Proof-of-Stake Mining Games with Perfect Randomness

MATHEUS V. X. FERREIRA, Princeton University, USA
S. MATTHEW WEINBERG*, Princeton University, USA

Proof-of-Stake blockchains based on a longest-chain consensus protocol are an attractive energy-friendly alternative to the Proof-of-Work paradigm. However, formal barriers to "getting the incentives right" were recently discovered, driven by the desire to use the blockchain itself as a source of pseudorandomness [4].

We consider instead a longest-chain Proof-of-Stake protocol with perfect, trusted, external randomness (e.g. a randomness beacon). We produce two main results.

First, we show that a strategic miner can strictly outperform an honest miner with just 32.8% of the total stake. Note that a miner of this size *cannot* outperform an honest miner in the Proof-of-Work model [21]. This establishes that even with access to a perfect randomness beacon, incentives in Proof-of-Work and Proof-of-Stake longest-chain protocols are fundamentally different.

Second, we prove that a strategic miner cannot outperform an honest miner with 30.8% of the total stake. This means that, while not quite as secure as the Proof-of-Work regime, desirable incentive properties of Proof-of-Work longest-chain protocols can be approximately recovered via Proof-of-Stake with a perfect randomness beacon.

The space of possible strategies in a Proof-of-Stake mining game is *significantly* richer than in a Proof-of-Work game. Our main technical contribution is a characterization of potentially optimal strategies for a strategic miner, and in particular a proof that the corresponding infinite-state MDP admits an optimal strategy that is positive recurrent.

CCS Concepts: • **Theory of computation → Algorithmic game theory**; • **Information systems → Digital cash**; • **Security and privacy**;

Additional Key Words and Phrases: cryptocurrency; proof-of-stake blockchains; energy-efficiency; random beacons; Nash equilibrium

ACM Reference Format:
Matheus V. X. Ferreira and S. Matthew Weinberg. 2021. Proof-of-Stake Mining Games with Perfect Randomness. In *Proceedings of the 22nd ACM Conference on Economics and Computation (EC '21), July 18–23, 2021, Budapest, Hungary.* ACM, New York, NY, USA, 21 pages. https://doi.org/10.1145/3465456.3467636

1 INTRODUCTION

Blockchains have been a resounding success as a disruptive technology. However, the most successful implementations (including Bitcoin [18] and Ethereum [22]) are built on a concept called proof-of-work. That is, participants in the protocol are selected to update the blockchain proportionally to their computational power. The consensus protocols underlying Bitcoin and Ethereum (and many other proof-of-work cryptocurrencies) have been secure in practice, and robust against strategic manipulation. There is even a theoretical foundation supporting this latter property: honestly following the Bitcoin protocol is a Nash equilibrium in a stylized model when no miner

*Supported by NSF CAREER Award CCF-1942497.

EC '21, July 18–23, 2021, Budapest, Hungary

ACM ISBN 978-1-4503-8554-1/21/07…$15.00
https://doi.org/10.1145/3465456.3467636

controls more than $\alpha^{\text{PoW}} \approx 0.329$ of the total computational power in the network [15, 21] (we will use the notation α^{model} to denote the supremum α such that whenever no miner is selected to create the next block with probability bigger than α, it is a Nash equilibrium for all miners to follow the longest-chain protocol in the referenced model).[1]

However, one major drawback of proof-of-work blockchains is their massive energy consumption. For example, Bitcoin currently consumes more electricity than all but 26 countries annually.[2] The need for specialized hardware and low-cost electricity/cooling/etc. also leads to concentration of the mining process among the few entities who have access to the necessary technology [1]. One popular emerging alternative is a paradigm termed proof-of-stake, where participants are selected proportionally to their stake in the currency itself.

Proof-of-stake cryptocurrencies do not suffer from this drawback, but raise new technical challenges, especially from the incentives perspective. Indeed, Brown-Cohen et al. [4] identifies several formal barriers to designing incentive compatible longest-chain proof-of-stake cryptocurrencies (that is, proof-of-stake protocols "like Bitcoin"). Their work highlights one key barrier: in existing proof-of-stake protocols, *the blockchain itself serves as a source of pseudorandomness*, whereas in proof-of-work protocols *the pseudorandom selection of participants is completely independent of the blockchain*. Specifically, they pose a stylized model with *No External Randomness* and show that it is *never* a Nash equilibrium for all miners to honestly follow the longest-chain protocol no matter how small they are (that is, $\alpha^{\text{PoSNER}} = 0$).

In this work, we investigate the incentive compatibility of longest-chain proof-of-stake protocols with access to *perfect external randomness, completely independent of the blockchain*, often termed a randomness beacon [20] (for brevity of notation, we'll refer to this model simply as PoS). We provide two main results, which give a fairly complete picture:

- We establish that $\alpha^{\text{PoS}} \leq 0.327 < \alpha^{\text{PoW}}$. That is, *even with access to perfect external randomness, longest-chain proof-of-stake protocols admit richer strategic manipulation than their proof-of-work counterparts*. We do this by designing a new strategic deviation that we term nothing-at-stake selfish mining, and establish that it is strictly more profitable than honest behavior for any miner with $\gtrapprox 0.327$ of the total stake (Theorem 3.4).
- We prove that $\alpha^{\text{PoS}} \gtrapprox 0.308$ (Theorem 6.1). In particular, this means that access to a randomness beacon fundamentally changes longest-chain proof of stake protocols: without one $\alpha^{\text{PoSNER}} = 0$, and any miner can profit by deviating. With a randomness beacon, the incentives are (quantitatively) almost as good as proof-of-work.

We now provide a high-level overview of our model (a significantly more detailed description of the model appears in Section 2), a brief overview of the key technical highlights, and an overview of related work.

1.1 Brief Overview of Model

Seminal work of Eyal and Sirer poses an elegant abstraction of the Bitcoin protocol (that we call the PoW model) [9]. Specifically, the game proceeds in infinitely many discrete rounds. In each round, a single miner is chosen proportionally to their computational power, and creates a block. Immediately upon creating a block, the miner must choose its contents (including its predecessor in the blockchain). The strategic decisions a miner makes are: a) which predecessor to select when they create a block, and b) when to publish that block to the other miners. The fact that predecessors must be chosen *upon creation* of the block captures that the contents of a block created via proof-of-work are fixed upon creation.

[1] Kiayias et al. [15] proves that $\alpha^{\text{PoW}} \gtrapprox 0.308$, and [21] estimates α^{PoW} to high precision as ≈ 0.329.
[2] Source: https://cbeci.org/. Accessed 3/25/2021.

