perfect randomness is less than 0.24198. This indicates that a selfish proposer whose stake share is greater than or equal to 0.24198 can increase his block ratio by deviating from the honest strategy in an LC-PoS protocol with perfect randomness.

We also show in Appendix D that the profitable stake share threshold of a selfish proposer with communication capability $\eta = 0$ in an LC-PoS protocol with perfect randomness is less than 0.323606. This threshold is lower than the profitable stake (mining) share threshold achieved by following the optimal PoW strategy and the nothing-at-stake selfish mining strategy introduced in [13], which are almost equal to 0.329 and 0.324718, respectively.

3 Selfish proposing in predictable LC-PoS protocols

In this section, we aim to assess the effect of proposer predictability in fortifying the selfish proposing attack. Note that a predictable PoS protocol can be either semi-predictable (Definition 14) or full-predictable (Definition 15). We discuss how malicious proposers can use information regarding the upcoming block proposers in both semi and full-predictable environments to apply a successful selfish proposing attack with a lower risk of losing their blocks.

In a full-predictable protocol, all the proposers including the attacker can predict the proposing sequence, the definition of which is presented in Definition 16. Being aware of the proposing sequence in advance can help the attacker modify his strategy to increase the selfish proposing block ratio. Consider a simple example in a full-predictable PoS protocol, where the attacker is eligible to propose a block, e.g., $B^{\mathcal{A}}$, at time slot $t = 1$. The attacker should decide whether he wants to publish block $B^{\mathcal{A}}$ immediately or keep it secret in the hope of orphaning some honest blocks. Due to the full predictability, the attacker can predict the block proposers of upcoming slots $t > 1$ at time slot $t = 1$. Assume the scenario in which the attacker is lucky and is responsible for proposing another block at time slot $t = 2$. In this scenario, since the attacker is sure that he can use his two consecutive blocks of slots $t = 1$ and $t = 2$ to orphan at least one honest block, he would keep block $B^{\mathcal{A}}$ secret. As the second scenario, assume the case in which the attacker is unlucky as he is not among the block proposers in the near subsequent slots. In this case, especially if the attacker does not enjoy a high amount of communication capability, the attacker would publish block $B^{\mathcal{A}}$ since he knows if he keeps the block secret, the block would get orphaned with a high probability.

In a semi-predictable protocol, although the attacker cannot predict a full list of upcoming block proposers, he is still capable of predicting those time slots in which he is responsible for proposing a block. The distance between these adversarial time slots can help the attacker to increase the selfish proposing success rate. Consider the same example where the attacker is eligible to propose a block, e.g., $B^{\mathcal{A}}$, at time slot t. Assume the scenario in which the attacker knows that he can propose his next block at time slot $t + \delta$, where δ is relatively small. In this scenario, the attacker may keep the block $B^{\mathcal{A}}$ secret due to the low probability of honest proposers proposing a block within the duration of δ slots. However, in the scenario where the attacker knows he has no chance to propose a block for a long period after time slot t, he would publish block $B^{\mathcal{A}}$ immediately.

3.1 Selfish proposing in full-predictable LC-PoS protocols

This section presents theoretical methods to analyze selfish proposing in full-predictable LC-PoS protocols for an attacker with communication capability $\eta = 0$. In this section, we assume that the underlying LC-PoS protocol is an infinite-full-predictable protocol, where the attacker can predict block proposers for an infinite number of future time slots in advance. The authors in [7] presented a selfish proposing strategy for a selfish proposer with communication capability $\eta = 0$ in a full-predictable environment. We denote this strategy by $\pi^{\mathtt{SP1}}$. Despite the introduction of strategy $\pi^{\mathtt{SP1}}$, the authors did not provide any theoretical analysis for this strategy in [7]. In this section, we first introduce strategy $\pi^{\mathtt{SP1}}$ and then calculate the attacker's block ratio under this strategy.

Then, we introduce another strategy denoted by $\pi^{\texttt{SP2}}$, which we prove can dominate strategy $\pi^{\texttt{SP1}}$. Finally, we show that none of the strategies $\pi^{\texttt{SP1}}$ and $\pi^{\texttt{SP2}}$ are optimal and present an optimal strategy for a selfish proposer with communication capability $\eta = 0$ in full-predictable LC-PoS protocols. To present the selfish proposing strategies, we use the concept of the "longest dominant chain" defined in Definition 19. Note that $\texttt{LDC}(t)$ and $L^{\texttt{LDC}}(t)$ denote the longest dominant chain of time slot t and its corresponding length. For presenting the following strategies, we assume that the LC-PoS protocol is compatible with the single-proposer model. Strategy $\pi^{\texttt{SP1}}$ is defined as follows:

(Strategy $\pi^{\texttt{SP1}}$). At each time slot t:

- If $L^{\texttt{LDC}}(t) = 0$, the attacker does not fork the honest chain and moves to the next time slot.
- If $L^{\texttt{LDC}}(t) > 0$, where $\texttt{LDC}(t)$ ends at time slot t', the attacker forks the honest chain and keeps the adversarial chain secret within the time slot interval $[t, t']$. At the end of time slot t', the attacker publishes $\texttt{LDC}(t)$ and orphans $L^{\texttt{LDC}}(t) - 1$ honest blocks proposed within the interval $[t, t']$.

Theorem 1. *Let α denote the stake share of an attacker $\mathcal{A}$. The block ratio of attacker $\mathcal{A}$ under strategy π^{SP1} can be obtained as follows:*

$$\texttt{BlkRatio}_{\mathcal{A}}(\pi^{SP1}) = \frac{\alpha}{1 - 2\alpha^2} \tag{2}$$

The proof of Theorem 1 is presented in Appendix E.

We introduce another strategy denoted by $\pi^{\texttt{SP2}}$, which can dominate startegy $\pi^{\texttt{SP1}}$.

(Strategy $\pi^{\texttt{SP2}}$). At each time slot t:

- If the slot proposer is honest, the attacker does not fork the honest chain and moves to the next time slot.
- If the slot proposer is the attacker himself, where $\texttt{LDC}(t)$ ends at time slot t', the attacker forks the honest chain and keeps the adversarial chain secret within the time slot interval $[t, t']$. At the end of time slot t', the attacker publishes $\texttt{LDC}(t)$ and orphans $L^{\texttt{LDC}}(t) - 1$ honest blocks proposed within the interval $[t, t']$.

Theorem 2. *Let α denote the stake share of an attacker $\mathcal{A}$. The block ratio of attacker $\mathcal{A}$ under strategy π^{SP2} can be obtained as follows:*

$$\texttt{BlkRatio}_{\mathcal{A}}(\pi^{SP2}) = \frac{\alpha(1 - \alpha(1 - \alpha))}{(1 - \alpha)^2(1 + \alpha)} \tag{3}$$

The proof of Theorem 2 is presented in Appendix F.

For all $0 < \alpha < 0.5$, $\texttt{BlkRatio}_{\mathcal{A}}(\pi^{\texttt{SP2}}) > \texttt{BlkRatio}_{\mathcal{A}}(\pi^{\texttt{SP1}})$. By following one of the strategies $\pi^{\texttt{SP1}}$ or $\pi^{\texttt{SP2}}$, none of the adversarial blocks get orphaned during the selfish proposing attack, which is impossible in PoW selfish mining attack. Moreover, in both of these two strategies, the attacker always waits until he can generate the longest possible adversarial fork. By doing so, the attacker

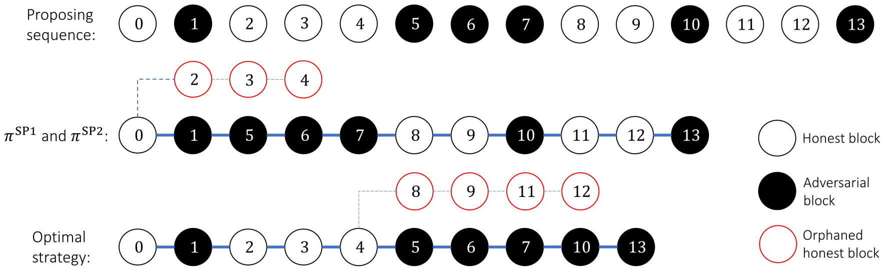


Fig. 2: Block ratio of selfish proposing strategies in a full-predictable protocol

aims to maximize the number of honest blocks that get orphaned during the attack. Although strategies $\pi^{\texttt{SP1}}$ or $\pi^{\texttt{SP2}}$ dominate the optimal PoW selfish mining strategy, they are not the optimal strategy that an attacker with communication capability $\eta = 0$ can follow in a full-predictable LC-PoS protocol. Consider the proposing sequence depicted in Figure 2. Following both strategies $\pi^{\texttt{SP1}}$ and $\pi^{\texttt{SP2}}$, the attacker's block ratio would be equal to $6/11$. However, the optimal block ratio that the attacker can achieve is equal to $6/10$. In the optimal scenario, the number of adversarial blocks that get added to the main chain is the same as that of when following $\pi^{\texttt{SP1}}$ or $\pi^{\texttt{SP2}}$. However, by following the optimal strategy, the attacker manages to orphan a greater number of honest blocks. The example in Figure 2 shows that to achieve the optimal block ratio, the attacker forking the public chain at time slot t should not always wait until $\texttt{LDC}(t)$ is generated. In fact, the attacker should sometimes publish the fork at an earlier time slot to create a better orphaning opportunity at future slots. To introduce the optimal strategy, we first define the checkpoint slot.

Definition 1 (Checkpoint slot). *An honest time slot t is called a checkpoint, if there is no adversarial time slot t', where $t' < t$, such that its longest dominant chain, i.e., $\texttt{LDC}(t')$, ends at a time slot t'' with $t'' \geq t$.*

An honest time slot t being a checkpoint implies that the honest block proposed in slot t cannot be orphaned by an adversarial fork initiated at a time slot earlier than t. Our checkpoint definition is similar to the Nakamoto block definition introduced in [11]. In the following, we introduce the selfish proposing optimal strategy for an attacker with communication capability $\eta = 0$ under the full-predictable model, which is denoted by $\pi^{\texttt{0-SP}}$. Let $\texttt{Seq}$ denote the proposing sequence within the interval $[0, \infty)$. In strategy $\pi^{\texttt{0-SP}}$, the attacker first needs to find all the checkpoint time slots denoted by $\{t_1, t_2, \cdots\}$. Let subsequence $\texttt{Seq}_i$ be a subset of proposing sequence $\texttt{Seq}$ that corresponds to time slot interval $T_i = (t_i, t_{i+1})$, i.e., $\texttt{Seq}_i = \texttt{Seq}(T_i)$. In Appendix H, we prove that the optimal strategy of selfish proposing in each of these subsequences is independent of strategies followed in the other subsequences. Therefore, to find the optimal selfish proposing strategy in the whole proposing sequence $\texttt{Seq}$, the attacker only needs to find the optimal strategy for each individual subsequence. The optimal selfish proposing strategy for an attacker with communication capability $\eta = 0$

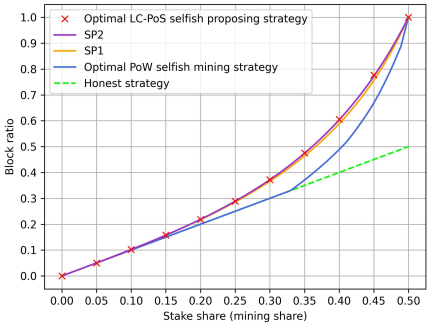


Fig. 3: Block ratio comparison in a full-predictable protocol ($\eta = 0$)

in a full-predictable model satisfies two properties: (1) it ensures that none of the adversarial blocks get orphaned, and (2) it maximizes the number of honest blocks that get orphaned. In an ideal scenario where $\eta = 1$, an attacker should be able to orphan one honest block per each adversarial block. However, since the attacker's communication capability is equal to 0, for each adversarial fork comprising of n adversarial blocks, the attacker can orphan at most $n-1$ honest blocks, which is one honest block less than the ideal scenario. This indicates that when $\eta = 0$, each adversarial fork results in the inclusion of an additional honest block in the main chain compared to the ideal scenario. Therefore, to increase the number of orphaned honest blocks in each subsequence $\texttt{Seq}_i$, the attacker should follow the strategy that minimizes the number of adversarial forks in that subsequence.

(Strategy $\pi^{\texttt{0-SP}}$). Find all the checkpoint time slots. For each pair of two consecutive checkpoint slots t_1 and t_2, use Algorithm 1 presented in Appendix G to find the optimal strategy.

In Appendix H, we prove that strategy $\pi^{\texttt{0-SP}}$ is the optimal strategy that an attacker with communication capability $\eta = 0$ can follow in full-predictable LC-PoS protocols. In Figure 3, we compare the block ratios achieved by strategies $\pi^{\texttt{SP1}}$, $\pi^{\texttt{SP2}}$, and $\pi^{\texttt{0-SP}}$ within a full-predictable LC-PoS protocol with the block ratios of the honest strategy and the optimal PoW selfish mining strategy presented in [22]. A more precise comparison among the block ratios of strategies $\pi^{\texttt{SP1}}$, $\pi^{\texttt{SP2}}$, and $\pi^{\texttt{0-SP}}$ is presented in Table 1.

Table 1: Block ratio comparison in a full-predictable protocol ($\eta = 0$)

stake share	0.3	1/3	0.35	0.4	0.45
$\pi^{\texttt{SP1}}$	0.3658	0.4285	0.4635	0.5882	0.7563
$\pi^{\texttt{SP2}}$	0.3720	0.4375	0.4740	0.6031	0.7720
$\pi^{\texttt{0-SP}}$	0.3722	0.4383	0.4753	0.6054	0.7766

4 Deep Q-Learning to analyze selfish proposing

Thus far, we have analyzed the selfish proposing attack in simplified LC-PoS environments such as LC-PoS protocols with perfect randomness and full-predictable LC-PoS protocols. In this section, we use Deep Q-Learning (DQL) to analyze the selfish proposing attack in more complicated scenarios, such as semi-predictable environments, and obtain the near-optimal selfish proposing strategy for all proposers with varying stake shares and communication capabilities.

4.1 DQL implementation in blockchain environments

A brief overview of deep Q-learning is presented in A.2. In a DQL implementation that is specifically designed for blockchains, each interaction experience with the blockchain environment has the following format:

$$e_t = (s_t, a_t, s_{t+1}, r_t^{\mathcal{A}}, r_t^{\mathcal{H}}) \ , \tag{4}$$

where s_t is the state visited at step t, a_t is the action performed by the attacker at step t, s_{t+1} is the subsequent state resulting from taking action a_t at state s_t, and $r_t^{\mathcal{A}}$ and $r_t^{\mathcal{H}}$ are the rewards the attacker and the honest proposers received due to moving from state s_t to state s_{t+1} under action a_t, respectively. For each pair of state s and action a, we define two Q values: the adversarial Q value denoted by $Q_\pi^{\mathcal{A}}(s,a)$ and the honest Q value denoted by $Q_\pi^{\mathcal{H}}(s,a)$, where $Q_\pi^{\mathcal{A}}(s,a)$ and $Q_\pi^{\mathcal{H}}(s,a)$ represent the expected discounted accumulated reward for the attacker and honest proposers, respectively, resulting from taking action a in state s under strategy π.

$$\begin{aligned} Q_\pi^{\mathcal{A}}(s,a) &= \mathbf{E}\big[\sum_{k=0}^{\infty} \gamma^k r_{t+k}^{\mathcal{A}} | s_t = s, a_t = a, \pi\big], \ \text{and} \\ Q_\pi^{\mathcal{H}}(s,a) &= \mathbf{E}\big[\sum_{k=0}^{\infty} \gamma^k r_{t+k}^{\mathcal{H}} | s_t = s, a_t = a, \pi\big] \ . \end{aligned} \tag{5}$$

DQL uses two neural networks to estimate the Q-values. The first network is the Q-network, denoted by $Q_{\texttt{estimate}}$, which is responsible for estimating the Q-values. The weights of the Q-network are constantly getting updated after each iteration. The second network is the target network denoted by $Q_{\texttt{target}}$, which is responsible for calculating the target Q-values. The parameters of the target network are updated with respect to the Q-network after a pre-determined number of steps. Both the Q-network and the target network take a state as their input, which indicates that the number of input nodes is equal to the dimension size of the state space. The number of output nodes in these networks is equal to the total number of available actions, with each output node representing the estimated or target Q-value for the given state under a specific action.

To calculate the block ratio of a selfish proposer in a blockchain environment, both honest and adversarial Q-values need to be estimated. Therefore, in total, we need to implement four separate neural networks: the adversarial $Q_{\texttt{estimate}}$ and $Q_{\texttt{target}}$ as well as the honest $Q_{\texttt{estimate}}$ and $Q_{\texttt{target}}$.

The agent, which is the selfish proposer, follows the ϵ-greedy algorithm to select the action at each state. As explained in [23], to select the greedy action, we define an objective function as follows:

$$O(s,a) = \frac{Q^{\mathcal{A}}(s,a)}{Q^{\mathcal{A}}(s,a) + Q^{\mathcal{H}}(s,a)} \ . \tag{6}$$

Here, we explain the process of training the adversarial networks. The same process is used to train the honest networks. Assume $e_t = (s_t, a_t, s_{t+1}, r_t^{\mathcal{A}}, r_t^{\mathcal{H}})$ represents a state transition experience according to which we want to train the adversarial networks. First, the estimated Q-value is calculated as $Q^{\mathcal{A}}_{\texttt{estimate}}(s_t, a_t)$. Then, the target network is used to calculate the target Q-value, which is equal to $r_t^{\mathcal{A}} + \gamma Q^{\mathcal{A}}_{\texttt{target}}(s_{t+1}, a')$. Here, a' is the action that maximizes the objective function at state s_{t+1}, i.e., $a' = \texttt{argmax}_a O(s_{t+1}, a)$. Once the estimated and the target Q-values are calculated, a loss function is used to measure the difference between these two values. The loss is then backpropagated to the adversarial Q-network to update the network parameters. For more information regarding the implementation part, readers are referred to [19].

In the following, we explain the state format for the implementation of the selfish proposing attack in different LC-PoS blockchain environments.

4.2 DQL implementation of nothing-at-stake selfish proposing

In our implementation, each state in LC-PoS blockchains with perfect randomness has the following format:

$$s_t = (l_{\mathcal{A}}, l_{\mathcal{H}}, n_{\mathcal{A}}^{l_{\mathcal{H}}-k_1+1}, \cdots, n_{\mathcal{A}}^{l_{\mathcal{H}}-1}, n_{\mathcal{A}}^{l_{\mathcal{H}}}, \texttt{publish}, \texttt{match}, \texttt{latest}) \ , \tag{7}$$

where $l_{\mathcal{A}}$ represents the adversarial fork length, $l_{\mathcal{H}}$ represents the honest fork length, $n^i_{\mathcal{A}}$ represents the number of adversarial blocks proposed after the time slot at which the i^{th} block in the honest fork is proposed, `publish` represents the number of blocks in the adversarial fork that are published to the honest network, `match` represents whether the action `match` is active or not, and `latest` represents whether the latest block is proposed by the honest proposers or not. Note that this state representation fits both single-proposer and multi-proposer models. The parameter k_1 represents the number of blocks in the honest fork for which information regarding their subsequent time slots is stored in the state.

As mentioned in Section 2.2, each action consists of two subactions, where the first subaction is `jump`, and the second one is chosen among 3 different possibilities: `override`, `match`, and `wait`. The number of possibilities for subaction `jump` gets limited by the value of k_1. The state stores information regarding the current fork as well as information regarding the latest k_1 honest blocks. Therefore, the agent can either continue working on top of the current fork or jump on top of one of those k_1 honest blocks and create a new fork. As a result, the total number of available actions that the agent can take is equal to $3(k_1 + 1)$.

This implies that the implemented neural networks should have $3(k_1+1)$ outputs, where each output represents the Q-value of the input state under one of the available actions. It is worth mentioning that each state has its own set of possible actions, which is a subset of the whole available actions.

4.3 DQL implementation of full-predictable selfish proposing

In our implementation, each state in full-predictable LC-PoS blockchains has the following format:

$$\begin{aligned} s_t =&(l_{\mathcal{A}}, l_{\mathcal{H}}, n_{\mathcal{A}}^{l_{\mathcal{H}}-k_1+1}, \cdots, n_{\mathcal{A}}^{l_{\mathcal{H}}-1}, n_{\mathcal{A}}^{l_{\mathcal{H}}}, \texttt{publish}, \texttt{match}, \texttt{latest}, \\ &h^1, h^2, \cdots, h^{k_2}) \ . \end{aligned} \tag{8}$$

In the state representation above, the first part is similar to the state representation of LC-PoS protocols with perfect randomness. The second part contains information regarding the proposers of future slots. Due to the full predictability, we can assume each slot has exactly one proposer. k_2 denotes the number of future slots whose proposers can be predicted at the current slot. h^i represents the height of the chain in the i^{th} upcoming time slot. Let h^0 (the height of the current slot) be defined to equal 0. In this case, h^i can be obtained as follows:

$$h^i = \begin{cases} h^{i-1}+1 & \text{if the } i^{\text{th}} \text{ upcoming time slot is adversarial.} \\ h^{i-1}-1 & \text{if the } i^{\text{th}} \text{ upcoming time slot is honest.} \end{cases} \tag{9}$$

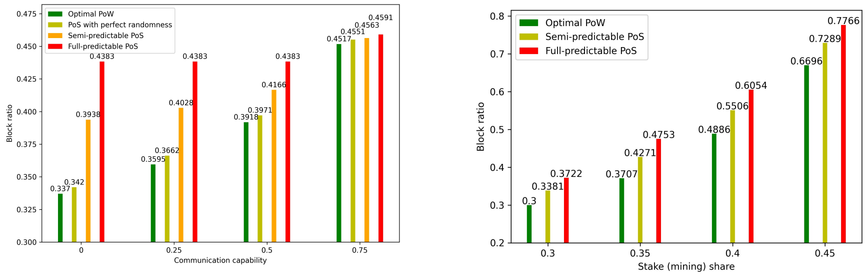
4.4 DQL implementation of semi-predictable selfish proposing

In our implementation, each state in semi-predictable LC-PoS blockchains has the following format:

$$\begin{aligned} s_t =&(l_{\mathcal{A}}, l_{\mathcal{H}}, n_{\mathcal{A}}^{l_{\mathcal{H}}-k_1+1}, \cdots, n_{\mathcal{A}}^{l_{\mathcal{H}}-1}, n_{\mathcal{A}}^{l_{\mathcal{H}}}, \texttt{publish}, \texttt{match}, \texttt{latest}, \\ &\Delta_{\mathcal{A}}^1, \Delta_{\mathcal{A}}^2, \cdots, \Delta_{\mathcal{A}}^{k_3}) \ . \end{aligned} \tag{10}$$

In the state representation above, the first part is similar to the state representation of LC-PoS protocols with perfect randomness. The second part contains information regarding the future adversarial slots. k_3 represents the number of future adversarial blocks whose proposing time slot can be predicted at the current time slot. $\Delta_{\mathcal{A}}^i$ represents the difference between the slot number at which the i^{th} future adversarial block will be proposed and the current slot number.

In Cardano, which can be considered the most well-known LC-PoS protocol, proposers use a verifiable random function (VRF) to determine the future slot proposers. Since the randomness used in the Cardano VRF function is extracted from the previous epoch, Cardano can be characterized as a semi-predictable LC-PoS protocol. Our DQL implementation to analyze the selfish proposing attack within various environments is presented in [1]. The graph depicted in Figure 4a shows a comparison of the block ratios achieved by a selfish proposer



(a) Stake (mining) share $\alpha = \frac{1}{3}$

(b) Communication capability $\eta = 0$

Fig. 4: Block ratio comparison

(miner) with a stake (mining) share equal to 1/3 under various environments and different communication capabilities. As can be seen in Figure 4a, for selfish proposers with limited communication capabilities, the predictability of future slot proposers can lead to a significant increase in the selfish proposing block ratio. The graph depicted in Figure 4b shows a comparison of the block ratios achieved by a selfish proposer (miner) with communication capability $\eta = 0$ under various environments and different stake (mining) shares. More details regarding implementation are presented in Appendix I.1.

To move towards a real-world scenario, in Appendix I.2, we present an implementation for the selfish proposing attack within an environment that encompasses the location of proposer nodes. The implementation of network node locations can help us analyze the selfish proposing attack while accounting for the complexities of real-world network conditions.

5 Conclusion

In this paper, we analyzed the selfish proposing attack in LC-PoS protocols, using both theoretical and implementation-based methods. The primary takeaway from this paper is that selfish proposing in LC-PoS protocols is more destructive compared to selfish mining in PoW protocols. Our analysis showed that while the nothing-at-stake phenomenon has a slight effect on a selfish proposer's block ratio, predictability can significantly increase it, especially when the attacker's communication capability is relatively low. In addition to providing theoretical analysis for selfish proposing in simple environments, such as full-predictable LC-PoS protocols and LC-PoS protocols with perfect randomness, we have introduced a deep Q-learning-based tool that enables us to analyze selfish proposing attacks in more complex environments, such as semi-predictable LC-PoS protocols. Using this tool, we can analyze the selfish proposing attack in a realistic environment, considering specifications such as node locations and the transaction pool structure.

References

1. Implementaion of the selfish proposing attack. https://github.com/RoozbehSrnch/Selfish-Proposing-Attack
2. Arulkumaran, K., Deisenroth, M.P., Brundage, M., Bharath, A.A.: Deep reinforcement learning: A brief survey. IEEE Signal Processing Magazine **34**(6), 26–38 (2017)
3. Bagaria, V., Dembo, A., Kannan, S., Oh, S., Tse, D., Viswanath, P., Wang, X., Zeitouni, O.: Proof-of-stake longest chain protocols: Security vs predictability. In: Proceedings of the 2022 ACM Workshop on Developments in Consensus. pp. 29–42 (2022)
4. Bano, S., Sonnino, A., Al-Bassam, M., Azouvi, S., McCorry, P., Meiklejohn, S., Danezis, G.: Sok: Consensus in the age of blockchains. In: Proceedings of the 1st ACM Conference on Advances in Financial Technologies. pp. 183–198 (2019)
5. Bar-Zur, R., Abu-Hanna, A., Eyal, I., Tamar, A.: Werlman: to tackle whale (transactions), go deep (rl). In: Proceedings of the 15th ACM International Conference on Systems and Storage. pp. 148–148 (2022)
6. Bar-Zur, R., Dori, D., Vardi, S., Eyal, I., Tamar, A.: Deep bribe: Predicting the rise of bribery in blockchain mining with deep rl. Cryptology ePrint Archive (2023)
7. Brown-Cohen, J., Narayanan, A., Psomas, A., Weinberg, S.M.: Formal barriers to longest-chain proof-of-stake protocols. In: Proceedings of the 2019 ACM Conference on Economics and Computation. pp. 459–473 (2019)
8. Chen, H.: Interesting series associated with central binomial coefficients, catalan numbers and harmonic numbers. J. Integer Seq. **19**(1), 16–1 (2016)
9. Daian, P., Pass, R., Shi, E.: Snow white: Robustly reconfigurable consensus and applications to provably secure proof of stake. In: Financial Cryptography and Data Security: 23rd International Conference, FC 2019, Frigate Bay, St. Kitts and Nevis, February 18–22, 2019, Revised Selected Papers 23. pp. 23–41. Springer (2019)
10. David, B., Gaži, P., Kiayias, A., Russell, A.: Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In: Advances in Cryptology–EUROCRYPT 2018: 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29-May 3, 2018 Proceedings, Part II 37. pp. 66–98. Springer (2018)
11. Dembo, A., Kannan, S., Tas, E.N., Tse, D., Viswanath, P., Wang, X., Zeitouni, O.: Everything is a race and nakamoto always wins. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. pp. 859–878 (2020)
12. Eyal, I., Sirer, E.G.: Majority is not enough: Bitcoin mining is vulnerable. Communications of the ACM **61**(7), 95–102 (2018)
13. Ferreira, M.V., Weinberg, S.M.: Proof-of-stake mining games with perfect randomness. In: Proceedings of the 22nd ACM Conference on Economics and Computation. pp. 433–453 (2021)
14. Göbel, J., Keeler, H.P., Krzesinski, A.E., Taylor, P.G.: Bitcoin blockchain dynamics: The selfish-mine strategy in the presence of propagation delay. Performance evaluation **104**, 23–41 (2016)
15. Goodman, L.: Tezos—a self-amending crypto-ledger white paper. URL: https://www. tezos. com/static/papers/white paper. pdf **4**, 1432–1465 (2014)
16. Grunspan, C., Pérez-Marco, R.: On profitability of selfish mining. arXiv preprint arXiv:1805.08281 (2018)

17. Kiayias, A., Russell, A., David, B., Oliynykov, R.: Ouroboros: A provably secure proof-of-stake blockchain protocol. In: Annual international cryptology conference. pp. 357–388. Springer (2017)
18. Li, Y.: Deep reinforcement learning: An overview. arXiv preprint arXiv:1701.07274 (2017)
19. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. nature **518**(7540), 529–533 (2015)
20. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. Decentralized business review (2008)
21. Neu, J., Tas, E.N., Tse, D.: Two more attacks on proof-of-stake ghost/ethereum. In: Proceedings of the 2022 ACM Workshop on Developments in Consensus. pp. 43–52 (2022)
22. Sapirshtein, A., Sompolinsky, Y., Zohar, A.: Optimal selfish mining strategies in bitcoin. In: Financial Cryptography and Data Security: 20th International Conference, FC 2016, Christ Church, Barbados, February 22–26, 2016, Revised Selected Papers 20. pp. 515–532. Springer (2017)
23. Wang, T., Liew, S.C., Zhang, S.: When blockchain meets ai: Optimal mining strategy achieved by machine learning. International Journal of Intelligent Systems **36**(5), 2183–2207 (2021)
24. Zur, R.B., Eyal, I., Tamar, A.: Efficient mdp analysis for selfish-mining in blockchains. In: Proceedings of the 2nd ACM Conference on Advances in Financial Technologies. pp. 113–131 (2020)

A Preliminaries

A.1 Selfish mining in PoW

In selfish mining, once a malicious miner mines a new block, he keeps the block secret and does not publish it immediately. Therefore, the other miners will continue mining on top of the chain that is no longer the longest chain. By performing selfish mining, some part of honest mining power gets wasted, and the attacker can increase his relative revenue. However, the selfish miner who keeps his block secret cannot make sure that his block will be added to the main chain in the future since the honest chain may manage to orphan the selfish miner's secret chain. Therefore, it is of huge importance that the selfish miner follows a proper strategy in each state of the chain race [12]. The authors in [22], proposed an algorithm to find the optimal strategy for selfish mining in Bitcoin, which uses the Markov Decision Process (MDP) to find the best possible action in each state based on the attacker's mining power and communication capability. The proposed solution is not Bitocin-specific and can be applied to all other PoW-based blockchains. As mentioned in this paper, each state of selfish mining in PoW protocols can be represented using a tuple $(l_a, l_h, \texttt{fork})$, where l_a denotes the length of the attacker's chain, l_h is the length of the honest chain, and `fork` gives information regarding the miner of the latest block. The set of actions is composed of four different actions, which are `adopt`, `override`, `match`, and `wait`. `adopt` means the selfish miner leaves his secret chain and continues mining on top of the honest chain. `override` represents that the attacker publishes his secret chain that is longer than the honest chain. `match` means once the honest miners mine a new block, the attacker publishes a conflicting block with the same height. And finally, `wait` means that the attacker continues mining on top of his secret chain.

A.2 Deep Q-Learning

The goal of reinforcement learning, which is one of the fields of AI, is to produce an autonomous agent that interacts with the environment to learn the optimal policy, i.e., the policy that can maximize the long-term reward, through trial and error [2]. Q-learning is a reinforcement learning technique that utilizes Markov Decision Process (MDP) to find the optimal action for each state. At each time step t, the agent observes a state s_t and takes an action a_t from the action space. As a result of this action, the agent receives a scalar reward r_t and moves to the next step s_{t+1}, according to the environment's behavior. The discounted accumulated reward at time step t is defined as $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$, where $\gamma \in (0, 1]$ is the discount factor. In Q-learning, for each pair of state s and action a, we define a quality function $Q_\pi(s, a) = \mathbf{E}\big[R_t | s_t = s, a_t = a, \pi\big]$ that represents the expected discounted accumulated reward for taking action a in state s and then following policy π. An optimal state-action quality function is the maximum expected discounted accumulated reward achievable by any policy at state s for action a. To calculate the optimal state-action quality values, the agent stores the Q-value of each state-action pair in a table and uses the following equation to update the table after each iteration with the environment:

$$Q(s_t, a_t) = (1 - \beta)Q(s_t, a_t) + \beta[r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})] \ , \tag{11}$$

where $\beta \in (0, 1]$ is the learning rate [18]. For more details regarding Q-learning, readers are referred to [23,18]. One of the main limitations of Q-learning is that it is applicable only to environments dealing with a limited number of states and actions since storing the quality value for each pair of state-action is infeasible for high-dimensional settings. Deep learning enables reinforcement learning to scale to high-dimensional states and action spaces [2]. Deep Q-learning [19], introduced by Mnih et al. in 2015, uses neural networks rather than look-up tables to approximate the state-action quality values. The neural network takes a state as its input and outputs a state-action quality value for each action in the action space. To properly approximate the quality function, the neural network needs to be trained for a sufficient number of iterations. At each iteration, a loss function is used to compare the result of predicted output value $Q(s_t, a_t)$ with the target value $r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})$. The neural network weights get updated with respect to the loss function. Reinforcement learning was known to face instability and divergence issues when a neural network is used to implement the Q function. To solve these issues, the authors in [19] have used two main ideas, implementing the experience replay mechanism and using two separate neural networks to implement the Q function and the target value. In the experience replay mechanism, the agent stores its interaction experiences with the environment at each time step, i.e., $e_t = (s_t, a_t, r_t, s_{t+1})$, in a database. In each learning step, a mini-batch of these experiences is randomly selected and used to train the neural network. By using the experience replay, the network can learn the new experiences without forgetting about the old ones. Moreover, to reduce the correlation between the predicted output value and the target value, two separate neural networks are trained to approximate the value and the target functions. The weights of the value function get updated at every iteration; however, the parameters of the target function get updated with respect to the value function after a pre-determined number of steps.

B Proof of Stake

The model we use to analyze PoS protocols in this paper is built upon the model introduced in [7]. In this section, we first present a brief summary of the model introduced in [7] and then extend the model with some new definitions and notations.

B.1 Overview of the model introduced in [7]

In each PoS protocol, there exists a set of proposers who follow the protocol to maintain and extend a blockchain ledger. A blockchain ledger consists of a chain of blocks, where each block is cryptographically connected to its previous block and stores pieces of semantic information such as transactions. Time in PoS protocols is divided into a set of slots represented by t. Each proposer, which is denoted by P, owns one or more pairs of public and private keys and has a local view. The local view of proposer P at time slot t contains a set of chains that proposer P is aware of at the start of time slot t. The basic unit of stake in PoS protocols is called a coin and is denoted by c. In the blockchain ledger, each coin belongs to an account denoted by $\texttt{acct}$. Each account, which may contain multiple coins, is linked to a public key and belongs to the proposer who owns the corresponding private key of that public key. A transaction enables the transfer of a coin from one account to another account. The proposers are responsible for proposing new blocks to extend the blockchain ledger. We use notation $\texttt{Proposer}(B)$ to represent the proposer of block B. Each block B contains three main items: a pointer to its previous (parent) block denoted by $\texttt{Parent}(B)$, a time slot t_B at which the block is created, and a reference to an account $\texttt{acct}_B$ that witnesses the block validity [7]. A block may contain other items such as content that includes the transactions.

B.2 Formal model to define PoS protocols

Let $\Pi^{\texttt{PoS}}$ denote a PoS protocol. We use notations $\texttt{CH}$, and $\texttt{Head}(\texttt{CH})$ to represent a chain and its rightmost (most recent) block, respectively. $B \in \texttt{CH}$ represents that block B is included in $\texttt{CH}$. $\texttt{CH}_1 \subseteq \texttt{CH}_2$ represents that chain $\texttt{CH}_1$ is a subchain of chain $\texttt{CH}_2$. If $\texttt{CH}_1$ results from pruning the k rightmost blocks of $\texttt{CH}_2$, we use the notation $\texttt{CH}_1 = \texttt{Prune}_k(\texttt{CH}_2)$.

In PoS protocols, the eligibility of proposers for proposing new blocks is assessed based on the coins they own. The probability of a proposer being selected as the block proposer is proportional to the total number of coins available in the proposer's accounts. At the start of each slot, a set[2] of accounts is randomly selected, whose owners are eligible to propose a new block. The random selection of accounts enhances the blockchain's decentralization and strengthens its resistance against censorship. Once a proposer is eligible to propose a block, the proposer needs to follow the fork-choice rule to select the best chain available in his local view and then propose the new block on top of the selected chain. The fork-choice rule is essential to guarantee the blockchain's liveness [21]. Different PoS protocols may define different fork-choice rules, leading to distinct interpretations of the term "best chain". Here, we define the longest-chain fork-choice rule.

Definition 2 (Longest-chain). *Let $\texttt{chainSet}_t$ represent the set of valid chains available in the view of proposer P at time slot t. According to the longest-chain fork-choice rule, the best chain available in $\texttt{chainSet}_t$ is the chain whose length is greater compared to the other chains available in $\texttt{ChainSet}_t$.*

To precisely explain how proposers are selected, the concept of coin ownership should be discussed. Since multiple proposers may have owned a specific coin c throughout the history of blockchain, it is of huge importance to determine which proposer can claim to own the coin c at time slot t. We first define the term coin owner with respect to a specific chain.

Definition 3 (Coin owner with respect to chain $\texttt{CH}$). *For a given coin c and a chain $\texttt{CH}$, coin c belongs only to the last account it has visited in $\texttt{CH}$. If assuming that the last visited account is linked to a public key $\texttt{PubK}$, the owner of coin c with respect to chain $\texttt{CH}$, denoted by $\texttt{Owner}_{\texttt{CH}}(c)$, is the proposer who owns the corresponding private key of the public key $\texttt{PubK}$ [7].*

Additionally, we need to define the term coin owner with respect to a specific time slot, which is a different concept from Definition 3. For a given coin c at time slot t, we define two types of ownership: a live owner and a frozen owner. The coin live owner at time slot t can be considered as the most recent owner of the coin with respect to the best available chain at time slot t. Whereas the coin frozen owner at time slot t is the owner of the coin with respect to a specific subchain of the best available chain at time slot t.