A key concept in Bitcoin is the *longest-chain protocol*. Specifically, the longest chain is the published block with the most ancestors.[3] Each miner's reward is equal to the fraction of blocks they produce in the longest chain (taking a limit as rounds go to ∞). A miner honestly follows the longest-chain protocol if: a) they always select the (current) longest chain as the predecessor of any created node, and b) they publish all created blocks during the round in which it's created. Eyal and Sirer [9] establishes that $\alpha^{\text{PoW}} \leq 1/3$ (previously, it was believed that $\alpha^{\text{PoW}} = 1/2$ as proposed in [18]), and follow-up work further nailed down $\alpha^{\text{PoW}} \approx 0.329$ [21].

Brown-Cohen et al. [4] modify this model to capture proof-of-stake with no external randomness. In their model, a random coin is selected in each round *independently for each block*. That is, for each round, and each block B, an independent random miner is selected who is eligible to create a block with B as a predecessor with probability equals to their fraction of all the coins in the system. This captures that the protocol must use the chain itself as a source of pseudorandomness. Their work establishes that $\alpha^{\text{PoSNER}} = 0$: it is never a Nash equilibrium to honestly follow the longest-chain protocol in their model. This result is entirely driven by the fact that the protocol has no external randomness, and therefore, miners can make non-trivial predictions about future pseudorandomness.

Our model lies between these two, and captures proof-of-stake with perfect external randomness. Specifically, in each round a single coin is chosen to create a block. Thus a single miner is chose to create a block (just as in PoW) with probability proportional to their fraction of the coins in the system. The strategic decisions are now better phrased as: a) when to publish a created block, and b) which predecessor to select *when publishing*. Our model captures the following: perfect external randomness allows the protocol to select a random miner independently of all previous selected miners and all previously published blocks. The distinction to proof-of-work is that it is now computationally tractable to set the contents of the block, including its predecessor, at any point before it is published.

Note that our model does stipulate that the winner of round t can publish a single block with timestamp t. In a proof-of-stake protocol, there is no technical barrier to creating and publishing any number of blocks using the same timestamp (indeed, this is precisely because it is computationally efficient to produce blocks in proof-of-stake). However, it will be immediately obvious to the rest of the network that a miner has deviated from the longest-chain protocol in this specific way, and it will be immediately obvious which miner cheated.[4] A common solution to strongly disincentivize such behavior is a *slashing protocol*: any miner can include pointers to two blocks created using the same timestamp and the cheating miner will be steeply fined. While we will not rigorously model the incentives induced by a slashing protocol, our model implicitly assumes a sufficiently strong disincentive for miners to publish multiple blocks (and capture this in our model by simply hard-coding that miners must publish at most a single block with each timestamp).

To get intuition for the types of protocols our stylized model aims to capture, below is a sample (simplified) protocol to have in mind:[5]

[3] An ancestor is any block that can be reached by following a path of predecessors. Observe that because each block has a single predecessor, there is a single path of predecessors out of any block.

[4] Observe that deviations from the longest-chain protocol that select strategic predecessors or publish at strategic times cannot be definitively attributed to a cheating miner, as these deviations have an innocent explanation: latency. That is, perhaps the reason a miner chose the wrong predecessor is because news of the true longest chain had not yet reached them. Alternatively, perhaps the miner tried to publish their block during the correct round, but it only propagated through the network several rounds later due. Like all prior work, we do not rigorously model latency, and stick to the elegant model proposed in [9].

[5] We are not claiming that this protocol is secure in a rich model, nor will we reason formally about properties of the proposed slashing mechanism. We provide this just to give intuition for why our stylized model captures the salient features of a longest-chain protocol with trusted external randomness.

- In order to be eligible to mine, a coin must be frozen for (large) T rounds, along with a deposit equal to (large) L times its value (that is, the coin and its deposit must be owned by the same miner for T rounds in a row).
- After being used for mining, a coin and its deposit must be frozen for T rounds.
- During each round t, the randomness beacon outputs a random number. This is mapped to a random eligible coin, and its owner is the selected miner at round t. The selected miner can create blocks with timestamp t.
- If a miner ever publishes distinct blocks with the same timestamp, any other miner can include pointers to those two blocks in a block of their own. This will cause the deviant miner to lose their entire deposit (if desired, a $1 - \varepsilon$ fraction of it can be destroyed, and an ε fraction can be awarded to the altruistic miner).

Importantly, we are claiming neither that randomness beacons exist (either in theory or in practice), nor that slashing protocols that perfectly disincentivize detectable cheating (without affecting any other incentives) exist. Theorem 3.4 shows that *even if* these primitives existed, a longest-chain proof-of-stake protocol assuming them would still be (slightly) more vulnerable to strategic manipulation than a proof-of-work protocol. On the other hand, Theorem 6.1 establishes in some sense a reduction from proof-of-stake protocols that nearly match the incentive guarantees of proof-of-work protocols to the design of randomness beacons and slashing protocols.

1.2 Brief Technical Overview

Theorem 3.4 ($\alpha^{\text{PoS}} \lessapprox 0.327$) follows by designing our nothing-at-stake selfish mining strategy, and analyzing its expected payoff. While the insights to design our strategy are novel, the analysis is similar to those used in prior work to analyze the payoff of the resulting Markov Decision Process (MDP). We defer to Section 3 a description of our strategy and intuition for why it succeeds.

The proof of Theorem 6.1 is the bulk of our technical work. To start, we observe that our model admits an infinite-state MDP (just as in [21]). However, the space of strategies available to a miner in our setting is *significantly* richer than in the PoW model. We provide several examples demonstrating why counterintuitive behavior (such as orphaning one's own blocks) could a priori be part of an optimal strategy. So our main technical results characterize possible optimal strategies for this infinite-state MDP, culminating in a strong enough characterization to lower bound the optimal payoff for Theorem 6.1 and concluding $\alpha^{\text{PoS}} \geq 0.308$.

1.3 Related Work

The most related work is already overviewed above: Eyal and Sirer [9] provide the PoW model, develop the selfish mining attack, and prove that $\alpha^{\text{PoW}} \leq 1/3$. Sapirshtein et al. [21] estimates $\alpha^{\text{PoW}} \approx 0.329$ by solving the associated MDP to high precision, and Kiayias et al. [15] prove that $\alpha^{\text{PoW}} \gtrapprox 0.308$. Brown-Cohen et al. [4] study a related proof-of-stake model with no external randomness, and show that $\alpha^{\text{PoSNER}} = 0$. Other works also study similar questions in variants of this model (e.g. [5, 19]).

There is a rapidly-growing body of work at the intersection of mechanism design and cryptocurrencies [1, 6, 10, 13, 17]. Some of these works further motiviate the consideration of proof-of-stake cryptocurrencies [1], while others motivate the choice to restrict attention to Bitcoin's proportional reward scheme [6, 17], but there is otherwise little overlap between our works.

In practice, implementing a random beacon is a complex task [2, 3, 7] and is outside the scope of this paper. As previously noted, our results can be viewed either as a reduction to designing a randomness beacon (Theorem 6.1), or an impossibility result even under the assumption of a randomness beacon (Theorem 3.4).

Finally, it is worth noting that many existing proof-of-stake protocols fit the longest-chain paradigm [8, 12, 16], while others are fundamentally different [11]. Protocols based on Byzantine consensus are a growing alternative to the longest-chain paradigm, although both paradigms are well-represented in theory and in practice. Byzantine consensus protocols are outside the scope of our analysis.

1.4 Roadmap

Section 2 provides a very detailed description of our model, along with examples to help illustrate its distinction from proof-of-work. Section 3 provides our nothing-at-stake selfish mining, and Theorem 3.4. Sections 5 and 4 narrow the space of optimal strategies through a series of reductions. Section 6 overviews Theorem 6.1. Various helpful examples and all omitted proofs are in the appendix.

2 MODEL

A mining protocol is a Nash equilibrium if no miner wishes to unilaterally change their strategy provided all miners are following the intended protocol. Thus it suffices to consider a two-player game between Miner 1 and Miner 2. Think of Miner 2 as the "rest of the network", which is honestly executing the longest-chain protocol, and think of Miner 1 as the "potential attacker" that optimizes his strategy provided Miner 2 is honest. Following [9, 15, 21] and subsequent works, the game proceeds in discrete time steps (abstracting away the exponential rate at which blocks are found) which we call *rounds*, and the rounds are indexed by $\mathbb{N}_+$. The state B of the game is a tuple $(\textsc{Tree}(B), \mathcal{U}_1(B), \mathcal{U}_2(B), T_1(B), T_2(B))$ (each of these terms will be explained subsequently).

Rounds. During every round n, a single miner creates a new block, and we denote that miner by $\gamma_n \in \{1, 2\}$. We denote by $\gamma := \langle \gamma_n \rangle_{n \in \mathbb{N}_+}$ the full ordered list of miners for each round. In an execution of the game, each γ_n is drawn i.i.d., and equal to 1 with probability $\alpha < 1/2$. We let $T_i := \{n \mid \gamma_n = i\}$ as the rounds during which Miner i creates a new block. $T_i(B)$ denotes all blocks created by Miner i at state B. We abuse notation and might refer to n as the state at round n (after block n is created and all actions are taken, but before round $n + 1$ starts). That is, $(\textsc{Tree}(n), \mathcal{U}_1(n), \mathcal{U}_2(n), T_1(n), T_2(n))$ is the state at round n.

Blocks. The second basic element is a *block*. Each block has a label in $\mathbb{N}$. Blocks are totally ordered by their labels and we say block s was created before block v if $s < v$. We overload notation and also use n to refer to the block produced in round n. All blocks are initially *unpublished*, and can later become *published* due to actions of the miners. Once a block n is published, it has a pointer to exactly one predecessor n' created earlier (that is $n' < n$) and we write $n \to n'$.

Block Tree. Because all published blocks have a pointer to an earlier block, this induces, at all rounds, a block tree $\textsc{Tree}$. We will also refer to V, E as the nodes and edges in $\textsc{Tree} = (\mathrm{V}, \mathrm{E})$. Here, the nodes are all blocks which have been published. Every node has exactly one outgoing (directed) edge towards its predecessor. Before the game begins, the block tree contains only block 0, which we refer to as the *genesis block* and not created by Miner 1 nor Miner 2. We let $\mathcal{U}_i$ denote the set of blocks which have been created by Miner i, but are not yet published. We refer to

$$B_0 := ((\{0\}, \emptyset), \emptyset, \emptyset, \emptyset, \emptyset) \tag{2.1}$$

as the initial state before any blocks are created and the block tree contains only the genesis block.

Ancestor Blocks. We say that block a is an *ancestor* of block $b \in V$ if there is a directed path from b to a (so b is an ancestor of itself). We write $A(b)$ to denote the set of all ancestors of b (observe that a block can never gain new ancestors, so this is well-defined without referencing the particular

state S, or the round, etc.). We use $h(b) := |A(b)| - 1$ as short-hand for the height of block b (the genesis block is the only block with height 0).

Longest Chain. The *longest chain* $C := \arg\max_{b \in V}\{h(b)\}$ is the leaf in V with the longest path to the genesis block, breaking ties in favor of the first block published, and then in favor of the earliest-indexed block. We use H_i to refer to the block in $v \in A(C)$ with height i and we refer to $A(C)$ as the *longest path*. We say a block q is *forked* (or orphaned) when $q \in A(C)$, but a new block C' becomes the longest chain and $q \notin A(C')$.

Successor blocks. We say block a is a *successor* of block $b \in V$ if $a \neq b$ is in the unique path from C to b. We write $\textsc{Succ}(b)$ to denote the set of successors of b.

Actions. During round n, Miner i knows γ_ℓ for all $\ell \leq n$, and can take the following actions:

(1) Wait: wait for the next round, and do nothing.
(2) PublishSet(V', E'): publish a set of blocks V' with pointers E'. This adds V' to V, E' to E, and changes all blocks in V' from unpublished to published. To be valid, it must be that:
   - $V' \subseteq \mathcal{U}_i$ (Miner i actually has blocks V' to publish).
   - For all $v \to v' \in E'$, $v \in V'$, $v' \in V \cup V'$ (syntax check for edges in E').
   - For all $v \to v' \in E'$, $v > v'$ (pointers are to earlier blocks).
   - For all $v \in V'$, there is exactly one outgoing edge in E' (every block has exactly one pointer).

Clarifying Order of Operations. At the beginning of round n, there is a block tree $\textsc{Tree} = \textsc{Tree}(n-1)$, and each miner i has a set of unpublished blocks $\mathcal{U}_i = \mathcal{U}_i(n-1)$. Then:

(1) γ_n is drawn, and equal to 1 with probability α, and 2 with probability $1-\alpha$. This updates $\mathcal{U}_{\gamma_n} := \mathcal{U}_{\gamma_n}(n-1) \cup \{n\}$. For the other miner, $\mathcal{U}_{3-\gamma_n} := \mathcal{U}_{3-\gamma_n}(n-1)$.
(2) Miner 2 takes an action. If that action is PublishSet(V', E'), add the nodes V' and edges E' to $\textsc{Tree}$, and update $\mathcal{U}_2 := \mathcal{U}_2 \setminus V'$.
(3) Miner 1 takes an action. If that action is PublishSet(V', E'), add the nodes V' and edges E' to $\textsc{Tree}$, and update $\mathcal{U}_1 := \mathcal{U}_1 \setminus V'$.
(4) At this point, round n is over, so $\textsc{Tree}(n) := \textsc{Tree}$, $\mathcal{U}_i(n) := \mathcal{U}_i$, etc.