Definition 4 (Coin live owner at slot t). *Let $\texttt{CH}^{best}$ be the best chain in the view of proposer P at time slot t. The live owner of coin c at time slot t in the view of proposer P is $\texttt{Owner}_{\texttt{CH}^{best}}(c)$.*

[2] Based on the PoS protocol, the account set corresponding to a specific time slot can be empty or contain one or more number of accounts.

PoS protocols define a stake reference subchain, which is a pruned version of the best chain, and use this subchain to specify frozen coin owners at each time slot.

Definition 5 (Stake reference subchain). *The stake reference subchain of chain* $\mathtt{CH}$ *at time slot* t*, which is denoted by* $\mathtt{RCH}_t^{stake}$*, is a subchain of* $\mathtt{CH}$ *resulting from pruning its* k *rightmost blocks, i.e.,* $\mathtt{RCH}_t^{stake} = \mathtt{Prune}_k\big(\mathtt{CH}\big)$*. Here,* k *can be calculated using a deterministic function* f_{prune}^{stake} *defined by protocol* Π^{PoS}*.*

The function $f_{\mathtt{prune}}^{\mathtt{stake}}$ takes as input a time slot t and a chin $\mathtt{CH}$ and outputs the number of blocks to be pruned. Function $f_{\mathtt{prune}}^{\mathtt{stake}}$ can vary in different PoS protocols.

Definition 6 (Coin frozen owner at slot t). *Let* $\mathtt{RCH}_t^{stake}$ *be the corresponding stake reference subchain of the best chain in the view of proposer* P *at time slot* t*. The frozen owner of coin* c *at time slot* t *in the view of proposer* P *is* $\mathtt{Owner}_{\mathtt{RCH}^{stake}}(c)$*.*

In PoS protocols, the concept of frozen coin owner is used to assess the eligibility of proposers to propose blocks. In other words, the probability of an account owner being selected as the block proposer is proportional to the account's frozen balance, where the frozen balance is calculated based on the number of coins belonging to that account with respect to the stake reference subchain.

Definition 7 (Coin-frozen owner mapping). *Let* $\mathtt{RCH}_t^{stake}$ *be the corresponding stake reference subchain of the best available chain in the view of proposer* P *at time slot* t*. The coin-frozen owner mapping in the view of proposer* P *at time slot* t*, which is denoted by* M_t^P*, represents the mapping between the set of coins available in the chain* $\mathtt{RCH}_t^{stake}$ *and their owners with respect to the chain* $\mathtt{RCH}_t^{stake}$*.*

In addition to the stake reference subchain, PoS protocols define another concept as the seed reference subchain. PoS protocols rely on pseudo-random seeds extracted from the ledger to randomly select a set of proposers who are eligible to propose new blocks.

Definition 8 (Seed reference subchain). *The seed reference subchain of chain* $\mathtt{CH}$ *at time slot* t*, which is denoted by* $\mathtt{RCH}_t^{seed}$*, is a subchain of* $\mathtt{CH}$ *resulting from pruning its* k' *rightmost blocks, i.e.,* $\mathtt{RCH}_t^{seed} = \mathtt{Prune}_{k'}\big(\mathtt{CH}\big)$*. Here,* k' *can be calculated using a deterministic function* f_{prune}^{seed} *defined by protocol* Π^{PoS}*.*

The function $f_{\mathtt{prune}}^{\mathtt{seed}}$ takes as input a time slot t and a chain $\mathtt{CH}$ and outputs the number of blocks to be pruned. The seed reference subchain is used to extract a random seed for time slot t.

Definition 9 (Time slot seed). *Let* $\mathtt{CH}^{best}$ *and* $\mathtt{RCH}_t^{seed}$ *be the best chain and its corresponding seed reference subchain in the view of proposer* P *at time slot* t*. The seed of time slot* t *in the view of proposer* P*, which is denoted by* s_t^P*, can be calculated using a deterministic function* f^{random} *defined by protocol* Π^{PoS}*, which takes the chain* $\mathtt{RCH}_t^{seed}$ *as its input and extracts a pseudo-random number as its output.*

For each time slot in PoS protocols, a set of accounts is selected, whose owners are eligible to propose a block at that time slot.

Definition 10 (Eligible account set). *Let s_t^P and M_t^P be the corresponding seed and the coin-frozen owner mapping of time slot t in the view of proposer P. The eligible account set of time slot t in the view of proposer P, denoted by $\mathtt{acctSet}_t^P$, represents the set of accounts whose owners are eligible to propose a block at time slot t with respect to the view of proposer P. The eligible coin set $\mathtt{acctSet}_t^P$ can be obtained using the deterministic function $f^{\mathtt{acctSet}}$ defined by protocol Π^{PoS}, which takes time slot seed s_t^P and the coin-frozen owner mapping M_t^P as its inputs and outputs a randomly selected set of accounts.*

Based on how the eligible account set is defined, PoS protocols can be divided into two groups: the single-proposer model and the multi-proposer model.

Definition 11. *In a general categorization, PoS protocols are divided into one of the following models:*

- ***Single-proposer model**: for each time slot t, the eligible account set exactly contains a single account.*
- ***Multi-proposer Model**: for each time slot t, the eligible account set can be empty or contain one or more accounts.*

One can consider the process of determining the eligible account set as an account selection lottery held at each slot. The randomness of the time slot seed results in a different lottery output for each slot. If in a longest-chain PoS protocol, the time slot seed is extracted from the best available chain rather than the seed reference subchain, a malicious proposer would be able to run as many account selection lotteries as the number of blocks in the block tree at each slot. This would increase the malicious proposer's chance to build a longer chain compared to the honest chain. To understand how the security of longest-chain PoS protocols is affected by the number of pruned blocks of the seed reference subchain, i.e., k', readers can refer to [3]. There exists an attack in the PoS blockchain called the coin-grinding attack that can be applied to the protocols where $\mathtt{RCH}_t^{\mathtt{seed}} \subseteq \mathtt{RCH}_t^{\mathtt{stake}}$. In this attack, since a malicious proposer can get aware of the time slot seed earlier than the coin-frozen owner mapping of that time slot, the malicious proposer can transfer coins between his accounts to increase his winning chance in the account selection lottery. To mitigate the coin-grinding attack, the PoS protocol should be designed in a way that $\mathtt{RCH}_t^{\mathtt{stake}} \subset \mathtt{RCH}_t^{\mathtt{seed}}$.

Each PoS protocol Π^{PoS} can be defined using two deterministic functions: a validating function $\mathtt{Validate}(\cdot)$ and a proposing function $\mathtt{Propose}(\cdot)$. Function $\mathtt{Validate}(\cdot)$ verifies the validity of blocks, and function $\mathtt{Propose}(\cdot)$ is used to propose new blocks and extend the blockchain. In the following, we will define these two functions. Note that our definition of validating function is almost the same as the definition presented in [7] with the difference that we use the concept of frozen coin owner to specify proposers.

Definition 12 (Validating function). *Function $\mathtt{Validate}(\cdot)$ takes a block as input and determines whether the block is valid or not. Note that each block*

$B \in \mathit{CH}$ includes a pointer to its parent $\mathtt{Parent}(B)$, a time slot t_B at which the block is proposed, a reference to an account $\mathtt{acct}_B$, and a signature [3] *σ_B proving the ownership over the account $\mathtt{acct}_B$. Block B is valid at time slot t from the point of view of proposer P if and only if:*

- *$\mathtt{Parent}(B)$ is valid.*
- *Assume $\mathit{CH}^{\lceil B}$ represent a subchain of CH resulting from pruning its rightmost blocks up to including block B. Assume further that $\mathtt{acctSet}_{t_B}$ be the corresponding eligible account set of time slot t_B with respect to the chain $\mathit{CH}^{\lceil B}$* [4]*. Then*

$$\mathtt{acct}_B \in \mathtt{acctSet}_{t_B} \tag{12}$$

and

$$\sigma_B \text{ is a valid signature.} \tag{13}$$

- $t_B \in (t_{\mathtt{Parent}(B)}, t]$

Definition 13 (Proposing function). *The function $\mathtt{Propose}(\cdot)$ takes as input a chain CH, a time slot t, an account $\mathtt{acct}$, and a private key $\mathtt{PvtK}$ corresponding to the account $\mathtt{acct}$. For any account $\mathtt{acct}$ at anytime t, if there exists a valid block B, where $\mathtt{Parent}(B) = \mathtt{Head}(\mathit{CH})$, $t_B = t$, and $\mathtt{acct}_B = \mathtt{acct}$, then $\mathtt{Propose}(\mathit{CH}, t, \mathtt{acct}, \mathtt{PvtK}) = B$; otherwise, $\mathtt{Propose}(\mathit{CH}, t, \mathtt{acct}, \mathtt{PvtK}) = \perp$.*

If in a PoS protocol, the seed and the coin-frozen owner mapping of a time slot are known prior to the start of the time slot, the proposers can specify the eligible account set of that time slot before its start. This phenomenon is called predictability in PoS protocols.

Definition 14 (D-semi predictable). *A PoS protocol is called D-semi predictable if at any given time slot t, any proposer P can predict whether his own accounts belong to $\mathtt{acctSet}^P_{t+D'}$ for all $0 < D' \leq D$.*

Definition 15 (D-full predictable). *A PoS protocol is called D-full predictable, if at any given time slot t, any proposer P can predict $\mathtt{acctSet}^P_{t+D'}$ for all $0 < D' \leq D$.*

To (partially) predict the eligible account set at time slot t, the seed and the coin-frozen owner mapping of time slot t should be known to the proposers. Therefore, having a D-semi/full predictable PoS protocol indicates that $\mathtt{RCH}^{\mathtt{seed}}_{t+D'} \subseteq \mathtt{CH}^{\mathtt{best}}_t$ and $\mathtt{RCH}^{\mathtt{stake}}_{t+D'} \subseteq \mathtt{CH}^{\mathtt{best}}_t$, for all time slot t and for all $0 < D' \leq D$.

[3] The signature should be applied to a piece of information related to the current block.

[4] $\mathtt{CH}^{\lceil B}$ can be considered as the best available chain in the view of $\mathtt{Proposer}(B)$ at time slot t_B.

B.3 Supplementary Definitions

The following definitions are specific to the single-proposer model.

Definition 16 (Proposing sequence). *For two time slots t_1 and t_2, where $t_1 \leq t_2$, the proposing sequence corresponding to the time-slot interval $T = [t_1, t_2]$, which is denoted by $\texttt{Seq}(T)$, is an ordered list of blocks that specifies the proposer of each block proposed during the interval $[t_1, t_2]$. The blocks in $\texttt{Seq}$ are ordered by the time slot at which they are proposed. $B_i^{\mathcal{H}}$ ($B_i^{\mathcal{A}}$) indicates that the i^{th} block in the sequence $\texttt{Seq}$ is an honest (adversarial) block.*

Note that not all the blocks of $\texttt{Seq}$ are guaranteed to be included in the main chain since some of them may get orphaned. In a full-predictable protocol, the attacker can predict the elements of the proposing sequence S in advance.

Definition 17 (Honest (adversarial) proposing subsequence). *Each proposing sequence can be partitioned into two subsequences known as the honest and the adversarial proposing subsequences. The honest (adversarial) proposing subsequence corresponding to the interval $T = [t_1, t_2]$, which is denoted by $\texttt{Seq}^{\mathcal{H}}(T)$ ($\texttt{Seq}^{\mathcal{A}}(T)$), is an ordered list of honest (adversarial) blocks proposed during the interval $[t_1, t_2]$. The length of a proposing subsequence is equal to the number of blocks included in that subsequence.*

For instance, let $\{B_1^{\mathcal{A}}, B_2^{\mathcal{A}}, B_3^{\mathcal{H}}, B_4^{\mathcal{A}}, B_5^{\mathcal{H}}\}$ be the proposing sequence within the time-slot interval $[1, 5]$. In this case, $\{B_3^{\mathcal{H}}, B_5^{\mathcal{H}}\}$ and $\{B_1^{\mathcal{A}}, B_2^{\mathcal{A}}, B_4^{\mathcal{A}}\}$ represent the honest and adversarial proposing subsequences within interval $[1, 5]$, respectively.

Definition 18 (Chain race). *For two slots t_1 and t_2, where $t_1 \leq t_2$, the chain race corresponding to the time-slot interval $[t_1, t_2]$ is the race between the adversarial private chain and the honest public chain that satisfies the following properties:*

- *Before t_1, both the adversarial and honest chains share the same subchain denoted as $\mathcal{C}^{\lceil t_1}$.*
- *Adversarial private chain is made up of the sequence $\mathcal{C}^{\lceil t_1} || \{B_i^{\mathcal{A}}, \cdots, B_j^{\mathcal{A}}\}$, where $\{B_i^{\mathcal{A}}, \cdots, B_j^{\mathcal{A}}\}$ is the adversarial proposing subsequence corresponding to the interval $[t_1, t_2]$.*
- *The honest public chain is made up of the sequence $\mathcal{C}^{\lceil B_i^{\mathcal{A}}} || \{B_{i'}^{H}, \cdots, B_{j'}^{H}\}$, where $\{B_{i'}^{H}, \cdots, B_{j'}^{H}\}$ is the honest proposing subsequence corresponding to the interval $[t_1, t_2]$.*

If the length of the adversarial proposing subsequence within the interval $[t_1, t_2]$ is greater than the length of its corresponding honest proposing subsequence, the attacker can **win** the chain race corresponding to the interval $[t_1, t_2]$. Winning the chain race within the interval $[t_1, t_2]$ indicates that if the attacker forks the main chain at time slot t_1 (creates a fork on top of the last block proposed before t_1), the adversarial chain within interval $[t_1, t_2]$ can orphan the honest proposers' consecutive blocks proposed within the same interval.

Definition 19 (Longest dominant chain). *The longest dominant chain starting at a time slot t_1, which is denoted by $\texttt{LDC}(t_1)$, is the chain made up of the adversarial proposing subsequence corresponding to the interval $[t_1, t_2]$, where the interval $[t_1, t_2]$ satisfies the following properties:*

- $t_2 \geq t_1$.
- *The attacker wins the chain race corresponding to the interval $[t_1, t_2]$.*
- *There is no time slot such as t_3, where $t_3 > t_2$ and the attacker wins the chain race corresponding to the interval $[t_1, t_3]$.*

If there is no such a chain starting at time slot t_1, $\texttt{LDC}(t_1)$ is called to be empty and is denoted by $\texttt{LDC}(t_1) = \emptyset$.

We denote by $L^{\texttt{LDC}}(t)$ the length of the longest dominant chain starting at t. If $\texttt{LDC}(t)$ is empty, then $L^{\texttt{LDC}}(t) = 0$. Note that if the attacker is the proposer of a block $B_t^{\mathcal{A}}$ at time slot t (slot t is adversarial), then $\texttt{LDC}(t)$ cannot be empty, and $\min(L^{\texttt{LDC}}(t)) = 1$. The minimum length occurs when $\texttt{LDC}(t)$ comprises only $B_t^{\mathcal{A}}$.

C On profitability of the selfish proposing attack

In this section, we define the terms "block ratio", "relative revenue", and "time-averaged profit" and discuss the effect of selfish proposing on these terms in LC-PoS protocols. We first define the honest strategy in LC-PoS protocols.

Definition 20 (Honest strategy). *A proposer always chooses to propose his new block on top of the longest chain available in his view. If the proposer manages to propose a new block, he immediately publishes the block to all the other proposers.*

We assume that block proposers are divided into two groups: (1) honest proposers, denoted by $\mathcal{H}$, who follow the honest strategy, and (2) the adversarial proposers who are under the control of an attacker, denoted by $\mathcal{A}$, and may deviate from the honest strategy to gain a higher profit. The blocks that are proposed by honest proposers (the attacker) are called honest (adversarial) blocks.

Definition 21 (Communication capability). *We denote by η the communication capability of attacker $\mathcal{A}$. This means, in the case of a block race, where two blocks are published simultaneously by attacker $\mathcal{A}$ and an honest proposer, the fraction of total honest proposers that receive the block proposed by the attacker first is equal to η. The honest proposers who receive the block proposed by the attacker first propose their new block on top of the attacker's block.*

C.1 Block ratio

The term "bock ratio" is defined as follows:

Definition 22 (Block ratio). *The block ratio of attacker $\mathcal{A}$ who follows strategy π is defined as follows:*

$$\mathtt{BlkRatio}_{\mathcal{A}}(\pi) = \lim_{t\to\infty} \frac{NB_t^{\mathcal{A}}}{NB_t^{\mathcal{A}} + NB_t^{\mathcal{H}}} = \lim_{t\to\infty} \frac{NB_t^{\mathcal{A}}}{NB_t^{\mathcal{T}}} \ , \tag{14}$$

where $NB_t^{\mathcal{A}}$, $NB_t^{\mathcal{H}}$, and $NB_t^{\mathcal{T}}$ denote the number of adversarial blocks, the number of honest blocks, and the total number of blocks in the main chain up to including time slot t, respectively.

It is obvious that selfish proposing can result in a change in the attacker's block ratio since some of the honest and adversarial blocks may get orphaned, i.e., remain out of the main chain. However, selfish proposing cannot always result in an increase in the attacker's block ratio, especially for the attackers who own a small amount of stake share. Depending on the underlying LC-PoS protocol and its detailed specifications of the longest-chain rule, the attacker needs to own a minimum threshold of stake share to make his block ratio under selfish proposing surpass that of honest proposing. To this end, we first discuss the specifications of the longest-chain rule and then define the profitable threshold.

In the longest-chain protocols, proposers always propose a new block on top of the longest chain. However, in the case when there are multiple chains with the same height, different longest-chain protocols may apply different choice rules to the same-height forks. For instance, the winning chain can be selected randomly, be the chain with a lower VRF output, or be the chain that the proposer has received first. In this paper, we assume when there exist multiple chains with the same height, the proposers choose the chain that is received in their local view earlier than the other chains.

Definition 23 (Profitable stake (mining) share threshold). *For each protocol Π, the minimum amount of stake (mining) share for an attacker with communication capability η that satisfies*

$$\mathtt{BlkRatio}_{\mathcal{A}}(\pi^{selfish}) \geq \mathtt{BlkRatio}_{\mathcal{A}}(\pi^{honest})$$

is called the profitable stake (mining) share threshold and is denoted by $\alpha^{\Pi}(\eta)$.

In this paper, we calculate the profitable stake share threshold for different LC-PoS protocols.

C.2 Relative revenue

In PoS protocols, proposers who actively participate in blockchain extension are incentivized with a reward. This reward, which is in the form of coins, can be derived from multiple sources; however, we will focus only on two of these sources: (1) the transaction fee reward, which is a number of coins paid by the transaction owner to proposers for the inclusion of his transaction in a block, and (2) the incentivizing reward, which is a number of coins paid by the protocol to proposers for generating new blocks.