Predecessor state. For state B, we define $B^{\textsc{Half}}$ as the state prior to B before Miner 1 took their most recent action and after Miner 2 took their most recent action. Similarly, we define $(\textsc{Tree}^{\textsc{Half}}(n), \mathcal{U}_1^{\textsc{Half}}(n), \mathcal{U}_2^{\textsc{Half}}(n), C^{\textsc{Half}}(n))$ as the subsequent state to $(\textsc{Tree}(n-1), \mathcal{U}_1(n-1), \mathcal{U}_2(n-1), C(n-1))$ after block n was created, Miner 2 takes their action and before Miner 1 takes their action.

Recall that Miner 2 acts first in every round, so the second tie-breaker in deciding what is the longest chain is only used to distinguish between two blocks of Miner 1 published during the same round, and will never be invoked in a "reasonable" strategy for Miner 1 (see Observation 4.3).[6] Like Kiayias et al. [15], we use FRONTIER to refer to the honest strategy, which never forks and it will be Miner 2's strategy.

Definition 2.1 (Frontier Strategy). *During all rounds n, the FRONTIER strategy for miner i does the following:*

- *If $\gamma_n \neq i$, Wait.*
- *If $\gamma_n = i$, PublishSet($\{n\}, \{n \to C\}$) (publishes the new block pointing to the longest chain).*

Definition 2.2 (Rewards). *For any two states B and B', define* Miner k's reward *as the integer-valued function r^k from state B to B' as the difference between the number of blocks created by Miner*

[6] Eyal and Sirer [9] also considers the case where Miner 1 wins a tie-breaking with probability β. In principle, our model can easily accommodate any $\beta \in [0, 1]$, but we focus on the case $\beta = 0$ since it is the most pessimistic for the attacker.

k in the longest path at state B' and B. That is,

$$r^k(B, B') := |A(C(B')) \cap T_k(B')| - |A(C(B)) \cap T_k(B)|. \tag{2.2}$$

Payoffs. As in [9] and follow-up work, miners receive steady-state revenue *proportional to their fraction of blocks in the longest chain.* Recall that in our notation, $C(n)$ denotes the longest chain after the conclusion of round n, $A(C(n))$ denotes the ancestors of $C(n)$, and T_i denotes all blocks created by Miner i. Therefore, we define the payoff to Miner 1, when Miner i uses strategy π_i as:

$$\textsc{Rev}_\gamma^{(n)}(\pi_1, \pi_2) := \frac{|A(C(n)) \cap T_1|}{h(C(n))}, \qquad \text{and} \qquad \textsc{Rev}(\pi_1, \pi_2) := \mathbb{E}_\gamma\left[\liminf_{n\to\infty} \textsc{Rev}_\gamma^{(n)}(\pi_1, \pi_2)\right],$$

where the expectation is taken over γ, recalling that each γ_n is i.i.d. and equals to 1 with probability α, and 2 with probability $1 - \alpha$.

We will study in particular the payoff of strategies π_1 against FRONTIER. For simplicity of notation later, we will refer to $\mathcal{U} := \mathcal{U}_1$ (because FRONTIER has no unpublished blocks, so $\mathcal{U}_2$ is unnecessary), $\pi := \pi_1$ (because $\pi_2 :=$ FRONTIER). We will also list all states only as $(\textsc{Tree}(n), \mathcal{U}(n), T_1(n))$ (because the other variables can be inferred from these, conditioned on $\pi_2 :=$ FRONTIER). We further define:

$$\textsc{Rev}_\gamma^{(n)}(\pi_1) := \textsc{Rev}_\gamma^{(n)}(\pi_1, \text{FRONTIER}), \quad \textsc{Rev}(\pi_1) := \textsc{Rev}(\pi_1, \text{FRONTIER}). \tag{2.3}$$

A strategy π^* is optimal if $\textsc{Rev}(\pi^*) = \max_\pi \textsc{Rev}(\pi)$. Thus FRONTIER is a *Nash equilibrium* if FRONTIER is an optimal strategy for Miner 1.

Proof-of-Work vs. Proof-of-Stake. Our model, as stated, captures Proof-of-Stake protocols with perfect/trusted external randomness. Importantly, this means that once a miner knows they created a block during round n, they do not need to decide precisely the contents of that block until they publish it (because there is no computational difficulty to produce a block). In Proof-of-Work and the model of Eyal and Sirer [9], the miner of block n must decide *in round n* the contents of that block (because the contents are locked in as soon as the miner succeeds in proving work). Crucially, the miner *must* decides the ancestor of block n during time $t = n$ while in our model, the miner decides the ancestor of block n any time $t \geq n$ before block n is published. This is the only difference between the two models. Observe that any Proof-of-Work strategy is also valid in our model: this would just be a strategy which chooses the pointer for block n in round n, and does not change it when publishing later. Example 2.3 helps illustrate this distinction.

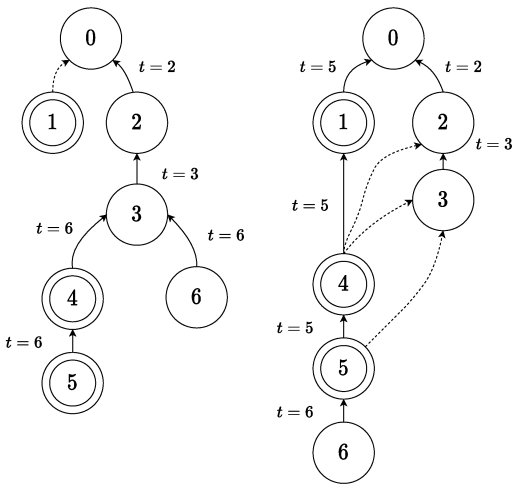

Fig. 1. Diagram representing the mining game in Example 2.3. We use double circles for blocks owned by Miner 1 and single circles for blocks owned by Miner 2. The genesis block – i.e., block 0 – is not owned by neither Miner 1 nor Miner 2. The time stamps near the solid edges represent the round where the edge was published, and the number inside the circle represents the round when the block was created. The dashed edges represent some edges that could have been created (edges from any nodes in $\{4, 5\}$ to any nodes in $\{0, 1, 2, 3\}$ are also feasible — i.e., any edge from a later node to an earlier node is feasible).