Definition 24 (Relative revenue). *The relative revenue of attacker $\mathcal{A}$ who follows strategy π is defined as follows:*

$$\texttt{RelRevenue}_{\mathcal{A}}(\pi) = \lim_{T\to\infty} \frac{\sum_{t=1}^{T} NC_t^{\mathcal{A}}}{\sum_{t=1}^{T} \left(NC_t^{\mathcal{A}} + NC_t^{\mathcal{H}}\right)} \ , \tag{15}$$

where $NC_t^{\mathcal{A}}$ and $NC_t^{\mathcal{H}}$ respectively denote the number of reward coins[5] *gained by the attacker and the honest proposers at time slot t*[6]*.*

Knowing that a proposer who owns the profitable stake share threshold can increase his block ratio by applying the selfish proposing attack, the first question that needs to be addressed is whether selfish proposing can also lead to an increase in the attacker's relative revenue or not. The answer depends on the reward mechanism of the underlying LC-PoS protocol. For instance, in the first version of Ouroboros [17], i.e., the underlying protocol of Cardano, the authors have claimed that the introduced protocol is resistant to the selfish proposing attack as its reward mechanism is designed in a way that the attacker has no incentive to deviate from the protocol. In the first version of Ouroboros, the epoch reward is distributed among the proposers proportional to their stake shares rather than their block ratio in the epoch. In other words, the reward mechanism is not sensitive to whether a slot leader (proposer) has issued a block or not in its assigned time slot. Therefore, although selfish proposing can result in a change in the attacker's block ratio, it cannot affect the reward ratio, i.e., relative revenue, the attacker receives. However, this kind of reward mechanism in which the reward is distributed according to the proposers' stake share rather than their effort on extending the chain can lead to an unfair reward distribution. An offline proposer can collect a higher amount of reward compared to an active proposer just because of owning a greater stake share. To bring fairness back to the implemented version of Cardano, the epoch reward assigned to a stake pool (proposer), which is calculated proportional to its stake share, is adjusted by the pool performance. The pool performance is defined as the block ratio divided by the stake ratio. Therefore, in the implemented version of Cardano, the relative revenue of a proposer is positively correlated with his block ratio, and selfish proposing can still threaten blockchain progress. As can be seen in the longest-chain blockchain protocols, such as Bitcoin, Cardano, etc., it seems that to fairly incentivize the proposers to participate in the blockchain extension, the reward mechanism should be designed in a way that the relative revenue of a proposer is positively correlated with his block ratio. And whenever there is a positive correlation between the relative revenue and the block ratio, the selfish proposing attack can result in an increase in the attacker's relative revenue.

[5] The coins a block proposer receives include the transaction fees as well as possibly a reward for his contribution in extending the blockchain.

[6] Note that based on the reward mechanism used in a protocol, a block proposer at slot t may receive coin(s) exactly in the same slot t or in a slot later than t. In this definition, we consider both of these scenarios as the proposer's coin(s) in slot t.

Long-range and short-range selfish proposing In PoS protocols, since the essence of both block reward and stake is the same thing, namely coin, the reward that a proposer collects can get added to his stake. When all the proposers follow the honest strategy, the relative revenue that a proposer collects is equal to his stake ratio. Therefore, if all the proposers add the collected reward to their stake, the proposers' stake ratios remain unchanged. However, if a malicious proposer starts the selfish proposing attack, his relative revenue becomes greater than his stake ratio. Conversely, honest proposers' relative revenue falls below their stake share. Therefore, after adding the collected reward to their stake, the selfish proposer's stake ratio will increase, while the stake ratio of others will decrease. This implies that by applying a long-range selfish proposing attack in LC-PoS protocols, the stake share of the selfish proposer will gradually grow and eventually exceed more than one-half. Therefore, we can conclude that in an LC-PoS protocol $\Pi^{\texttt{LC-PoS}}$, a selfish proposer $\mathcal{A}$ who owns stake share $\alpha_{\mathcal{A}} < 51\%$ and communication capability η, can perform 51% attack at a point in the future provided that: i) proposers are rewarded proportional to the number of blocks they have contributed to the main chain, ii) protocol $\Pi^{\texttt{LC-PoS}}$ runs infinitely, and iii) $\alpha_{\mathcal{A}} > \alpha_{\eta}^{\Pi^{\texttt{LC-PoS}}}$.

To assess the selfish proposer's relative revenue in a short period of time, we define the term "short-range relative revenue".

Definition 25 (Short-range relative revenue). *Let $B_1^{\texttt{Ref}}$ and $B_2^{\texttt{Ref}}$ be two consecutive stake reference blocks in the main chain. Assume attacker $\mathcal{A}$ follows strategy π during the interval $T = \big(t(B_1^{\texttt{Ref}}), t(B_2^{\texttt{Ref}})\big]$. The short-term relative revenue of attacker $\mathcal{A}$ in the interval T is defined as follows:*

$$\texttt{RelRevenue}_{\mathcal{A}}\big(\pi, T\big) = \frac{\sum_{t \in T} NC_t^{\mathcal{A}}}{\sum_{t \in T} \Big(NC_t^{\mathcal{A}} + NC_t^{\mathcal{H}}\Big)} \ . \tag{16}$$

In this paper, we will show that with the same amount of (stake or mining) share and communication capability, selfish proposing in LC-PoS protocols can result in a higher short-term relative revenue compared to selfish mining in PoW protocols. If assuming all the blocks receive the same amount of transaction fees and the incentivizing reward is distributed exactly proportional to the proposers' block ratio, a proposer's relative revenue will be equal to his block ratio.

C.3 Time-averaged profit

We can assess the profitability of selfish proposing using both concepts of block ratio and time-averaged profit. We first define the time-averaged profit. Note that in the following definition, we ignore the cost of block production as it is negligible in PoS protocols compared to PoW.

Definition 26 (Time-averaged profit). *The time-averaged profit (profit per time slot) of attacker $\mathcal{A}$ who follows strategy π is defined as follows:*

$$\texttt{Profit}_{\mathcal{A}}(\pi) = \lim_{T \to \infty} \frac{\sum_{t=1}^{T} NC_t^{\mathcal{A}}}{T} \ , \tag{17}$$

where $NC_t^{\mathcal{S}}$ is the number of coins the attacker received at time slot t.

Most of the papers related to PoW protocols have used the concept of relative revenue to analyze selfish mining profitability. However, the authors in papers such as [16] have argued that time-averaged profit is a better benchmark compared to the relative revenue to assess profitability. Here, we first briefly explain why the relative revenue does not reflect the whole story behind the profitability of selfish proposing in LC-PoS protocols.

Let $B_1^{\texttt{Ref}}$ and $B_2^{\texttt{Ref}}$ be two consecutive stake reference blocks in the main chain and the time slot interval T is defined as $T = \big(t(B_1^{\texttt{Ref}}), t(B_2^{\texttt{Ref}})\big]$. Assume attacker $\mathcal{A}$ has followed the honest strategy and owned stake share $\alpha > \alpha_\eta^{\Pi^{\text{LC-PoS}}}$ before and including the reference block $B_1^{\texttt{Ref}}$ and decides to follow selfish strategy $\pi^{\texttt{selfish}}$ in the interval T. Due to the selfish proposing attack, there is an increase in the expected value of relative revenue during the interval T, i.e.,

$$\mathbf{E}\Big[\texttt{RelRevenue}_{\mathcal{A}}^{\texttt{short}}\big(\pi^{\texttt{selfish}}, T\big)\Big] > \alpha \ . \tag{18}$$

However, since the amount of $\mathcal{A}$'s stake share is the same before and during the interval T, the attacker's chance of winning the proposer selection lottery is the same before and after the start of the attack. This implies that the expected number of blocks added to the main chain by attacker $\mathcal{A}$ in the interval T is equal to or even less[7] than the expected number of blocks attacker $\mathcal{A}$ contributed to the main chain during the same period of time slots before starting the attack. Therefore, an increase in the relative revenue after starting the selfish proposing attack does not necessarily reflects an increase in the selfish proposer's time-averaged profit. Using time-averaged profit, we can assess the amount of reward a selfish proposer receives regardless of the honest proposers' collected reward.

Difference in time-averaged profit between LC-PoS and PoW: In PoW protocols, there exists a difficulty adjustment mechanism (DAM) that controls transaction throughput by modifying the mining difficulty (mining target). It is shown that before a DAM, the time-averaged profit of selfish mining cannot surpass the time-averaged profit of honest mining in PoW protocols. However, once the difficulty level of mining gets adjusted, the mining puzzle gets easier and the selfish miner can mine a new block in a shorter period of time. Therefore, by applying DAM, the selfish miner's time-averaged profit increases. This shows that a selfish miner in PoW protocols should maintain the attack for a couple of difficulty periods to increase his time-averaged profit and compensate for his loss at the start of the attack.

In LC-PoS protocols, the stake share of a selfish proposer will gradually grow during the selfish proposing attack, which can guarantee an increase in the long-term time-averaged profit of the attacker. However, to assess the short-term time-averaged profit of selfish proposing in LC-PoS protocols, we should take into account the rewarding mechanism used in the LC-PoS protocol.

[7] Note that some of the adversarial blocks may get orphaned after starting the selfish proposing attack.

Transaction reward per slot: If assuming that the total amount of transaction fee a block can receive is the same before and after the attack, the attacker's transaction reward per slot will not increase after the attack because the attacker's stake share, and consequently, his block generation rate have not changed. However, in a real-world scenario, the attacker can utilize selfish proposing to increase his transaction reward per slot. As a result of a selfish proposing attack, the block generation rate decreases, resulting in a reduction in transaction throughput. In such a competitive situation, a transaction owner who wants his transaction to get included in a block should increase the transaction fee. Therefore, there will be a rise in the amount of transaction fees, and accordingly, the attacker's transaction reward per slot will increase. Moreover, when a selfish proposer manages to orphan an honest block, the selfish proposer can steal the valuable or MEV transactions included in the honest block and put them in his own block to further enhance his time-averaged profit.

Incentivizing reward per slot: We consider two scenarios for the distribution of incentivizing rewards. In the first scenario, a block proposer who generates a new block receives an incentivizing reward through a transaction included in the block. In this case, the selfish proposing attack cannot increase the attacker's incentivizing reward per slot. In the second scenario, at the end of each `epoch`, which is a predetermined set of time slots, the incentivizing reward is distributed among proposers proportional to the number of blocks they have contributed to the main chain during the `epoch`. In this scenario, if the epoch incentivizing reward is fixed, a selfish proposer can increase his incentivizing reward per slot immediately after the start of the attack.

Trade off between transaction throughput and selfish proposer's time-averaged profit: Once an attacker starts the selfish proposing attack in an LC-PoS blockchain, due to orphan occurrence, the average number of blocks per slot will decrease, which results in a decrease in the transaction throughput. As already mentioned, in PoW protocols, there is a difficulty adjustment mechanism that aims to balance the transaction throughput by modifying the mining difficulty. Although in PoS protocols, there is no DAM, there exist some predetermined parameters such as slot duration and the average number of blocks per slot that affect the transaction throughput. In most of the PoS protocols, these parameters are determined prior to the start of the protocol; however, blockchain proposers or designers can decide to modify these parameters to balance the transaction throughput in the occurrence of selfish proposing. The transaction throughput can get balanced at the cost of an increase in the average number of blocks per slot, which results in an increase in the selfish proposer's time-averaged profit.

Table 2: States and their corresponding sequences

State	Proposing sequence
$S_{0,0}$	-
$S_{1,0}$	$\{B_1^{\mathcal{A}}\}$
$S_{1,1}$	$\{B_1^{\mathcal{A}}, B_2^{\mathcal{H}}\}$
$S_{2,0}$	$\{B_1^{\mathcal{A}}, B_2^{\mathcal{A}}\}$
$S_{1,2}$	$\{B_1^{\mathcal{A}}, B_2^{\mathcal{H}}, B_3^{\mathcal{H}}\}$

D Selfish proposing attack in LC-PoS protocols with perfect randomness

D.1 Strategy $\pi^{\text{S-PR}}$

Strategy $\pi^{\text{S-PR}}$ is suitable for LC-PoS protocols with perfect randomness. We assume that the LC-PoS environment is compatible with the single-proposer model. Let α and η denote the attacker's stake share and communication capability, respectively. Let B_t^P denote the block proposed by proposer $P \in \{\mathcal{A}, \mathcal{H}\}$ at time slot t. For the sake of simplicity, the time slot at which the latest common block between the honest and adversarial chains is proposed is normalized to be 0. This normalization implies that a fork always starts at time slot $t = 1$. In our model, each state corresponds to a specific proposing sequence, where the concept of proposing sequence is introduced in Definition 16. By following the strategy $\pi^{\text{S-PR}}$, the system transitions among five main states, each of them represented by $S_{l^{\mathcal{A}}, l^{\mathcal{H}}}$, where $l_{\mathcal{A}}$ ($l_{\mathcal{H}}$) denotes the length of the adversarial (honest) fork. The corresponding proposing sequences of these 5 states are represented in Table 2. The selfish proposing strategy $\pi^{\text{S-PR}}$ is presented in Table 3. $\texttt{action}_0$, $\texttt{action}_1$, and $\texttt{action}_2$, which are mentioned in Table 3, are defined as follows:

$\texttt{action}_0$: The action corresponds to state $S_{0,0}$, where $l_{\mathcal{H}} = l_{\mathcal{A}} = 0$. The action represents that the attacker waits until a new block is proposed. If the new block is an honest block, the attacker adopts the honest chain and remains

Table 3: Strategy $\pi^{\text{S-PR}}$

current state	action	next state	probability	reward
$S_{0,0}$	$\texttt{action}_0$	$S_{0,0}$	$1 - \alpha$	$R_{\mathcal{A}} = 0,\ R_{\mathcal{H}} = 1$
		$S_{1,0}$	α	$R_{\mathcal{A}} = 0,\ R_{\mathcal{H}} = 0$
$S_{1,0}$	`wait`	$S_{1,1}$	$1 - \alpha$	$R_{\mathcal{A}} = 0,\ R_{\mathcal{H}} = 0$
		$S_{2,0}$	α	$R_{\mathcal{A}} = 0,\ R_{\mathcal{H}} = 0$
$S_{1,1}$	`match`	$S_{1,2}$	$(1-\eta)(1-\alpha)$	$R_{\mathcal{A}} = 0,\ R_{\mathcal{H}} = 0$
		$S_{0,0}$	$\eta(1-\alpha)$	$R_{\mathcal{A}} = 1,\ R_{\mathcal{H}} = 1$
		$S_{0,0}$	α	$R_{\mathcal{A}} = 2,\ R_{\mathcal{H}} = 0$
$S_{2,0}$	$\texttt{action}_1$	$S_{0,0}$	1	$R_{\mathcal{A}} = R_1,\ R_{\mathcal{H}} = 0$
$S_{1,2}$	$\texttt{action}_2$	$S_{1,1}$	$1 - \alpha$	$R_{\mathcal{A}} = 0,\ R_{\mathcal{H}} = R_2$
		$S_{0,0}$	α	$R_{\mathcal{A}} = R_3,\ R_{\mathcal{H}} = R_4$

in the same state. If the new block is an adversarial block, the attacker moves to state $S_{1,0}$.

$\texttt{action}_1$: The action corresponds to state $S_{2,0}$, where $l_{\mathcal{H}} = l_{\mathcal{A}} - 2$. The action represents that the attacker waits until the point where his lead reduces to 1, i.e., $l_{\mathcal{H}} = l_{\mathcal{A}} - 1$, and then publishes the whole adversarial fork to override the honest fork.

$\texttt{action}_2$: The action corresponds to state $S_{1,2}$, the sequence of which is $\{B_1^{\mathcal{A}}, B_2^{\mathcal{H}}, B_3^{\mathcal{H}}\}$. Let L denote the number of consecutive honest blocks proposed after block $B_3^{\mathcal{H}}$ and before the next adversarial block, where $L \geq 0$. This implies that the first adversarial block after visiting state $S_{1,2}$ is proposed at time slot $t = 4 + L$ and can be represented by $B_{4+L}^{\mathcal{A}}$.

1. In the case that the block proposed at time slot $t = 5 + L$ is an honest block denoted by $B_{5+L}^{\mathcal{H}}$, the attacker gives up on block $B_1^{\mathcal{A}}$ and generates a new fork (jump) on top of honest block $B_{3+L}^{\mathcal{H}}$. In this case, the new state is $S_{1,1}$, where the honest fork only contains block $B_{5+L}^{\mathcal{H}}$, and the adversarial fork only contains block $B_{4+L}^{\mathcal{A}}$.
2. In the case that the block proposed at time slot $t = 5 + L$ is an adversarial block denoted by $B_{5+L}^{\mathcal{A}}$, the attacker waits until one of the following cases happens:
   (a) If at a future time slot, the condition $l_{\mathcal{A}} = l_{\mathcal{H}} + 1$ holds, the attacker immediately publishes the whole adversarial fork to override the honest fork. In this case, the new state is $S_{0,0}$.
   (b) If at a future time slot, the condition $l_{\mathcal{A}} = l_{\mathcal{H}} - L$ holds, the attacker gives up on block $B_1^{\mathcal{A}}$ and generates a new fork (jump) on top of honest block $B_{3+L}^{\mathcal{H}}$, where the new adversarial (honest) fork contains all the adversarial (honest) blocks proposed after time slot $t = 3 + L$. By doing so, the new adversarial fork has a one-block lead over the new honest fork. Then, the attacker immediately publishes the new adversarial fork to override the new honest fork. In this case, the final state is $S_{0,0}$.

Note that states $S_{2,0}$ and $S_{1,2}$ contain transitions to hidden states that are not included in the five main states. The set of states and possible transitions are depicted in Figure 5.

D.2 Block ratio of strategy $\pi^{\textbf{S-PR}}$

To obtain the block ratio achieved by following strategy $\pi^{\text{S-PR}}$, rewards R_1, R_2, R_3, and R_4, which are mentioned in Table 3, need to b calculated. To calculate the rewards, we use random walking on a (x, y)-grid. Throughout this paper, a two-dimensional (x, y)-grid is used to represent the chain race between the honest and adversarial forks. Each proposing sequence can be represented by a path on this grid, which is referred to as the "block path". Whenever the time slot is honest, i.e., the block proposer of the time slot is honest, the block path moves one step up. Whenever the time slot is adversarial, i.e., the block proposer of the time slot is the attacker, the block path moves one step to the right.

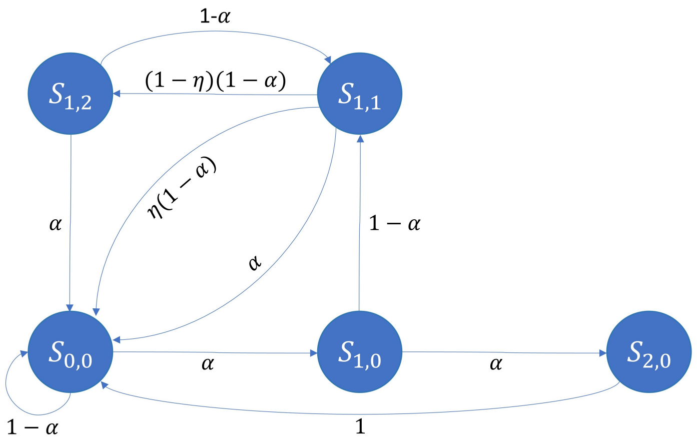


Fig. 5: State representation of strategy $\pi^{\text{S-PR}}$

According to Lemma D.2. presented in [13], the adversarial reward of following $\texttt{action}_1$ at state $S_{2,0}$, i.e., R_1, can be obtained as follows:

$$R_1 = 2 + \frac{\alpha}{1-2\alpha} \quad . \tag{19}$$

Lemma 1. *The average honest reward resulting from the transition from state $S_{1,2}$ to state $S_{1,1}$ under $\texttt{action}_2$ (Scenario 1 occurrence) can be obtained as follows:*

$$R_{\mathcal{H}} = R_2 = \frac{\sum_{i=0}^{\infty}(2+i)\alpha(1-\alpha)^{i+1}}{\texttt{Pr}(\textit{Scenario 1})} = 2 + \frac{1-\alpha}{\alpha} \quad . \tag{20}$$

Proof. We use a two-dimensional (x, y)-grid to represent the chain race as depicted in Figure 6. Being at state $S_{1,2}$ implies that the block path started at point $(0, 0)$ has reached the point $(1, 2)$. Scenario 1 is equivalent to the event that once the block path starting at point $(1, 2)$ reaches the line $x = 2$ at one of the points $(2, 2+i)$ for $i \geq 0$, it immediately moves one step upward to reach the point $(2, 3+i)$. The probability that the block path starting at $(1, 2)$ reaches the line $x = 2$ for the first time at point $(2, 2+i)$ and then moves to the point $(2, 3+i)$ is equal to $\alpha(1-\alpha)^{i+1}$. Therefore, the probability of Scenario 1 occurrence can be obtained as follows:

$$\texttt{Pr}(\text{Scenario 1}) = \sum_{i=0}^{\infty}\alpha(1-\alpha)^{i+1} = 1-\alpha \quad . \tag{21}$$

The adversarial reward of Scenario 1 is equal to 0. If assuming L denotes the number of consecutive honest blocks proposed after being at state $S_{1,2}$ and before the next adversarial block is proposed, the honest reward resulting from Scenario 1 is equal to $2 + L$. Therefore, the average honest reward under Sce-

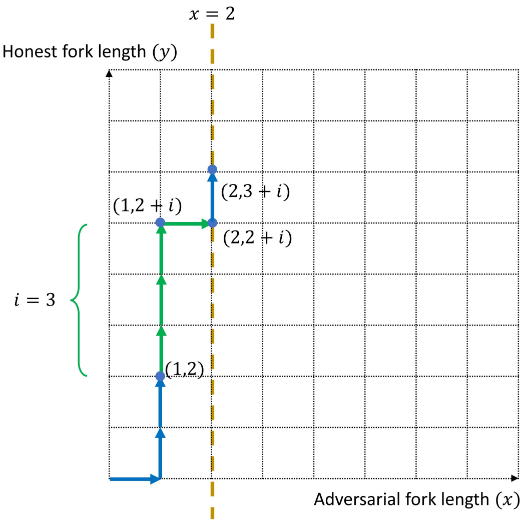


Fig. 6: Block path in Scenario 1

nario 1 can be obtained as follows:

$$R_2 = \frac{\sum_{i=0}^{\infty} (2+i)\alpha(1-\alpha)^{i+1}}{\texttt{Pr}(\text{Scenario 1})} = 2 + \frac{1-\alpha}{\alpha} \quad . \tag{22}$$

□

To find R_3 and R_4, we first present a couple of lemmas.

Lemma 2. *Let P_r denote the probability of the event that the block path starting at point $(0,0)$ reaches the line $y = x - r$ for $r \geq 1$ at least once. We have $P_r = \left(\frac{\alpha}{1-\alpha}\right)^r$.*

Proof. We prove the lemma using induction. First, we find the probability that the block path reaches the line $y = x - 1$ at least once, i.e., P_1. Assume the block path reaches the line $y = x - 1$ for the first time at point $(s+1, s)$. To achieve this, the block path should reach point (s, s) without passing below the line $y = x$ and then move one step to the right. The number of paths from point $(0,0)$ to point (s,s) without passing below the line $y = x$ is equal to the s^{th} Catalan number denoted by c_s. Therefore, the probability of reaching the line $y = x - 1$ for the first time at point $(s+1, s)$ is equal to $c_s \alpha^{s+1}(1-\alpha)^s$. Thus, the probability of reaching the line $y = x - 1$ can be calculated as follows:

$$P_1 = \alpha \sum_{s=0}^{\infty} c_s \big(\alpha(1-\alpha)\big)^s = \frac{\alpha}{1-\alpha} \quad . \tag{23}$$

The formula for solving the series above is presented in [8]. Next, we find the probability that the block path reaches the line $y = x - 2$ at least once, i.e.,

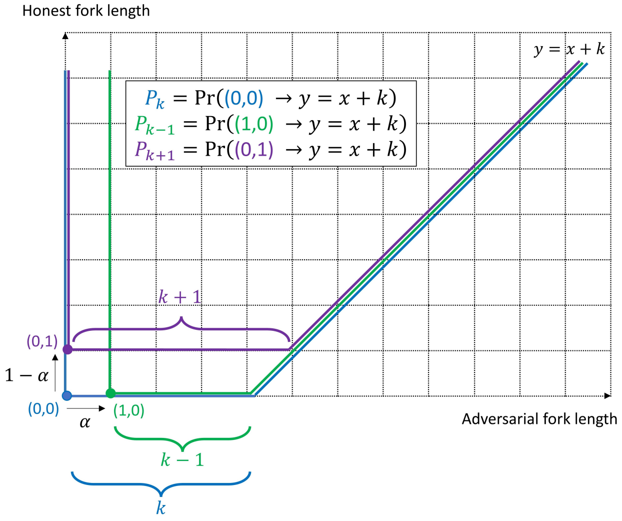

Honest fork length
y = x + k
P_k = Pr((0,0) → y = x + k)
P_{k−1} = Pr((1,0) → y = x + k)
P_{k+1} = Pr((0,1) → y = x + k)
k + 1
(0,1)
1 − α
(0,0)
α
(1,0)
Adversarial fork length
k − 1
k

Fig. 7: Representation of probability P_k

P_2. Assume the block path reaches the line $y = x - 2$ for the first time at point $(s+2, s)$. To achieve this, the block path should reach point $(s+1, s)$ without passing below the line $y = x - 1$ and then move one step to the right. The number of paths from point $(0, 0)$ to point $(s+1, s)$ without passing below the line $y = x - 1$ is equal to c_{s+1}. Therefore, the probability of reaching the line $y = x-2$ for the first time at point $(s+2, s)$ is equal to $c_{s+1}\alpha^{s+2}(1-\alpha)^s$. Thus, the probability of reaching the line $y = x - 2$ can be calculated as follows:

$$P_2 = \alpha^2 \sum_{s=0}^{\infty} c_{s+1} \big(\alpha(1-\alpha)\big)^s = \frac{\alpha}{1-\alpha} \sum_{s=1}^{\infty} c_s \big(\alpha(1-\alpha)\big)^s = \left(\frac{\alpha}{1-\alpha}\right)^2 . \tag{24}$$

As the next step, assume $P_{k-1} = \left(\frac{\alpha}{1-\alpha}\right)^{k-1}$ and $P_k = \left(\frac{\alpha}{1-\alpha}\right)^k$ for $k \geq 2$. We need to prove $P_{k+1} = \left(\frac{\alpha}{1-\alpha}\right)^{k+1}$. Considering the grid depicted in Figure 7, we obtain:

$$P_k = \alpha P_{k-1} + (1-\alpha)P_{k+1} \implies P_{k+1} = \frac{P_k - \alpha P_{k-1}}{1-\alpha} = \left(\frac{\alpha}{1-\alpha}\right)^{k+1} . \tag{25}$$

□

Lemma 3. *Let $\overline{P_r}$ denote the probability of the event that the block path starting at $(0, 0)$ never reaches the line $y = x - r$ for $r \geq 1$. We have $\overline{P_r} = 1 - \left(\frac{\alpha}{1-\alpha}\right)^r$.*

Proof. The probability of the complementary event of never reaching the line $y = x - r$, i.e., P_r, is calculated in Lemma 2. Therefore, we have $\overline{P_r} = 1 - P_r$. □

Lemma 4. *Let G^r_s for $r \geq 1$ denote the number of paths in a (x,y)-grid from start point $(0,0)$ to the point (s,s) without reaching the lines $y = x+1$ and $y = x-r$ (define G^r_0 to be equal to 1). We have:*

$$\begin{aligned}
&\sum_{s=0}^{\infty} G^r_s\big(\alpha(1-\alpha)\big)^s = \Big(\frac{1}{1-\alpha}\Big)\frac{1-(\frac{\alpha}{1-\alpha})^r}{1-(\frac{\alpha}{1-\alpha})^{r+1}} \ , \\
&\sum_{s=0}^{\infty} sG^r_s\big(\alpha(1-\alpha)\big)^s = \frac{\alpha\big(1-(\frac{\alpha}{1-\alpha})^r\big) - r(\frac{\alpha}{1-\alpha})^r}{(1-\alpha)(1-2\alpha)\big(1-(\frac{\alpha}{1-\alpha})^{r+1}\big)} \\
&+ \frac{(r+1)\alpha\big(1-(\frac{\alpha}{1-\alpha})^r\big)(\frac{\alpha}{1-\alpha})^r}{(1-\alpha)^2(1-2\alpha)\big(1-(\frac{\alpha}{1-\alpha})^{r+1}\big)^2} \ .
\end{aligned} \tag{26}$$

Proof. Assume $\texttt{Event}_1(s)$ is defined as follows: (i) the block path starting at $(0,0)$ reaches the point (s,s), and (ii) before reaching the point (s,s), the block path never passes the line $y = x$ and never reaches the line $y = x-r$. According to the definition of G^r_s, the probability of $\texttt{Event}_1(s)$ is equal to $G^r_s\alpha^s(1-\alpha)^s$. Assume $\texttt{Event}_2(s)$ is defined as follows: (i) the block path starting at $(0,0)$ passes the line $y = x$ for the first time at point (s,s), and (ii) before reaching the point (s,s), the block path never reaches the line $y = x-r$. $\texttt{Event}_2(s)$ happens if, after the occurrence of $\texttt{Event}_1(s)$ and reaching the point (s,s), the block path immediately moves one step up to pass the line $y = x$ and reach the point $(s, s+1)$. Therefore, the probability of $\texttt{Event}_2(s)$ is equal to $G^r_s\alpha^s(1-\alpha)^{s+1}$. Assume $\texttt{Event}_3(s)$ is defined as follows: (i) the block path starting at $(0,0)$ passes the line $y = x$ for the first time at point (s,s), and (ii) the block path never reaches the line $y = x-r$ both before and after reaching the point (s,s). $\texttt{Event}_3(s)$ happens if, after the occurrence of $\texttt{Event}_2(s)$ and reaching the point $(s, s+1)$, the block path never reaches the line $y = x-r$. The event that the block path starting at $(s, s+1)$ never reaches the line $y = x-r$ is equivalent to the event that the block path starting at $(0,0)$ never reaches the line $y = x-(r+1)$, the probability of which is equal to $1-(\frac{\alpha}{1-\alpha})^{r+1}$ according to Lemma 3. Therefore, the probability of $\texttt{Event}_3(s)$ is equal to $G^r_s\alpha^s(1-\alpha)^{s+1}\big(1-(\frac{\alpha}{1-\alpha})^{r+1}\big)$. Note that since $1-\alpha > \alpha$, a block path will eventually pass the line $y = x$. Therefore, the sum of $\texttt{Event}_3(s)$ probabilities over all values of s is equal to the probability that the block path never reaches the line $y = x-r$, the probability of which is equal to $1-(\frac{\alpha}{1-\alpha})^r$ according to Lemma 3. As a result,

$$\begin{aligned}
&\sum_{s=0}^{\infty} G^r_s\alpha^s(1-\alpha)^{s+1}\big(1-(\frac{\alpha}{1-\alpha})^{r+1}\big) = 1-(\frac{\alpha}{1-\alpha})^r \\
&\Rightarrow \sum_{s=0}^{\infty} G^r_s\big(\alpha(1-\alpha)\big)^s = \Big(\frac{1}{1-\alpha}\Big)\frac{1-(\frac{\alpha}{1-\alpha})^r}{1-(\frac{\alpha}{1-\alpha})^{r+1}} \ .
\end{aligned} \tag{27}$$

To prove the second equality in Lemma 4, we use the variable substitution $\alpha(1-\alpha) = x$ in the equality above. We have:

$$\sum_{s=0}^{\infty} G_s^r x^s = \Big(\frac{2}{1+\sqrt{1-4x}}\Big)\frac{1-(\frac{1-\sqrt{1-4x}}{1+\sqrt{1-4x}})^r}{1-(\frac{1-\sqrt{1-4x}}{1+\sqrt{1-4x}})^{r+1}} \quad . \tag{28}$$

By taking the derivative from both sides, then multiplying both sides to x, and finally substituting $x = \alpha(1-\alpha)$, the second equality in Lemma 13 can be obtained. □

Lemma 5. *The probability that the block path starting at* $(0,0)$ *reaches the line* $y = x+1$ *before reaching the line* $y = x-r$ *for* $r \geq 1$ *is equal to* $\frac{1-(\frac{\alpha}{1-\alpha})^r}{1-(\frac{\alpha}{1-\alpha})^{r+1}}$.

Proof. The event that the block path reaches the line $y = x+1$ for the first time at point $(s, s+1)$ without having reached the line $y = x-r$ is the same as $\texttt{Event}_2(s)$ introduced in proof of Lemma 4, the probability of which is equal to $G_s^r \alpha^r (1-\alpha)^{r+1}$. The probability that the block path starting at $(0,0)$ reaches the line $y = x+1$ before reaching the line $y = x-r$ is equal to the sum of the $\texttt{Event}_2(s)$ probabilities over all values of s, which can be obtained as follows:

$$\sum_{s=0}^{\infty} G_s^r \alpha^s (1-\alpha)^{s+1} = \frac{1-(\frac{\alpha}{1-\alpha})^r}{1-(\frac{\alpha}{1-\alpha})^{r+1}} \quad . \tag{29}$$

□

Lemma 6. *The probability that the block path starting at* $(0,0)$ *reaches the line* $y = x-r$ *for* $r \geq 1$ *before reaching the line* $y = x+1$ *is equal to* $1 - \frac{1-(\frac{\alpha}{1-\alpha})^r}{1-(\frac{\alpha}{1-\alpha})^{r+1}}$.

Proof. The event that the block path starting at $(0,0)$ reaches the line $y = x-r$ for $r \geq 1$ before reaching the line $y = x+1$ is the complement of the event introduced in Lemma 5. □

Lemma 7. *Let* F_s^r *for* $r \geq 1$ *denote the number of paths in a* (x,y)*-grid from start point* $(0,0)$ *to the point* $(s+r-1, s)$ *without reaching the lines* $y = x+1$ *and* $y = x-r$ *(define* F_0^1 *to be equal to 1). We have:*

$$\begin{aligned} \sum_{s=0}^{\infty} F_s^r \big(\alpha(1-\alpha)\big)^s &= \Big(\frac{1}{\alpha^r}\Big)\Big(1-\frac{1-(\frac{\alpha}{1-\alpha})^r}{1-(\frac{\alpha}{1-\alpha})^{r+1}}\Big) \quad , \\ \sum_{s=0}^{\infty} sF_s^r \big(\alpha(1-\alpha)\big)^s &= \frac{\alpha(r+1)(1+(\frac{\alpha}{1-\alpha})^r)}{(1-\alpha)^{r+1}(1-(\frac{\alpha}{1-\alpha})^{r+1})^2} - \\ &\frac{2\alpha}{(1-2\alpha)(1-\alpha)^r(1-(\frac{\alpha}{1-\alpha})^{r+1})} \quad . \end{aligned} \tag{30}$$

Proof. Assume $\texttt{Event}_1(s)$ is defined as follows: (i) the block path starting at $(0,0)$ reaches the point $(s+r-1, s)$, and (ii) before reaching the point $(s+r-1, s)$, the block path never reaches the lines $y = x + 1$ and $y = x - r$. According to the definition of F_s^r, the probability of $\texttt{Event}_1(s)$ is equal to $F_s^r \alpha^{s+r-1}(1-\alpha)^s$. Assume $\texttt{Event}_2(s)$ is defined as follows: (i) the block path starting at $(0, 0)$ reaches the line $y = x - r$ for the first time at point $(s + r, s)$, and (ii) before reaching the point $(s + r, s)$, the block path never reaches the line $y = x + 1$. $\texttt{Event}_2(s)$ happens if, after the occurrence of $\texttt{Event}_1(s)$ and reaching the point $(s + r - 1, s)$, the block path immediately moves one step to the right to pass to reach the point $(s + r, s)$, which is located in the line $y = x - r$. Therefore, the probability of $\texttt{Event}_2(s)$ is equal to $F_s^r \alpha^{s+r}(1-\alpha)^s$. The sum of $\texttt{Event}_2(s)$ probabilities over all values of s is equal to the probability that the block path starting at $(0, 0)$ reaches the line $y = x - r$ before reaching the line $y = x + 1$, the probability of which is calculated in Lemma 6. Therefore, we have:

$$\sum_{s=0}^{\infty} F_s^r \alpha^{s+r}(1-\alpha)^s = 1 - \frac{1 - (\frac{\alpha}{1-\alpha})^r}{1 - (\frac{\alpha}{1-\alpha})^{r+1}} \ , \tag{31}$$

which results in the first equality in Lemma 7. To prove the second equality in Lemma 7, we use the variable substitution $\alpha(1 - \alpha) = x$ in the equality above. We have:

$$\sum_{s=0}^{\infty} F_s^r x^s = \Big(\frac{2}{1-\sqrt{1-4x}}\Big)^r \Big(1 - \frac{1 - (\frac{1-\sqrt{1-4x}}{1+\sqrt{1-4x}})^r}{1 - (\frac{1-\sqrt{1-4x}}{1+\sqrt{1-4x}})^{r+1}}\Big) \ . \tag{32}$$

By taking the derivative from both sides, then multiplying both sides to x, and finally substituting $x = \alpha(1 - \alpha)$, the second equality in Lemma 7 can be obtained. □

Lemma 8. *The average adversarial and honest rewards resulting from the transition from state $S_{1,2}$ to state $S_{0,0}$ under $\texttt{action}_2$ (Scenario 2 occurrence) can be obtained as follows:*

$$\begin{aligned} &R_{\mathcal{A}} = R_3 = 3\alpha + \\ &\sum_{r=1}^{\infty}\sum_{s=0}^{\infty} (2+s)G_s^r \alpha^{1+s}(1-\alpha)^{r+s+1} + (3+r+s)F_s^r \alpha^{1+r+s}(1-\alpha)^{r+s} \ , \\ &R_{\mathcal{H}} = R_4 = \sum_{r=1}^{\infty}\sum_{s=0}^{\infty} (2+r)G_s^r \alpha^{1+s}(1-\alpha)^{r+s+1} \ , \end{aligned} \tag{33}$$

where the series above can be calculated using series introduced in Lemmas 4 and 7.

Proof. If assuming $L = 0$ (recall that L denotes the number of consecutive honest blocks proposed after being at state $S_{1,2}$ and before the next adversarial block is proposed), Scenario 2a is equivalent to the event that the block path starting at $(1, 2)$ reaches the point $(3, 2)$. The probability of this event is α^2, and its corresponding adversarial reward and honest reward are equal to 3 and 0, respectively.