Example 2.3 (Proof-of-Work vs Proof-of-Stake). *Consider the following 6-round example where Miner 1 creates blocks 1, 4 and 5, and Miner 2 create blocks 2, 3 and 6 as depicted in Figure 1. If Miner 1 was following the Selfish Mining strategy [9] (Definition 3.1), they would decide to create and withhold* $1 \longrightarrow 0$ *(at time* $t = 1$*). Miner 2 would then publish* $3 \longrightarrow 2 \longrightarrow 0$*. At this point, Miner 1 would never attempt to publish* $1 \longrightarrow 0$ *and we say block 1 becomes permanently orphan. Next, Miner 1 creates and withhold* $4 \longrightarrow 3$ *(at* $t = 4$*) and* $5 \longrightarrow 4$ *(at* $t = 5$*). When Miner 2 publishes* $6 \longrightarrow 3$*, Miner 1 publishes* $5 \longrightarrow 4 \longrightarrow 3$ *(at* $t = 6$*),* forking *block* 6 *from the longest path. This nets 2 blocks in the longest path for Miner 1, and 2 for Miner 2.*

*Here is a viable strategy in the Proof-of-Stake mining game: Miner 1 still withholds block 1 (*but does not yet decide where it will point*), and it is still initially orphaned when Miner 2 publishes blocks 2 and 3. When Miner 1 creates block 4, they withhold it (*but does not yet decide where it will point*). When they create block 5, they decide to publish block 4 (*deciding only now to point to block 1*) and block 5 (*deciding only now to point to block 4*). This creates a new longest path. Miner 2 then publishes* $6 \longrightarrow 5$*. This nets 3 blocks in the longest path for Miner 1, and 1 for Miner 2.*

Importantly, observe that in the proof-of-work model, it would be exceptionally risky for Miner 1 to pre-emptively decide to point block 4 to block 1 at time $t = 4$ *without knowing that they will create block 5 (because maybe Miner 2 creates block 5, and then they would be an additional block behind). But in the proof-of-stake model, Miner 1 can wait to gather more information before deciding where to point. In particular, if they happened to instead create block 6 but not 5, they could have published* $6 \longrightarrow 4 \longrightarrow 3$*. In proof-of-stake, Miner 1 has the flexibility to make this decision later. In proof-of-work, they have to decide immediately whether to have block 4 pointing to 1 or 3.*

Reminder of Notation. Table **??** in Appendix **??** is a reminder of our notation.

2.1 Payoff as Fractional of Blocks in the Longest Path

Eyal and Sirer [9] motivates the use of the fraction of blocks as a miner's utility due to the difficulty adjustment in Bitcoin's PoW protocol: Bitcoin adjusts PoW difficult so that, on expectation, miners create one block every 10 minutes and the creator of each block receives new Bitcoins as block reward. Thus a miner maximizes their expected number of blocks in the longest path up to time T by maximizing their expected fraction of blocks in the longest path up to time T.

For PoS, a random beacon outputs a random string at a fixed rate, independent of the blockchain state.[7] Although difficult adjustment is absent in proof-of-stake, the probability of a miner creating the next block is proportional to α, their fraction of coins in the system. Although α is approximately constant over short time horizons, over long time horizons, α will depend on the fraction of block rewards Miner 1 collects. Thus, in the *long-term*, Miner 1 will maximize block rewards by maximizing their fraction of blocks in the longest path.

In Figure 2, we simulate the fraction of coins owned by Miner 1 overtime when Miner 1 follows FRONTIER or the Nothing-at-Stake Selfish Mining (NSM in Definition 3.3) and Miner 2 follows FRONTIER. From the simulation, we observe NSM allows Miner 1 to add a higher fraction of blocks in the longest path when compared with FRONTIER as long as Miner 1 owns more than 32.8% of the coins. We confirm this empirical result in Theorem 3.4. This allows *Miner 1 to eventually own an arbitrarily large fraction of the coins.* Thus the security of a longest chain proof-of-stake protocol depend on a formal guarantees that *no strategy is more profitable than FRONTIER* when a profit maximizing miner is maximizing their fraction of blocks in the longest path. We accomplish this task in Theorem 6.1.

[7] The National Institute of Standards and Technology (NIST) random beacon outputs 512 bits every 60 seconds [14].

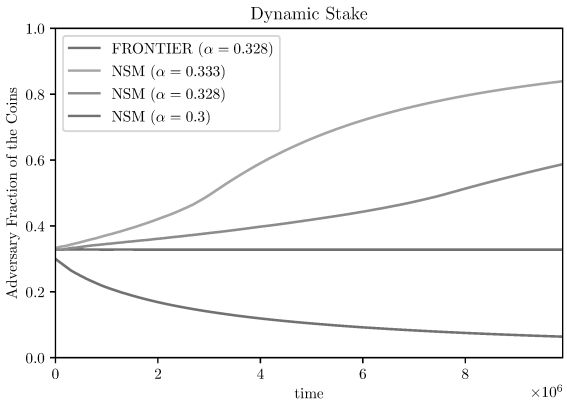

Fig. 2. Simulation of Miner 1's dynamic stake over one million rounds when Miner 1 either follow FRONTIER or the Nothing-at-Stake Selfish Mining (NSM) strategy for distinct initial values of α. As initial condition, the system has 100 thousand coins with Miner 1 initialing owning α fraction of the coins. We observe when following FRONTIER, Miner 1 cannot increase their fraction of the stake. We also observe NSM allows Miner 1 to increase their fraction of the stake when they initially own 32.8% of the coins, but NSM is not profitable when Miner 1 owns less than 32.77%, as we find in Theorem 3.4.

2.2 Capitulating a State

For some states of the game, Miner 1 might follow a strategy that will never fork some blocks from the block tree. Then, it is safe to say that Miner 1 deletes those blocks from the state (or treats the one with highest height as the new genesis block) and consider a trimmed version of the state variable. As example, define $B_{0,1}$ where Miner 2 creates and publishes block 1. Thus

$$B_{0,1} := (\{\{0,1\}, \{1 \to 0\}), \emptyset, \emptyset). \tag{2.4}$$

If Miner 1 never forks block 1, then it is safe to say that in the view of Miner 1 state $B_{0,1}$ is equivalent to state B_0 (after treating block 1 as the new genesis block). Then, we say Miner 1 capitulates from state $B_{0,1}$ to B_0. Since Miner 1 can induce the mining game to return to prior states, it is convenient to think of Miner 1 optimizing an underlying Markov Decision Process. Next, we provide a definition and formalize the payoff of the MDP in Appendix **??**.

Markov Decision Process. A *Markov Decision Process* (MDP) for the mining game where Miner 1 follows strategy π and Miner 2 follows FRONTIER is a sequence $(X_t)_{t\geq 0}$ where X_t is a random variable representing the state by the end of round t and before any actions have been take in round $t+1$. Unless otherwise stated, we initialize $X_0 = B_0$ (Equation 2.1). The game transitions from state X_t to X_{t+1} once the next block is created followed by Miner 2 taking their action followed by Miner 1 taking their action.