Consider the picture depicted in Figure 8. If assuming $L = r$, where $r \geq 1$, Scenario 2a is equivalent to the event that the block path starting at $(1, 2)$ reaches the point $(1, 2+r)$, then moves 2 steps to the right to reach the point $(3, 2+r)$, and then reaches one of the points $\{(3+r+i, 2+r+i) | i \geq 0\}$ (line $y = x - 1$) without reaching any of the points $\{(3+i, 3+i+r) | i \geq 0\}$ (line $y = x+r$). The probability that the block path starting at $(1, 2)$ reaches the point $(1, 2+r)$ and then moves to the point $(3, 2+r)$ is equal to $\alpha^2(1-\alpha)^r$. Assume the block path reaches the line $y = x-1$ at point $(3+r+s, 2+r+s)$. The event that the block path starting at the point $(3, 2+r)$ reaches the line $y = x - 1$ for the first time at point $(3+r+s, 2+r+s)$ without reaching the line $\{(3+i, 3+i+r) | i \geq 0\}$ beforehand is equivalent to the event that that the block path starting at $(0, 0)$ reaches the line $y = x - r$ for the first time at point $(r + s, s)$ without reaching the line $y = x + 1$ beforehand. The probability of the latter event is equal to $F_s^r \alpha^{r+s}(1-\alpha)^s$, where F_s^r for $r \geq 1$ is introduced in Lemma 7. Therefore, the probability that the block path starting at $(1, 2)$ moves to the points $(1, 2+r)$, and then moves to the point $(3, 2+r)$, and finally reaches point $(3+r+s, 2+r+s)$ before reaching the line $\{(3+i, 3+i+r) | i \geq 0\}$ is equal to $F_s^r \alpha^{r+s+2}(1-\alpha)^{s+r}$. If the block path reaches the line $y = x - 1$ at point $(3 + r + s, 2 + r + s)$, the adversarial reward and the honest reward resulting from Scenario 2a is equal to $3 + r + s$ and 0, respectively. Therefore, the average adversarial reward and honest reward under Scenario 2a can be obtained as follows:

$$\begin{aligned} R'_3 &= \frac{3\alpha^2 + \sum_{r=1}^{\infty}\sum_{s=0}^{\infty}(3+r+s)F_s^r\alpha^{r+s+2}(1-\alpha)^{r+s}}{\texttt{Pr}(\text{Scenario 2a})}\ , \\ R'_4 &= 0\ . \end{aligned} \tag{34}$$

If assuming $L = 0$, Scenario 2b cannot happen.

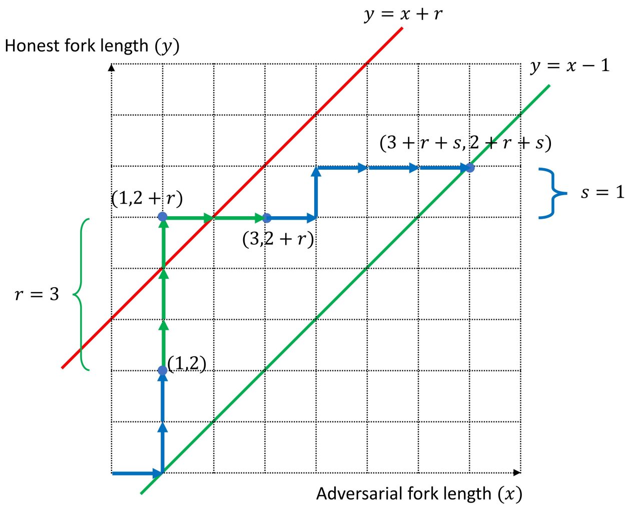


Fig. 8: Block path in Scenario 2a

Consider the picture depicted in Figure 9. If assuming $L = r$, where $r \geq 1$, Scenario 2b is equivalent to the event that the block path starting at $(1, 2)$ reaches the point $(1, 2+r)$, then moves 2 steps to the right to reach the point $(3, 2+r)$, and then reaches one of the points $\{(3+i, 3+i+r)|i \geq 0\}$ (line $y = x+r$) before reaching any of the points $\{(3+r+i, 2+r+i)|i \geq 0\}$ (line $y = x-1$). The probability that the block path starting at $(1, 2)$ reaches the point $(1, 2+r)$ and then moves to the point $(3, 2+r)$ is equal to $\alpha^2(1-\alpha)^r$. Assume the block path reaches the line $y = x+r$ at point $(3+s, 3+r+s)$. The event that the block path starting at the point $(3, 2+r)$ reaches the line $y = x+r$ for the first time at point $(3+s, 3+r+s)$ without reaching the line $y = x-1$ beforehand is equivalent to the event that the block path starting at $(0, 0)$ reaches the line $y = x+1$ for the first time at point $(s, s+1)$ without reaching the line $y = x-r$ beforehand. The probability of the latter event is equal to $G_s^r \alpha^s (1-\alpha)^{s+1}$, where G_s^r for $r \geq 1$ is introduced in Lemma 4. Therefore, the probability that the block path starting at $(1, 2)$ moves to the point $(1, 2+r)$, and then moves to the point $(3, 2+r)$, and finally reaches the point $(3+s, 3+r+s)$ before reaching the line $y = x-1$ is equal to $G_s^r \alpha^{s+2} (1-\alpha)^{r+s+1}$. If the block path reaches the line $y = x+r$ at point $(3+s, 3+r+s)$, the adversarial reward and the honest reward resulting from Scenario 2b is equal to $2+s$ and $2+r$, respectively. Therefore, the average adversarial reward and honest reward under Scenario 2b can be obtained as follows:

$$\begin{aligned}
R_3'' &= \frac{\sum_{r=1}^{\infty} \sum_{s=0}^{\infty} (2+s) G_s^r \alpha^{2+s} (1-\alpha)^{r+s+1}}{\mathtt{Pr}(\text{Scenario 2b})} \quad , \\
R_4'' &= \frac{\sum_{r=1}^{\infty} \sum_{s=0}^{\infty} (2+r) G_s^r \alpha^{2+s} (1-\alpha)^{r+s+1}}{\mathtt{Pr}(\text{Scenario 2b})} \quad .
\end{aligned} \tag{35}$$

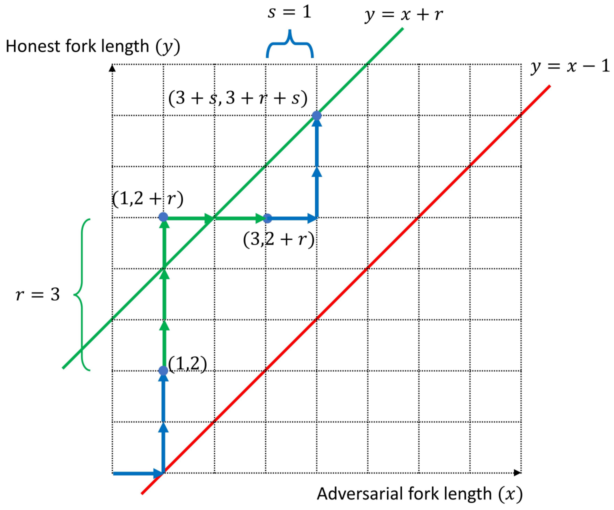


Fig. 9: Block path in Scenario 2b

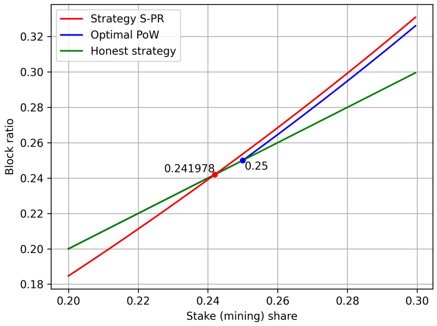


Fig. 10: Block ratio comparison (communication capability $\eta = 0.5$)

Finally, the average adversarial and honest rewards resulting from the transition from state $S_{1,2}$ to state $S_{0,0}$ under $\texttt{action}_2$ (Scenario 2 occurrence) can be obtained as follows:

$$\begin{aligned} R_{\mathcal{A}} = R_3 &= \frac{\texttt{Pr}(\text{Scenario 2a})R'_3 + \texttt{Pr}(\text{Scenario 2b})R''_3}{\texttt{Pr}(\text{Scenario 2})} \ , \\ R_{\mathcal{H}} = R_4 &= \frac{\texttt{Pr}(\text{Scenario 2a})R'_4 + \texttt{Pr}(\text{Scenario 2b})R''_4}{\texttt{Pr}(\text{Scenario 2})} \ . \end{aligned} \tag{36}$$

As $\texttt{Pr}(\text{Scenario 2}) = \alpha$, the equations presented in 33 can be obtained. □

Having access to R_1, R_2, R_3, and R_4, we can use the Markov chain depicted in Figure 5, to calculate the block ratio achieved by following strategy $\pi^{\text{S-PR}}$. In Figure 10, we compare the block ratio achieved by strategy $\pi^{\text{S-PR}}$ with those achieved by the honest strategy and the optimal PoW selfish mining strategy [22] for an attacker with communication capability $\eta = 0.5$. As can be seen in Figure 5, strategy $\pi^{\text{S-PR}}$ can dominate the optimal PoW selfish mining strategy for some range of stake shares.

In Figure 11, we compare the block ratio achieved by strategy $\pi^{\text{S-PR}}$ with those achieved by the honest strategy, the optimal PoW selfish mining strategy [22], and the nothing-at-stake selfish mining strategy introduced in [13] (denoted by S-PR2) for an attacker with communication capability $\eta = 0$. Strategy $\pi^{\text{S-PR}}$ can dominate both the optimal PoW selfish mining strategy and the strategy S-PR2 for some range of stake shares.

E Strategy π^{SP1} in full-predictable protocols

We use a two-dimensional (x, y)-grid to depict the chain race. The proposing sequence is represented by a path on this grid, which is referred to as the"block path". Whenever the time slot is honest, i.e., the block proposer of the time slot is honest, the block path moves one step up. Whenever the time slot is

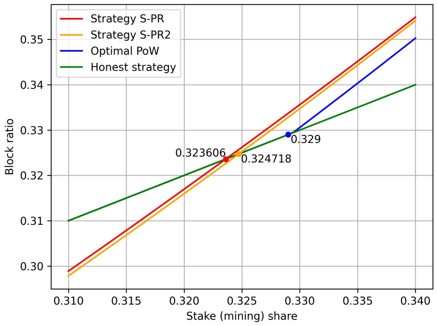


Fig. 11: Block ratio comparison (communication capability $\eta = 0$)

adversarial, i.e., the block proposer of the time slot is the attacker, the block path moves one step to the right.

To calculate the block ratio of strategy π^{SP1}, we first present two helpful lemmas.

Lemma 9. *The probability that the block path starting at* $(0, 0)$ *never reaches the line* $y = x$ *for* $x \geq 1$ *is equal to* $1 - 2\alpha$.

Proof. To never reach the line $y = x$ for $x \geq 1$, the block path should move one step up to the point $(0, 1)$ as the first step, the probability of which is equal to $1 - \alpha$. We need to calculate the probability of the event that the block path starting at $(0, 1)$ never reaches the line $y = x$ for $x \geq 1$. To this end, we calculate the probability of the complementary event, i.e., the event that the block path starting at $(0, 1)$ reaches the line $y = x$ at least once. Assume the block path starting at $(0, 1)$ reaches the line $y = x$ for the first time at point (s, s) for $s \geq 1$. The number of paths from point $(0, 1)$ to the point (s, s) for $s \geq 1$ without reaching the line $y = x$ before point (s, s) is equal to $s - 1^{\text{th}}$ Catalan number denoted by c_{s-1}. Therefore, the probability of the event that the block path starting at $(0, 1)$ reaches the line $y = x$ for the first time at point (s, s) for $s \geq 1$ is equal to $c_{s-1}\alpha^s(1-\alpha)^{s-1}$. As a result, the probability of the event that the block path starting at $(0, 1)$ reaches the line $y = x$ at least once can be obtained as follows:

$$\sum_{s=1}^{\infty} c_{s-1}\alpha^s(1-\alpha)^{s-1} = \alpha \sum_{s=0}^{\infty} c_s(\alpha(1-\alpha))^s = \frac{\alpha}{1-\alpha} \quad . \tag{37}$$

The formula for calculating the series in the equation above is presented in [8]. Therefore, the probability of the event that the block path starting at $(0, 1)$ never reaches the line $y = x$ is equal to $1 - \frac{\alpha}{1-\alpha} = \frac{1-2\alpha}{1-\alpha}$. Finally, the probability that the block path starting at $(0, 0)$ never reaches the line $y = x$ for $x \geq 1$ is equal to $(1-\alpha)\frac{1-2\alpha}{1-\alpha} = 1 - 2\alpha$. □

Lemma 10. *The probability that the block path starting at* $(0,0)$ *reaches the line* $y = x - 1$ *but never passes it is equal to* $\frac{\alpha(1-2\alpha)}{(1-\alpha)^2}$.

Proof. The number of paths from point $(0,0)$ to point $(s+1,s)$ without passing the line $y = x-1$ is equal to $s+1^{\text{th}}$ Catalan number denoted by c_{s+1}. Therefore, the probability of the event that the block path starting at point $(0,0)$ reaches the line $y = x-1$ at point $(s+1,s)$ without passing the line $y = x-1$ is equal to $c_{s+1}\alpha^{s+1}(1-\alpha)^s$. The event that the block path starting at point $(s+1,s)$ never reaches the line $y = x-1$ again is equivalent to the event that the block path starting at $(0,0)$ never reaches the line $y = x$ again, the probability of which is presented in Lemma 9. Therefore, the probability of the event that the block path starting at point $(0,0)$ reaches the line $y = x-1$ for the last time at point $(s+1,s)$ without ever passing the line $y = x-1$ is equal to $c_{s+1}\alpha^{s+1}(1-\alpha)^s(1-2\alpha)$. Therefore, the probability that the block path starting at $(0,0)$ reaches the line $y = x-1$ but never passes it can be obtained as follows:

$$\begin{aligned}\sum_{s=0}^{\infty} c_{s+1}\alpha^{s+1}(1-\alpha)^s(1-2\alpha) &= \frac{1-2\alpha}{1-\alpha}\sum_{s=1}^{\infty} c_s(\alpha(1-\alpha))^s \\ &= \frac{1-2\alpha}{1-\alpha}(\frac{1}{1-\alpha}-1) = \frac{\alpha(1-2\alpha)}{(1-\alpha)^2} \quad .\end{aligned} \tag{38}$$

□

Following strategy SP1, the main chain can be divided into two recurrent cycles: the adversarial cycle and the honest cycle. An adversarial cycle starts at time slot t, where t represents the time slot whose corresponding longest dominant chain is a non-empty set, i.e., $L^{\texttt{LDC}}(t) > 0$. During adversarial cycles, a set of consecutive adversarial blocks is added to the main chain, and honest blocks get orphaned. Between every two adversarial cycles, there is an honest cycle, where a set of consecutive honest blocks is added to the main chain. Honest cycles include the time slots whose corresponding longest dominant chain is an empty set. Therefore, to obtain the block ratio, the average length of both honest and adversarial cycles should be calculated.

Lemma 11. *Let* $L_{\mathcal{H}}^{\mathit{SP1}}$ *denote the length of the honest cycle* $\texttt{Cycle}^{\mathcal{H}}$ *under the strategy SP1. We have:*

$$\mathbb{E}(L_{\mathcal{H}}^{\mathit{SP1}}) = \frac{1+\alpha}{\alpha} \quad . \tag{39}$$

Proof. Assume the honest cycle $\texttt{Cycle}^{\mathcal{H}}$ starts at time slot t from the point $(0,0)$ as depicted in Figure 12. The assumption that $\texttt{Cycle}^{\mathcal{H}}$ starts from point $(0,0)$ results in the block path never reaching the line $y = x$ at $x > 0$. To illustrate this fact, assume the previous adversarial cycle before $\texttt{Cycle}^{\mathcal{H}}$ starts at time slot t', where $t' < t$, and ends at time slot $t-1$. Assume further that there exists a time slot such as $t'' > t$ at which the block path reaches the line $y = x$ at $x > 0$. In this case, the attacker could win the chain race within the interval $[t', t'']$, which is against our assumption that the chain race within the interval $[t', t-1]$ is the

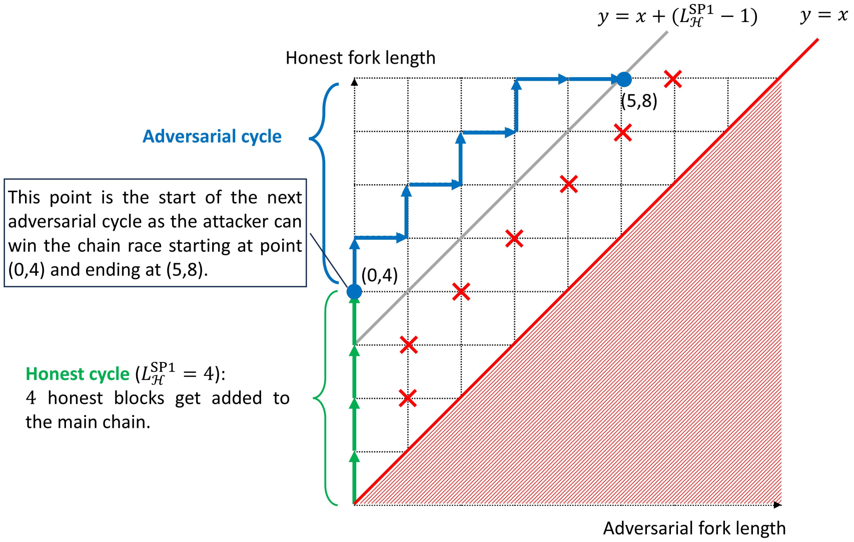

$y = x + (L_{\mathcal{H}}^{\text{SP1}} - 1)$
$y = x$
Honest fork length
Adversarial cycle
This point is the start of the next adversarial cycle as the attacker can win the chain race starting at point (0,4) and ending at (5,8).
(0,4)
(5,8)
Honest cycle ($L_{\mathcal{H}}^{\text{SP1}} = 4$):
4 honest blocks get added to the main chain.
Adversarial fork length

Fig. 12: Honest cycle of the strategy SP1

longest dominant chain starting at time slot t'. It is obvious that $L_{\mathcal{H}}^{\text{SP1}}$ cannot be equal to 0 or 1 because in such cases the block path reaches the line $y = x$ at $x > 0$. Therefore, $L_{\mathcal{H}}^{\text{SP1}} \geq 2$.