For a mining game $(X_t)_{t\geq 0}$ that starts at state $X_0 = B_0$, let

$$\tau := \min\{t \geq 1 : \text{State } X_t \text{ is equivalent to state } B_0 \text{ in the view of Miner 1}\} \tag{2.5}$$

be the first time step Miner 1 capitulates to state B_0. Similary, let τ' be the second time step Miner 1 capitulates to state B_0. Then, the sequences of rewards

$$r^k(X_0, X_1), r^k(X_1, X_2), \ldots, r^k(X_{\tau-1}, X_\tau) \qquad r^k(X_\tau, X_{\tau+1}), r^k(X_{\tau+1}, X_{\tau+2}), \ldots, X_{\tau'-1}, X_{\tau'})$$

are independent and identically distributed for $k = 1, 2$. One fundamental question is to understand if $\mathbb{E}[\tau] < \infty$ when Miner 1 is following an optimal strategy (that is, does Miner 1 capitulate to state B_0 at some point with probability 1?). In the proof-of-stake setting, this is not obviously true, and Example 2.3 gives some intuition why: while a proof-of-work Selfish Miner capitulates to state B_0 at round 3 (allowing Miner 2 to keep block 2 in the longest path), a Proof-of-Stake miner might prefer to wait for an opportunity to use block 1, and it is not a priori clear at what point it is safe to conclude that any optimal strategy would have given up on block 1 by now.

2.3 Recurrence

Definition 2.4 (Recurrence). *Consider a mining game starting at state $X_0 = B_0$ where Miner 1 follows strategy π. Let E be the event Miner 1 capitulates to state B_0 at some time $\tau < \infty$. We say π is*

- *Transient if $Pr[E] < 1$.*
- *Recurrent if $Pr[E] = 1$.*
- *Null recurrent if it is recurrent and $\mathbb{E}[\tau] = \infty$.*
- *Positive recurrent if it is recurrent and $\mathbb{E}[\tau] < \infty$.*

Observe Miner 1 never forks the longest chain when using FRONTIER. Thus Miner 1 capitulates to state B_0 at every time step and $\tau = 1$.

Observation 2.5. *FRONTIER is positive recurrent.*

For Proof-of-Work mining games, Kiayias et al. [15] and Sapirshtein et al. [21] assumes Miner 1 will always follow a positive recurrent strategy. To motivate this technical assumption, they assume Miner 1 will never fork a block published by himself, which is sensible because Miner 1 can only mine on a single branch of the blockchain. This is not the case for Proof-of-Stake blockchains, and it is not a priori clear that Miner 1 will never fork a block that they created themselves. To see why this may occur, consider the following example.

First, define $B_{k,0}$, for $k \geq 0$, as the state where Miner 1 creates and withhold blocks $\{1, 2, \ldots, k\} = [k]$. Thus

$$B_{k,0} := ((\{0\}, \emptyset), [k], [k]). \tag{2.6}$$

Then define $B_{1,1}$ as the state after $B_{1,0}$ where Miner 2 creates block 2 and publishes $2 \to 0$. Thus

$$B_{1,1} := ((\{0, 2\}, \{2 \to 0\}), \{1\}, \{1\}). \tag{2.7}$$

Example 2.6. *Consider a game at state $B_{1,1}$. After round 2, Miner 2 creates blocks $3, 4, \ldots, 9, 10, 12$ and publishes $12 \to 10 \to 9 \ldots \to 4 \to 3 \to 2$ and Miner 1 creates and withholds blocks $11, 13, 14, \ldots, 24$. At time step 13, Miner 1 publishes $13 \to 11 \to 10$ (this follows the classical selfish mining strategy: it gives up on block 1, but publishes $11 \to 10$ and $13 \to 11$ to fork block 12). This is reasonable, because it is unlikely that Miner 1 can add block 1 to longest path and Miner 1 risks losing blocks 11 and 13 if Miner 2 creates and publishes block 14. However, in the event Miner 1 is lucky and creates blocks $14, 15, \ldots, 24$, Miner 1 can fork all blocks from the current longest path (including his own blocks 11 and 13), resulting in a new longest path with blocks $1, 14, 15, \ldots, 24$ (consisting entirely of Miner 1's blocks). Indeed, upon creating block 14, Miner 1 need not immediately decide whether to make its predecessor 13, or whether to wait and see if they get an extremely lucky run to override the entire chain.*

Note that we are not claiming that this is the optimal decision for Miner 1 from this state, or even that an optimal strategy may ever find itself in this state.[8] However, this example helps demonstrate that a significantly richer space of strategies are potentially optimal in our model, as compared to proof-of-work.

This example shows that we must be careful to not exclude optimal strategies when claiming any restrictions on strategies considered by Miner 1. We will eventually address this by introducing the notion of a *checkpoint*, Section 5.1, and prove that there are *some* conditions that allow us to claim that any optimal strategy for Miner 1 will not fork a checkpoint (however, we do not prove that Miner 1 will never consider forking their own blocks).

[8] For example, if this were the optimal decision from this state, it would likely be because α is small, and Miner 1 should just take whatever opportunities they have to publish blocks. However, if α is small, that may mean it is better for Miner 1 to just be honest, and they would never find themselves in this situation. The point is that some quantitative comparison is necessary in order to determine whether the optimal strategy for Miner 1 would ever take this action.

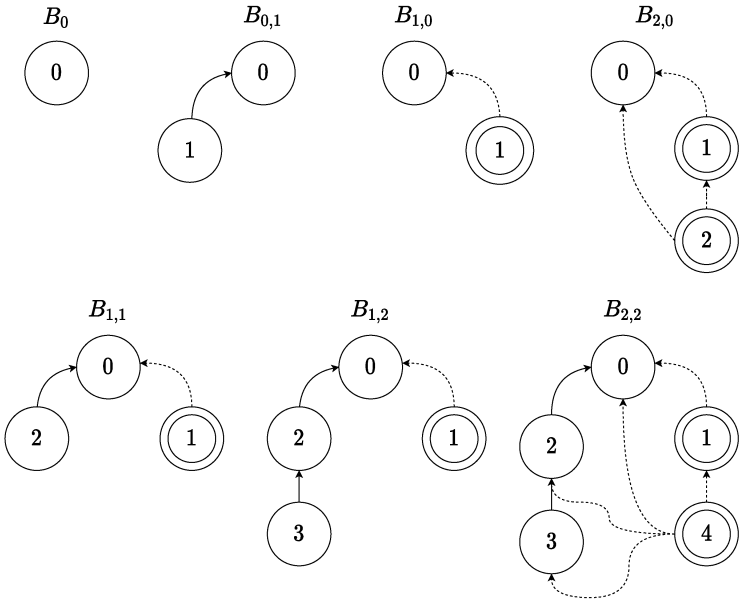

Fig. 3. Diagram representing states B_0, $B_{0,1}$, $B_{0,2}$, $B_{1,0}$, $B_{1,1}$, $B_{1,2}$ and $B_{2,2}$. Block 0 has no owner. All double circles are Miner 1's hidden blocks. All circles are Miner 2's published blocks. Dashed lines are edges Miner 1 can publish.