Knowing that the block path never reaches the line $y = x$ at $x > 0$, $L_{\mathcal{H}}^{\text{SP1}} = i$ represents the event that the block path reaches the line $y = x + (i-1)$ at $x > 0$ without reaching any of the lines $y = x + (j-1)$ at $x > 0$ for $1 < j < i$ given that the block path never reaches the line $y = x$. This means that during the honest cycle, the block path goes i steps up (i consecutive honest blocks are added to the main chain within the interval $[t, t + (i-1)]$) to reach point $(0, i)$, and in the subsequent adversarial cycle which starts at $t + i$, the block path reaches the line $y = x + (i-1)$ at $x > 0$ starting from point $(0, i)$ without never passing it. Reaching the line $y = x + (i-1)$ at $x > 0$ starting from point $(0, i)$ indicates that $\texttt{LDC}(t + i) > 0$, and thus, slot $t + i$ is the starting slot of a new adversarial cycle. Note that the block path cannot pass the line $y = x + (i-1)$ at $x > 0$ because otherwise there should have been a time slot $t' \in [t, t + (i-1)]$ with $L^{\texttt{LDC}}(t') > 0$, which means that the honest cycle length should have been less than i. Let E_1 and E_2 represent the event that the block path starting at point $(0, 0)$ reaches the line $y = x + (i-1)$ at $x > 0$ without reaching any of the lines $y = x + (j-1)$ at $x > 0$ for $1 < j < i$ and the event that the block path starting at point $(0, 0)$ never reaches the line $y = x$ at $x > 0$, respectively.

$$\texttt{Pr}(L_{\mathcal{H}}^{\text{SP1}} = i) = \texttt{Pr}(E_1 | E_2) = \frac{\texttt{Pr}(E_1 \cap E_2)}{\texttt{Pr}(E_2)} \quad . \tag{40}$$

Event $E_1 \cap E_2$ represents the event that the block path reaches the line $y = x + (i-1)$ at $x > 0$ without reaching any of the lines $y = x + (j-1)$ at $x > 0$

for $0 < j < i$. The probability that the block path reaches point $(0, i)$ is equal to $(1-\alpha)^i$. The event that the block path starting at point $(0, i)$ reaches the line $y = x + (i-1)$ but never passes it is equivalent to the event that the block path starting at point $(0, 0)$ reaches the line $y = x - 1$ but never passes it. The probability of the latter event, which is presented in Lemma 10, is equal to $\frac{\alpha(1-2\alpha)}{(1-\alpha)^2}$. Therefore, for $i \geq 2$, we have:

$$\mathtt{Pr}(E_1 \cap E_2) = (1-\alpha)^i \frac{\alpha(1-2\alpha)}{(1-\alpha)^2} \quad . \tag{41}$$

As according to Lemma 9, the probability of event E_2 is equal to $1-2\alpha$, $\mathtt{Pr}(L_{\mathcal{H}}^{\mathtt{SP1}} = i)$ for $i \geq 2$ can be obtained as follows:

$$\mathtt{Pr}(L_{\mathcal{H}}^{\mathtt{SP1}} = i) = (1-\alpha)^i \frac{\alpha}{(1-\alpha)^2} \quad . \tag{42}$$

Therefore, the average length of an honest cycle can be calculated as follows:

$$\mathbb{E}(L_{\mathcal{H}}^{\mathtt{SP1}}) = \sum_{i=2}^{\infty} i \cdot \mathtt{Pr}(L_{\mathcal{H}}^{\mathtt{SP1}} = i) = \frac{1+\alpha}{\alpha} \quad . \tag{43}$$

□

Lemma 12. *Let $L_{\mathcal{A}}^{SP1}$ denote the length of an adversarial cycle $\mathtt{Cycle}^{\mathcal{A}}$ under the strategy SP1. We have:*

$$\mathbb{E}(L_{\mathcal{A}}^{SP1}) = \frac{1}{1-2\alpha} \quad . \tag{44}$$

Proof. Assume the adversarial cycle $\mathtt{Cycle}^{\mathcal{A}}$ starts at time slot t from the point $(0, 0)$ as depicted in Figure 13. The assumption that $\mathtt{Cycle}^{\mathcal{A}}$ starts at point $(0, 0)$ results in the event that the block path starting at $(0, 0)$ reaches the line $y = x - 1$ but never passes it. Because, if the block path starting at $(0, 0)$ never reaches the line $y = x - 1$, then the $\mathtt{LDC}(t)$ would be an empty set, indicating that the time slot t cannot be the starting time slot of an adversarial cycle. Besides, if the block path starting at $(0, 0)$ passes the line $y = x - 1$, there should be a time slot $t' < t$ with $L^{\mathtt{LDC}}(t') > 0$ within the previous honest cycle. This indicates that the adversarial cycle $\mathtt{Cycle}^{\mathcal{A}}$ should start at an earlier time slot rather than t, which is against our assumption.

Knowing that the block path reaches the line $y = x - 1$ but never passes it, $L_{\mathcal{A}}^{\mathtt{SP1}} = i$ represents the event that the block path reaches the line $y = x - 1$ for the last time at point $(i, i-1)$, where $i \geq 1$, given that the block path reaches the line $y = x - 1$ but never passes it. Let E_1 and E_2 represent the event that the block path reaches the line $y = x - 1$ for the last time at point $(i, i-1)$ and the event that the block path reaches the line $y = x - 1$ but never passes it. We have:

$$\mathtt{Pr}(L_{\mathcal{A}}^{\mathtt{SP1}} = i) = \mathtt{Pr}(E_1 | E_2) = \frac{\mathtt{Pr}(E_1 \cap E_2)}{\mathtt{Pr}(E_2)} \quad . \tag{45}$$

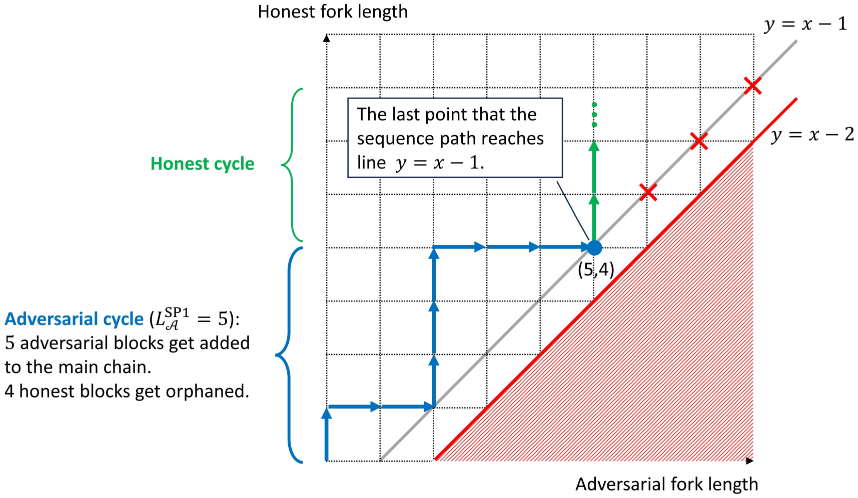

Honest fork length
$y = x - 1$
$y = x - 2$
The last point that the sequence path reaches line $y = x - 1$.
Honest cycle
(5,4)
Adversarial cycle ($L_{\mathcal{A}}^{\text{SP1}} = 5$):
5 adversarial blocks get added to the main chain.
4 honest blocks get orphaned.
Adversarial fork length

Fig. 13: Adversarial cycle of the strategy SP1

Event $E_1 \cap E_2$ represents the event that the block path reaches the line $y = x - 1$ for the last time at point $(i, i-1)$ without ever passing it. According to the proof presented in Lemma 10, the probability $\texttt{Pr}(E_1 \cap E_2)$ can be calculated as follows:

$$\texttt{Pr}(E_1 \cap E_2) = c_i \alpha^i (1-\alpha)^{i-1}(1-2\alpha) \quad . \tag{46}$$

Since according to Lemma 10, the probability of event E_2 is equal to $\frac{\alpha(1-2\alpha)}{(1-\alpha)^2}$, $\texttt{Pr}(L_{\mathcal{A}}^{\texttt{SP1}} = i)$ can be obtained as follows:

$$\texttt{Pr}(L_{\mathcal{A}}^{\texttt{SP1}} = i) = \frac{c_i \alpha^i (1-\alpha)^{i-1}(1-2\alpha)}{\frac{\alpha(1-2\alpha)}{(1-\alpha)^2}} \quad . \tag{47}$$

Therefore, the average length of an adversarial cycle can be obtained as follows:

$$\mathbb{E}(L_{\mathcal{A}}^{\texttt{SP1}}) = \sum_{i=1}^{\infty} i \cdot \texttt{Pr}(L_{\mathcal{A}}^{\texttt{SP1}} = i) = \frac{1}{1-2\alpha} \quad . \tag{48}$$

Note that during an adversarial cycle with length $L_{\mathcal{A}}^{\texttt{SP1}} = i$, i adversarial blocks are added to the main chain, and $i-1$ honest blocks get orphaned. □

Proof (Proof of Theorem 1). Using Lemmas 11 and 12, the block ratio of an attacker under strategy $\pi^{\texttt{SP1}}$ can be obtained as follows:

$$\texttt{BlkRatio}_{\mathcal{A}}(\pi^{\texttt{SP1}}) = \frac{\mathbb{E}(L_{\mathcal{A}}^{\texttt{SP1}})}{\mathbb{E}(L_{\mathcal{A}}^{\texttt{SP1}}) + \mathbb{E}(L_{\mathcal{H}}^{\texttt{SP1}})} = \frac{\alpha}{1-2\alpha^2} \quad . \tag{49}$$

□

F Strategy π^{SP2} in full-predictable protocols

To calculate the block ratio of strategy π^{SP2}, we first present a couple of helpful lemmas.

Lemma 13. *If assuming the number of paths in a (x, y)-grid from start point $(0, 0)$ to the point (s, s) without reaching the line $y = x - r$ for $r \geq 1$ is denoted by D^r_s (define D^r_0 to be equal to 1), then we have:*

$$\begin{aligned}&\sum_{s=0}^{\infty} D^r_s \big(\alpha(1-\alpha)\big)^s = \frac{1}{1-2a}\Big(1 - (\frac{\alpha}{1-\alpha})^r\Big) \ ,\\ &\sum_{s=0}^{\infty} sD^r_s \big(\alpha(1-\alpha)\big)^s = \frac{2\alpha(1-\alpha)}{(1-2\alpha)^3}\\ &-\Big(\frac{\alpha}{1-\alpha}\Big)^r\Big(\frac{2\alpha(1-\alpha)+r(1-2\alpha)}{(1-2\alpha)^2}\Big) \ .\end{aligned} \tag{50}$$

Proof. Let $r \geq 1$. The probability that a block path starting at $(0, 0)$ reaches the point (s, s) without reaching the line $y = x - r$ is equal to $D^r_s\big(\alpha(1-\alpha)\big)^s$. Therefore, the probability that a block path starting at $(0, 0)$ reaches the line $y = x$ for the last time at the point (s, s) without ever reaching the line $y = x - r$ is equal to $P^r_s = D^r_s(1-2\alpha)\big(\alpha(1-\alpha)\big)^s$. The sum of all the probabilities P^r_s for $s \in \mathbb{W}$ is the same as the probability that a block path never reaches the line $y = x - r$. Therefore, using Lemma 3, we obtain:

$$\begin{aligned}&\sum_{s=0}^{\infty} D^r_s(1-2\alpha)\big(\alpha(1-\alpha)\big)^s = 1 - (\frac{\alpha}{1-\alpha})^r \Rightarrow\\ &\sum_{s=0}^{\infty} D^r_s \big(\alpha(1-\alpha)\big)^s = \frac{1}{1-2a}\Big(1 - (\frac{\alpha}{1-\alpha})^r\Big) \ .\end{aligned} \tag{51}$$

To prove the second equality in Lemma 13, we use the variable substitution $\alpha(1-\alpha) = x$ in the equality above. We have:

$$\sum_{s=0}^{\infty} D^r_s x^s = \frac{1 - \Big(\frac{1-\sqrt{1-4x}}{1+\sqrt{1-4x}}\Big)^r}{\sqrt{1-4x}} \ . \tag{52}$$

By taking the derivative from both sides, we obtain:

$$\begin{aligned}&\sum_{s=0}^{\infty} sD^r_s x^{s-1} = \frac{2}{\sqrt{(1-4x)^3}} +\\ &\frac{4(1-\sqrt{1-4x})^{r-1}\Big(8x^2 - 2x - r\sqrt{(1-4x)^3}\Big)}{(1+\sqrt{1-4x})^{r+1}\sqrt{(1-4x)^5}} \ .\end{aligned} \tag{53}$$

By multiplying both sides to x and substituting $x = \alpha(1-\alpha)$, the second equality in Lemma 13 can be obtained. □

Following strategy SP2, the main chain can be divided into two recurrent cycles: the adversarial cycle and the honest cycle. Note that in strategy SP2, an adversarial cycle always starts at an adversarial time slot. To obtain the block ratio of strategy SP2, we first calculate the average length of both honest and adversarial cycles.

Lemma 14. *Let $L_{\mathcal{H}}^{SP2}$ denote the length of the honest cycle $\mathtt{Cycle}^{\mathcal{H}}$ under the strategy SP2. We have:*

$$\mathbb{E}(L_{\mathcal{H}}^{SP2}) = \frac{1}{\alpha(1-\alpha)} \quad . \tag{54}$$

Proof. Assume the honest cycle $\mathtt{Cycle}^{\mathcal{H}}$ starts at time slot t from the point $(0,0)$ as depicted in Figure 14. The assumption that $\mathtt{Cycle}^{\mathcal{H}}$ starts from the point $(0,0)$ results in the event that the block path starting at point $(0,0)$ never reaches the line $y = x$ at $x > 0$. The reason is the same as the explanation presented in Lemma 11. It is obvious that $L_{\mathcal{H}}^{\mathsf{SP2}}$ cannot be equal to 0 or 1 because in such cases, the block path reaches the line $y = x$ at $x > 0$. Therefore, $L_{\mathcal{H}}^{\mathsf{SP2}} \geq 2$.

Knowing that the block path never reaches the line $y = x$ at $x > 0$, $L_{\mathcal{H}}^{\mathsf{SP2}} = i$ represents the event that the block path reaches point $(0, i)$ and then moves to point $(1, i)$ given that it never reaches the line $y = x$ at $x > 0$. It means that during the honest cycle, the block path moves i steps up to reach point $(0, i)$ (i consecutive honest blocks are added to the main chain within the interval $[t, t + (i - 1)]$), and then, the block path moves one step right at time slot $t + i$ to reach point $(1, i)$. This indicates that time slot $t + i$ is the starting time slot of the subsequent adversarial cycle. Let E_1 and E_2 represent the event that the

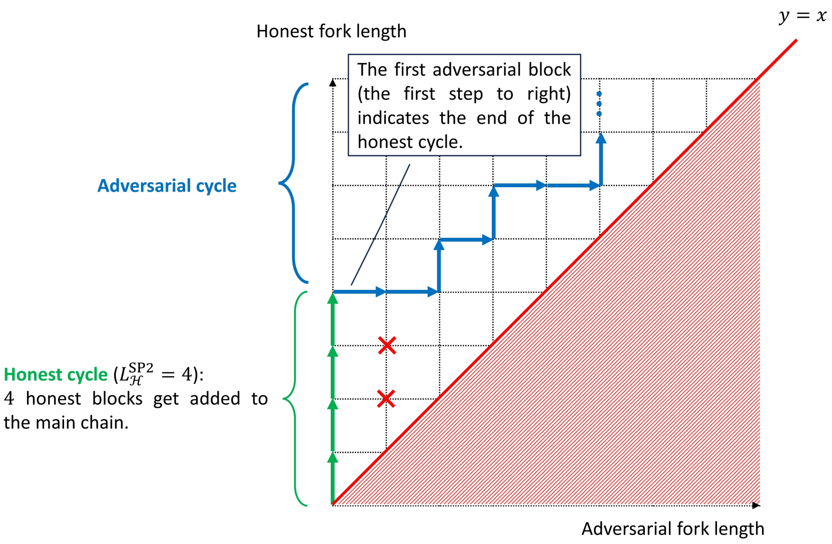

$y = x$
Honest fork length
The first adversarial block (the first step to right) indicates the end of the honest cycle.
Adversarial cycle
Honest cycle ($L_{\mathcal{H}}^{\mathsf{SP2}} = 4$):
4 honest blocks get added to the main chain.
Adversarial fork length

Fig. 14: Honest cycle of the strategy SP2

block path starting at point $(0,0)$ reaches point $(0,i)$ and then point $(1,i)$ and the event that the block path starting at point $(0,0)$ never reaches the line $y=x$ at $x>0$, respectively.

$$\texttt{Pr}(L_{\mathcal{H}}^{\texttt{SP2}}=i)=\texttt{Pr}(E_1|E_2)=\frac{\texttt{Pr}(E_1\cap E_2)}{\texttt{Pr}(E_2)} \quad . \tag{55}$$

Event $E_1\cap E_2$ represents the event that the block path reaches point $(0,i)$ and then point $(1,i)$ without ever reaching the line $y=x$ at any future points with $x>0$. The probability that the block path reaches point $(0,i)$ and then point $(1,i)$ is equal to $(1-\alpha)^i\alpha$. The event that the block path starting at point $(1,i)$ never reaches the line $y=x$ at $x>0$ is equivalent to the event of never reaching the line $y=x-(i-1)$ starting at point $(0,0)$. The probability of the latter event, which is presented in Lemma 3, is equal to $1-(\frac{\alpha}{1-\alpha})^{i-1}$. For $i\geq 2$, we have:

$$\texttt{Pr}(E_1\cap E_2)=(1-\alpha)^i\alpha\Big(1-(\frac{\alpha}{1-\alpha})^{i-1}\Big) \quad . \tag{56}$$

Since, according to Lemma 9, the probability of event E_2 is equal to $1-2\alpha$, $\texttt{Pr}(L_{\mathcal{H}}^{\texttt{SP2}}=i)$ for $i\geq 2$ can be obtained as follows:

$$\texttt{Pr}(L_{\mathcal{H}}^{\texttt{SP2}}=i)=\frac{(1-\alpha)^i\alpha\Big(1-(\frac{\alpha}{1-\alpha})^{i-1}\Big)}{1-2\alpha} \quad . \tag{57}$$

Therefore, the average length of an honest cycle can be calculated as follows:

$$\mathbb{E}(L_{\mathcal{H}}^{\texttt{SP2}})=\sum_{i=2}^{\infty}i\cdot\texttt{Pr}(L_{\mathcal{H}}^{\texttt{SP2}}=i)=\frac{1}{\alpha(1-\alpha)} \quad . \tag{58}$$

□

Lemma 15. *Let $L_{\mathcal{A}}^{SP2}$ denote the length of an adversarial cycle under strategy SP2. We have:*

$$\mathbb{E}(L_{\mathcal{A}}^{SP2})=\frac{1-\alpha(1-\alpha)}{(1-\alpha)(1-2\alpha)} \quad . \tag{59}$$

Proof. Assume the adversarial cycle $\texttt{Cycle}_{\texttt{new}}^{\mathcal{A}}$ starts at time slot t from the point $(0,0)$ as depicted in Figure 15. Assume further that before $\texttt{Cycle}_{\texttt{new}}^{\mathcal{A}}$, there exists an honest cycle $\texttt{Cycle}^{\mathcal{H}}$ whose length is equal to k. Since the adversarial cycle always starts with a time slot whose proposer is adversarial, the block path always moves one step to the right to reach the point $(1,0)$. The assumptions that $\texttt{Cycle}_{\texttt{new}}^{\mathcal{A}}$ starts from point $(0,0)$ and the length of the previous honest cycle is equal to k result in the event that the block path never reaches the line $y=x-k$. To illustrate this fact, assume that before $\texttt{Cycle}^{\mathcal{H}}$, there exists another adversarial cycle $\texttt{Cycle}_{\texttt{old}}^{\mathcal{A}}$ that starts at time slot t_1 and ends at t_2. Assume further that there exists a time slot such as $t_3>t$ at which the block path reaches the line $y=x-k$. In this case, the attacker can win the chain race within the interval $[t_1,t_3]$, which is against our assumption that the chain race within the interval $[t_1,t_2]$ is the longest dominant chain starting at t_1.