3 ENHANCING SELFISH MINING WITH NOTHING-AT-STAKE

In this section, we show explicitly an strategy that outperforms FRONTIER even when $\alpha = 0.3277$. From Sapirshtein et al. [21], FRONTIER is optimal for Proof-of-Work mining games when Miner 1 has mining power $\alpha \leq 0.329$ (i.e., $\alpha^{\text{PoW}} \approx 0.329$). Thus our strategy witnesses that Proof-of-Stake mining games admits strategies that are more profitable than any strategy in a Proof-of-Work mining game – that is, will establish $\alpha^{\text{PoS}} < \alpha^{\text{PoW}}$.

Our strategy will be a subtle modification from the Selfish Mining strategy of Eyal and Sirer [9] that leverages the Nothing-at-Stake vulnerability in Proof-of-Stake blockchains.[9]

Let's first define the states of interest for our strategy. Recall B_0 is the state where the block tree contains only the genesis block; $B_{1,0}$ is the state after Miner 1 creates and withholds block 1 (Equation 2.6); $B_{2,0}$ is the state after Miner 1 creates and withholds blocks 1 and 2 (Equation 2.6); $B_{0,1}$ is the state after Miner 2 publishes $1 \to 0$ (Equation 2.4); $B_{1,1}$ is the state after $B_{1,0}$ if Miner 2 publishes $2 \to 0$ (Equation 2.7). Additionally, define the following states.

- $B_{1,2}$ is the state after $B_{1,1}$ if Miner 2 creates block 3 and publishes $3 \to 2$:

$$B_{1,2} := (\{\{0, 2, 3\}, \{3 \to 2 \to 0\}\}), \{1\}, \{1\}). \tag{3.1}$$

- $B_{2,2}$ is the state after $B_{1,2}$ if Miner 1 creates and withhold block 4:

$$B_{2,2} := (\{\{0, 2, 3\}, \{3 \to 2 \to 0\}\}), \{1, 4\}, \{1, 4\}). \tag{3.2}$$

These and other relevant states are depicted in Figure 3.

Definition 3.1 (Selfish Mining [9]). *Let $(X_t)_{t \geq 0}$ be a mining game starting at state $X_0 = B_0$. Miner 1 uses the* Selfish Mining (SM) *strategy, Figure 4, which takes the following actions:*

- *Wait at states B_0 and $B_{1,0}$.*
- *At state $B_{0,1}$, capitulate to state B_0.*

[9]The nothing-at-stake vulnerability refers to the fact the algorithm for which a miner can verify a block validity is computationally efficient. Thus miners have no cost to choosing the block content (including its ancestor) at the moment the block is about to be published .

- *If $X_2 = B_{1,1}$ and Miner 1 creates block 3, publishes $3 \to 1 \to 0$, then capitulates to state B_0.*
- *If $X_2 = B_{1,1}$ and Miner 2 creates block 3 and publishes $3 \to 2$, then Miner 1 capitulates to state B_0.*
- *If $X_2 = B_{2,0}$, Miner 1 plays Wait until the first time step $\tau \geq 3$ where $|T_2(X_\tau)| = |T_1(X_\tau)| - 1$. At time step τ, Miner 1 publishes all of $\mathcal{U}(X_\tau) = T_1(X_\tau)$ pointing to 0 and forking $T_2(X_\tau)$, then capitulates to state B_0.*

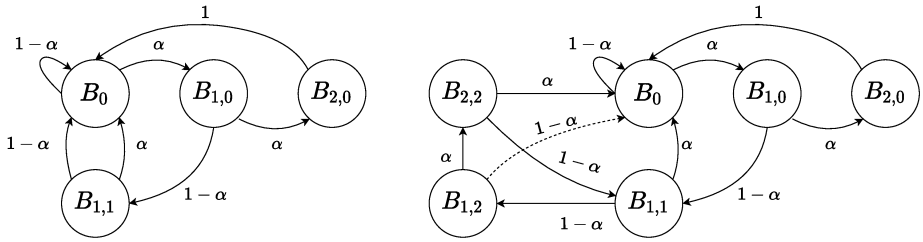

Fig. 4. Markov chain representing the Selfish Mining strategy (left) and the Nothing-at-Stake Selfish Mining strategy (right).

Theorem 3.2 (Equation 8 in [9]). *For the selfish mining strategy*

$$\textsc{Rev}(SM) = \frac{\alpha^2(4\alpha^2 - 9\alpha + 4)}{\alpha^3 - 2\alpha^2 - \alpha + 1}.$$

Moreover, $\textsc{Rev}(SM) > \alpha = \textsc{Rev}(FRONTIER)$ for $\alpha > 1/3$.

Our Nothing-at-Stake Selfish Mining is similar, with one key difference: In Selfish Mining, Miner 1 capitulates immediately after a loss (specifically, if Miner 2 creates block 3 and publishes $3 \to 2$ from state $B_{1,1}$, Miner 1 immediately accepts that block 1 is now permanently orphaned and capitulates to B_0). Nothing-at-Stake Selfish Mining instead remembers this orphaned block, and considers bringing it back later. Importantly, Nothing-at-Stake Selfish Mining can wait to see whether it finds many blocks (in which case it will try to publish the block 1) or not (in which case it will let block 1 remain orphaned) before deciding what to do. Below is a formal description.

Definition 3.3 (Nothing-at-Stake Selfish Mining). *Let $(X_t)_{t\geq 0}$ be a mining game starting at state $X_0 = B_0$. Miner 1 uses the* Nothing-at-Stake Selfish Mining (NSM) *strategy , right of Figure 4, which takes the following actions:*

- *Wait at states B_0, $B_{1,0}$ and $B_{1,2}$.*
- *At state $B_{0,1}$, capitulate to state B_0.*
- *If $X_t = B_{1,1}$ and Miner 1 creates block 3, publishes $3 \to 1 \to 0$, then capitulates to state B_0.*
- *If $X_t = B_{1,2}$ and Miner 2 creates block 4 and publishes $4 \to 3$, then Miner 1 capitulates to state B_0.*
- *If $X_t = B_{2,2}$ and Miner 1 creates block 5, publishes $5 \to 4 \to 1 \to 0$, then Miner 1 capitulates to state B_0.*
- *If $X_t = B_{2,2}$ and Miner 2 creates block 5 and publishes $5 \to 3$, Miner 1 capitulates to state $B_{1,1}$. That is, Miner 1 allows Miner 2 to walk away with blocks 2 and 3 and forgets about unpublished block 1, but remembers unpublished block 4 in the hope of forking block 5 in the future. The resulting state is equivalent to $B_{1,1}$ since we can relabel block 3 as 0, 4 as 1 and 5 as 2.*
- *If $X_2 = B_{2,0}$, Miner 1 plays Wait until the first time step $\tau \geq 3$ where $|T_2(X_\tau)| = |T_1(X_\tau)| - 1$. At time step τ, Miner 1 publishes all of $\mathcal{U}(X_\tau) = T_1(X_\tau)$ pointing to 0 forking blocks $T_2(X_\tau)$, then Miner 1 capitulates to state B_0.*