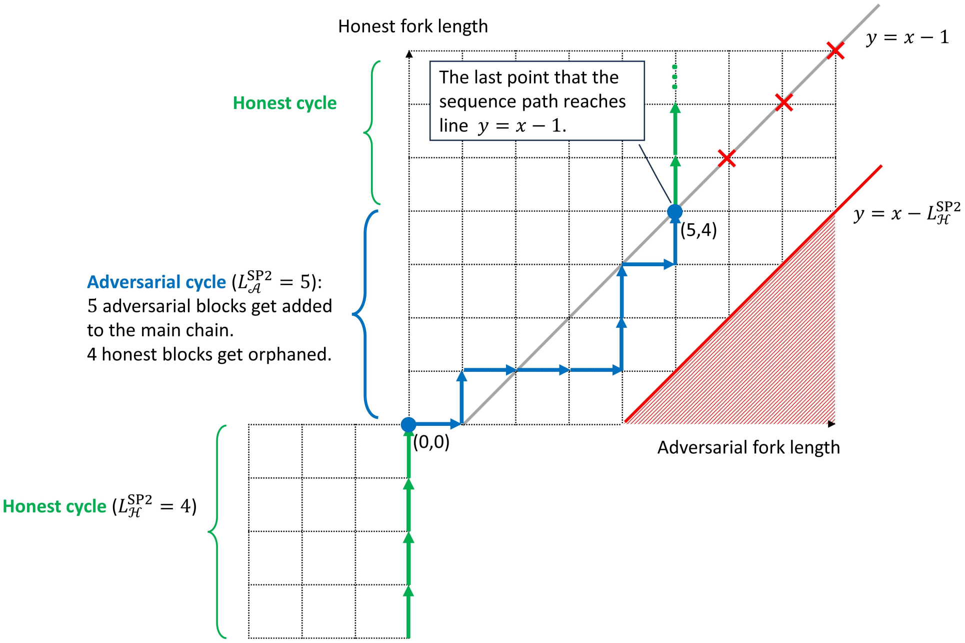

Honest fork length
Honest cycle
The last point that the sequence path reaches line $y = x - 1$.
$y = x - 1$
$y = x - L_{\mathcal{H}}^{\mathsf{SP2}}$
(5,4)
Adversarial cycle ($L_{\mathcal{A}}^{\mathsf{SP2}} = 5$):
5 adversarial blocks get added to the main chain.
4 honest blocks get orphaned.
(0,0)
Adversarial fork length
Honest cycle ($L_{\mathcal{H}}^{\mathsf{SP2}} = 4$)

Fig. 15: Adversarial cycle of the strategy SP2

Knowing that the block path never reaches the line $y = x - k$, $L_{\mathcal{A}}^{\mathsf{SP2}} = i$ represents the event that the block path reaches the line $y = x - 1$ for the last time at point $(i, i-1)$ given that the block path never reaches the line $y = x - k$. Let E_1 and E_2 represent the event that the block path starting at point $(1, 0)$ reaches the line $y = x - 1$ for the last time at point $(i, i-1)$ and the event that the block path starting at point $(1, 0)$ never reaches the line $y = x - k$. We have:

$$\mathtt{Pr}(L_{\mathcal{A}}^{\mathsf{SP2}} = i) = \mathtt{Pr}(E_1 | E_2) = \frac{\mathtt{Pr}(E_1 \cap E_2)}{\mathtt{Pr}(E_2)} \ . \tag{60}$$

Event $E_1 \cap E_2$ represents the event that the block path reaches the line $y = x - 1$ for the last time at point $(i, i-1)$ without ever reaching the line $y = x - k$. The number of paths from point $(1, 0)$ to point $(i, i-1)$ without passing through the line $y = x - k$ is equal to D_{i-1}^{k-1} as introduced in Lemma 13. Therefore, the probability of the event that the block path starting at point $(1, 0)$ reaches the point $(i, i-1)$ without ever passing the line $y = x - k$ is equal to $D_{i-1}^{k-1}\big(\alpha(1-\alpha)\big)^{i-1}$. To ensure that the block path reaches the line $y = x - 1$ for the last time at point $(i, i-1)$, the block path should never reach the line $y = x - 1$ again starting from point $(i, i-1)$, the probability of which is equal to $1 - 2\alpha$. We have:

$$\mathtt{Pr}(E_1 \cap E_2) = D_{i-1}^{k-1}\big(\alpha(1-\alpha)\big)^{i-1}(1 - 2\alpha) \ . \tag{61}$$

Since according to Lemma 3, the probability of event E_2 is equal to $1-\left(\frac{\alpha}{1-\alpha}\right)^{k-1}$, $\mathtt{Pr}(L_{\mathcal{A}}^{\mathtt{SP2}}=i)$ can be obtained as follows:

$$\mathtt{Pr}(L_{\mathcal{A}}^{\mathtt{SP2}}=i)=\frac{D_{i-1}^{k-1}\big(\alpha(1-\alpha)\big)^{i-1}(1-2\alpha)}{1-\left(\frac{\alpha}{1-\alpha}\right)^{k-1}}\ . \tag{62}$$

Therefore, using Lemma 13, the average length of an adversarial cycle can be obtained as follows:

$$\mathbb{E}(L_{\mathcal{A}}^{\mathtt{SP2}})=\sum_{i=1}^{\infty}i\cdot\mathtt{Pr}(L_{\mathcal{A}}^{\mathtt{SP1}}=i)=\frac{1-\alpha(1-\alpha)}{(1-\alpha)(1-2\alpha)}\ . \tag{63}$$

Note that during an adversarial cycle with length $L_{\mathcal{A}}^{\mathtt{SP2}}=i$, i adversarial blocks are added to the main chain, and $i-1$ honest blocks get orphaned. □

Proof (Proof of Theorem 2). Using Lemmas 14 and 15, the block ratio of an attacker under strategy $\pi^{\mathtt{SP1}}$ can be obtained as follows:

$$\mathtt{BlkRatio}_{\mathcal{A}}(\pi^{\mathtt{SP2}})=\frac{\mathbb{E}(L_{\mathcal{A}}^{\mathtt{SP2}})}{\mathbb{E}(L_{\mathcal{A}}^{\mathtt{SP2}})+\mathbb{E}(L_{\mathcal{H}}^{\mathtt{SP2}})}=\frac{\alpha(1-\alpha(1-\alpha))}{(1-\alpha)^2(1+\alpha)}\ . \tag{64}$$

□

G Strategy $\pi^{\mathtt{O\text{-}SP}}$

The algorithm used in the description of strategy $\pi^{\mathtt{O\text{-}SP}}$ is presented in Algorithm 1.

H Justifying the optimality of strategy $\pi^{\mathtt{O\text{-}SP}}$

Let t_i and t_{i+1} be two consecutive checkpoint slots. We first discuss how to find the optimal selfish proposing strategy within the interval (t_i,t_{i+1}) and then prove that this optimal strategy is independent of the strategy that the attacker follows outside the interval (t_i,t_{i+1}). Assume $N^{\mathcal{A}}$ and $N^{\mathcal{H}}$ represent the number of adversarial blocks and the number of honest blocks within the interval (t_i,t_{i+1}), respectively. Note that $N^{\mathcal{H}}\geq N^{\mathcal{A}}-1$ since otherwise t_{i+1} could not be a checkpoint slot. When following the selfish proposing attack within the interval (t_i,t_{i+1}), not all the $N^{\mathcal{A}}+N^{\mathcal{H}}$ get added to the main chain due to the orphan occurrence. We want to calculate the number of adversarial and honest blocks added to the main chain under the optimal strategy. Due to the proposing sequence predictability in a full-predictable protocol, the attacker can publish his blocks in a way that none of them get orphaned. Therefore, all the $N^{\mathcal{A}}$ adversarial blocks within interval (t_i,t_{i+1}) can get added to the main chain. To calculate the number of min-chain honest blocks, we first define the term "adversarial cycle". An adversarial cycle is a set of $n\geq 1$ consecutive adversarial

Algorithm 1 Optimal selfish proposing strategy in full-predictable LC-PoS environments for attacker with communication capability $\eta = 0$

```
Input: $Seq$            ▷ $Seq$ is a list of -1 and 1 that represents the proposing
                          sequence within the range $(t_i, t_{i+1})$, where $t_i$ and $t_{i+1}$ are
                          two consecutive checkpoint slots. The adversarial (honest)
                          slot is denoted by 1 (-1).
Output: $leastCycle$    ▷ $leastCycle$ is a list that contains the adversarial slots that
                          serve as the starting points of the adversarial cycles within
                          the range $(t_i, t_{i+1})$. The number of adversarial slots in
                          $leastCycle$ is equal to the number of adversarial cycles
                          within $Seq$.
$indx \leftarrow 1$
for $t \leftarrow 1$ to len($Seq$) do
    $H[t] \leftarrow H[t-1] + Seq[t]$    ▷ $H[t]$ represents the height of a block that is proposed at time slot $t$.
    if $Seq[t] == 1$ then
        $aSlots[indx] = t$    ▷ $aSlots$ is a list of adversarial time slots.
        $indx \leftarrow indx + 1$
    end if
end for
$N \leftarrow$ len($aSlots$)    ▷ $N$ represents the number of adversarial slots within the range $(t_i, t_{i+1})$.
$Chain \leftarrow$ an empty list    ▷ $Chain$ contains multiple lists, where each list represents a specific combination of the adversarial slots that serve as the starting points of the adversarial cycles within the range $(t_i, t_{i+1})$.
for $indx \leftarrow 1$ to $N$ do
    $Chain_{indx} \leftarrow$ an empty list    ▷ $Chain_{indx}$ contains multiple lists, where each list represents a specific combination of the adversarial slots that serve as the starting points of the adversarial cycles before time slot $aSlots[indx] \in (t_i, t_{i+1})$.
end for
for $indx \leftarrow 1$ to $N - 1$ do
    if $indx == 1$ or $Chain_{indx} \neq \emptyset$ then
        $leastCycle_{indx} \leftarrow$ the list with the least number of adversarial cycles in $Chain_{indx}$
        $newLeastCycle_{indx} \leftarrow$ concatenate($leastCycle_{indx}, aSlots[indx]$)
        for $indx' \leftarrow indx$ to $N - 1$ do
            if $H[aSlots[indx]] \leq H[aSlots[indx']]$ and $H[aSlots[indx]] > H[aSlots[indx' + 1]]$ then
                $Chain[indx' + 1]$.append($newLeastCycle_{indx}$)
            end if
        end for
        if $H[aSlots[indx]] \leq H[aSlots[N]]$ then
            $Chain$.append($newLeastCycle_{indx}$)
        end if
    end if
end for
if $Chain_N \neq \emptyset$ then
    $leastCycle_N \leftarrow$ the list with the least number of adversarial cycles in $Chain_N$
    $newLeastCycle_N \leftarrow$ concatenate($leastCycle_N, aSlots[N]$)
    $Chain$.append($newLeastCycle_N$)
end if
$leastCycle \leftarrow$ the list with the least number of adversarial cycles in $Chain$
```

blocks in the main chain sandwiched by two honest main-chain blocks. Each pair of two consecutive adversarial cycles in the main chain are disconnected by one or more honest blocks. If assuming that the attacker's communication capability is equal to zero, each adversarial cycle comprising n adversarial blocks can orphan at most $n-1$ honest blocks. The number of honest blocks added to the main chain during the selfish proposing attack depends on the number of adversarial cycles. Assume that c denotes the number of adversarial cycles within the interval (t_i, t_{i+1}). In this case, the number of honest blocks added to the main chain within the interval (t_i, t_{i+1}) is equal to $N^{\mathcal{H}} - N^{\mathcal{A}} + c$. An optimal strategy minimizes the number of main-chain honest blocks. Therefore, to obtain the optimal strategy, we should find a strategy that minimizes the number of adversarial cycles. The method presented in Algorithm 1 brute-forces all the possible combinations of adversarial cycles between two checkpoint slots to find a combination with the lowest number of adversarial cycles.

Assume $\pi_i^{\texttt{0-SP}}$ is the optimal strategy corresponding to interval (t_i, t_{i+1}), which is obtained from Algorithm 1. Assume further that the number of orphaned honest blocks within interval (t_i, t_{i+1}) under strategy $\pi_i^{\texttt{0-SP}}$ is equal to $N^{\mathcal{A}} - c$, where c represent the minimum possible number of adversarial cycles within interval (t_i, t_{i+1}). In this part, we discuss knowing about the proposing sequences in intervals $[0, t_i]$ and $[t_{i+1}, \infty)$ does not change the optimal strategy $\pi_i^{\texttt{0-SP}}$ within interval (t_i, t_{i+1}). To prove this claim we need to show that the adversarial blocks within interval (t_i, t_{i+1}) cannot orphan more than $N^{\mathcal{A}} - c$ honest blocks within and outside interval (t_i, t_{i+1}). Assume that there is an honest block $B^{\mathcal{H}}$ proposed at t', where $t' \notin (t_i, t_{i+1})$. We want to prove that either it is infeasible for the attacker to orphan block $B^{\mathcal{H}}$ using his adversarial blocks within interval (t_i, t_{i+1}), or if he does succeed in orphaning block $B^{\mathcal{H}}$ using his adversarial blocks within interval (t_i, t_{i+1}), the number of orphaned honest blocks will not increase. Consider the scenario that $t' \in [t_{i+1}, \infty)$. If adversarial blocks within interval (t_i, t_{i+1}) can orphan block $B^{\mathcal{H}}$, they can also orphan the honest block proposed at checkpoint time slot t_{i+1}, which is contradictory to the definition of checkpoint slots. Now consider the scenario that $t' \in [0, t_i]$. In this case, block $B^{\mathcal{H}}$ may get orphaned by the adversarial blocks within interval (t_i, t_{i+1}). Assume the attacker orphans block $B^{\mathcal{H}}$ by an adversarial fork denoted by f, where f contains at least one adversarial block within interval (t_i, t_{i+1}). We claim that if f contains one adversarial block within interval (t_i, t_{i+1}), then all the other adversarial blocks in fork f also belong to interval (t_i, t_{i+1}). To justify this claim, assume fork f contains one adversarial block within interval $[0, t_i]$. In this case, an adversarial fork starting at a time slot earlier than t_i and ending at a time slot later than t_i can orphan an honest block proposed at $t' \leq t_i$, which is contradictory to the assumption that t_i is a checkpoint slot. Knowing that f only contains adversarial blocks within interval (t_i, t_{i+1}), we can prove that modifying strategy $\pi_i^{\texttt{0-SP}}$ to generate fork f cannot result in a greater number of orphaned honest blocks. By following strategy $\pi_i^{\texttt{0-SP}}$, the minimum number of adversarial cycles within interval (t_i, t_{i+1}) is equal to c. Orphaning the honest block proposed at $t' \leq t_i$ using $N^{\mathcal{A}}$ adversarial blocks within interval

(t_i, t_{i+1}) leads to the outcome where the number of adversarial cycles either stay the same or becomes greater than c. This is because the interval $[t', t_{i+1})$ contains a greater number of honest blocks compared to interval (t_i, t_{i+1}), while the number of adversarial blocks is the same in both intervals. Therefore, orphaning honest blocks outside interval (t_i, t_{i+1}) using the adversarial blocks within interval (t_i, t_{i+1}) cannot lead to a greater block ratio.

I Deep Q-Learning

I.1 Implementation details

In our implementation, the target and Q neural networks are structured with 3 hidden layers, comprising 128, 128, and 64 neurons each. We assigned $k_1 = 5$, indicating that each state stores information regarding the current fork as well as the possible forks that can be generated on top of the 5 most recent honest blocks. Therefore, the total number of actions and neurons in the output layer is equal to 18. In the full-predictable implementation, we set $k_2 = 54$, indicating that the attacker can predict the block proposers of upcoming 54 slots. In the semi-predictable implementation, we set $k_3 = 54$, indicating that the attacker is aware of the time slot for up to 54 future adversarial blocks. Therefore, in LC-PoS implementations with perfect randomness, full predictability, and semi-predictability, the number of neurons in the input layer is 10, 64, and 64, respectively. In semi-predictable LC-PoS implementation, we set the average block generation rate to be equal to 1 block per 20 slots. We set that if during the training process, $l_{\mathcal{H}}$ becomes greater than $l_{\mathcal{A}} + 5$, the action "`wait` in the current fork" gets removed from the possible action set. This is to speed up the training process. A greater bound may result in a more profitable strategy; however, it reduces the training speed. The results shown in Figures 4a and 4b are obtained by taking the average of the block ratio achieved over 1,000,000 steps of state transitions under the trained neural networks.

I.2 DQL implementation considering node locations

To obtain the optimal selfish proposing strategy, a proposer should be aware of his stake share and communication capability. In PoS protocols, it is almost straightforward for the proposers to calculate their stake share using account information available in the blockchain ledger. This shows that proposers have a good understanding of their stake shares. However, the concept of communication capability is a bit unclear since it is not straightforward to specify the number of nodes that receive the selfish proposer's block first during a block race. To move towards a more practical implementation, rather than passing communication capability as an input parameter to the environment, we can implement an environment that encompasses the location of proposer nodes.

We implement the network according to the model introduced in [14]. We assume that the proposer nodes are distributed according to the Poisson point

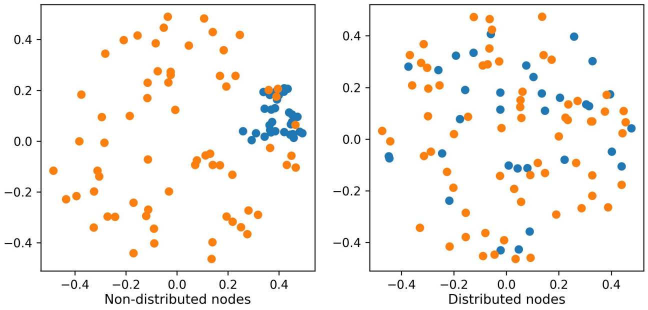


Fig. 16: Node locations

process within a spatial disk. Let $d_{i,j}$ and $cd_{i,j}$ denote the distance and the communication delay between proposer nodes P_i and P_j in the spatial disk. We assume further that the communication delay between proposer nodes P_i and P_j follows a normal distribution with mean $\mu = k \cdot d_{i,j}$, which is proportional to the distance between the nodes P_i and P_j, and a constant variance σ^2. Let $P^{\mathcal{A}}$ denote the set of adversarial nodes. Consider the scenario in which honest proposer $P_i^{\mathcal{H}}$ and the attacker simultaneously publish two new blocks to the network. Assume honest proposer $P_j^{\mathcal{H}}$ is the proposer of the upcoming time slot. The attacker wins the block race if there exists at least one adversarial proposer $P_k^{\mathcal{A}} \in P^{\mathcal{A}}$ that can satisfy the following condition:

$$cd_{i,k} + cd_{k,j} \leq cd_{i,j} \ \ . \tag{65}$$

The condition represented above implies that the proposer of the next slot receives the adversarial block earlier than the honest block and, consequently, proposes his new block on top of the adversarial block. We trained the DQL agent for an attacker with stake share $\alpha = \frac{1}{3}$ under two different network conditions depicted in Figure 16: the distributed model and the non-distributed model. In the distributed model, the adversarial nodes are well-distributed among the honest nodes. Conversely, in the non-distributed model, the adversarial nodes are almost highly concentrated. In our implementation, we assigned $k = 1$ and $\sigma = \frac{1}{25}$. The block ratios achieved under the distributed model and the non-distributed model are equal to 0.4466 and 0.4051, respectively. This demonstrates that a high degree of node distribution can lead to an increase in the attacker's block ratio.