Let's quickly understand why this strategy is not possible in the proof-of-work model. Zero in on Miner 1's behavior at $B_{2,2}$. If Miner 1 creates block 5, Miner 1 publishes blocks 1, 4, 5, and in particular has their block 4 point to block 1. However, if Miner 2 creates block 5, Miner 1 capitulates to state $B_{1,1}$. From here, if Miner 1 creates block 6, they immediately publish 4 and 6, *having block* 4 *point to block* 3.

That is, while using this strategy, Miner 1 does not decide where block 4 will point upon mining it, but only upon publishing it. In a Proof-of-Work blockchain, Miner 1 must commit to the ancestor of block 4 at time step 4, so this strategy cannot be used. Intuitively, a nothing-at-stake selfish miner remembers an orphaned block to see if they might get lucky in the future. Importantly, in the PoS model they can *still* wait to decide whether to try and bring this block into the longest chain *even after finding their next block* (but before deciding where to publish it). This extra power enables not only a slight improvement over standard selfish mining but also a strategy the is strictly better than any other valid strategies for the Proof-of-Work mining game.

Theorem 3.4. *For the nothing-at-stake selfish mining strategy,*

$$\textsc{Rev}(NSM) = \frac{\alpha^2(3\alpha^7 - 13\alpha^6 + 18\alpha^5 - 4\alpha^4 - 12\alpha^3 + 15\alpha^2 - 12\alpha + 4)}{3\alpha^9 - 17\alpha^8 + 40\alpha^7 - 50\alpha^6 + 36\alpha^5 - 14\alpha^4 + \alpha^3 + \alpha^2 - 2\alpha + 1}.$$

Moreover $\textsc{Rev}(NSM) > \alpha$ *for* $\alpha > 0.3277$.

Recall Sapirshtein et al. [21] estimates $\alpha^{PoW} \approx 0.329$. Thus Theorem 3.4 implies

$$\alpha^{PoS} < 0.3277 < 0.329 \approx \alpha^{PoW}.$$

Interestingly, Nothing-at-Stake Selfish Mining is not better than Selfish Mining for all α. In Figure 6, by plotting the difference Rev(NSM) − Rev(SM) as a function of α, we observe Selfish Mining is better than Nothing-at-Stake Selfish Mining for $\alpha > 0.44$.

To get intuition why this happens, consider how SM and NSM differs in the event Miner 1 creates blocks 1, 4, 5 and Miner 2 creates blocks 2 and 3. By the end of the 5-th round, SM is at state $B_{2,0}$ while NSM just moved from state $B_{2,2}$ to B_0. The main intuition is that being at state $B_{2,0}$ is a highly profitable for Miner 1 when α is large. In the proof of Theorem 3.2, Appendix **??**, we show Miner 1 creates, on expectation, $\frac{\alpha}{1-2\alpha}$ blocks from the moment the game reaches state $B_{2,0}$ until the moment the game first returns to state B_0. Moreover, SM has a bigger probability of being at state $B_{2,0}$ than NSM because SM capitulates to state B_0 once it reaches state $B_{1,2}$ but NSM does not.

4 TRIMMING THE STRATEGY SPACE

Analyzing the revenue of *all* possible strategies for Miner 1 is quite unwieldy. Therefore, our first goal is to reduce the space of possible strategies to ones which are simpler to analyze *while guaranteeing that this simpler space still contains an optimal strategy.* We accomplish this through a series of reductions. This section provides a series of three "elementary" reductions which build upon each other. That is, the conclusions in each section should not be surprising, although it is challenging to rigorously prove this (examples throughout Appendix **??** are used to highlight the challenges). Sections 4.1 through 4.3 provide our three reductions. Section 4.4 provides the main theorem statement of this section: there is an optimal *trimmed* strategy. Many proofs are omitted, and can be found in Appendix **??**.

4.1 Step 1: Timeserving

We first show that, w.l.o.g., every strategy only publishes blocks which will be ancestors of the longest chain at the end of that round.

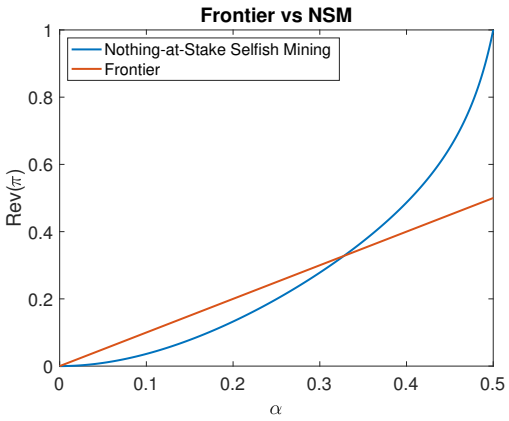

Fig. 5. Payoff comparison between FRONTIER and Nothing-at-Stake Selfish Mining.

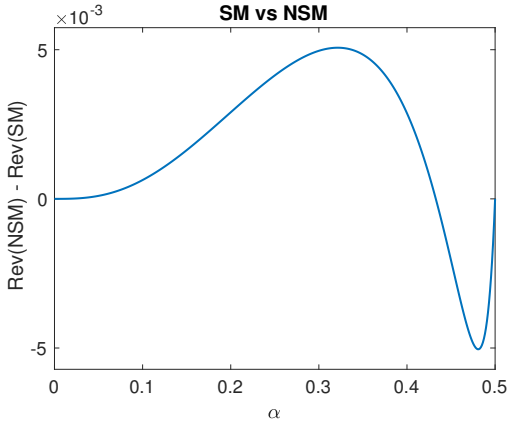

Fig. 6. Payoff comparison between Selfish Mining and Nothing-at-Stake Selfish Mining. Observe NSM is slightly better than SM for α close to $1/3$, but SM outperforms NSM for $\alpha > 0.44$.

Definition 4.1 (Timeserving). *The action PublishSet(V', E') is* Timeserving *if all blocks in V' immediately enter the longest chain (formally: if the action is taken during round n, then $V' \subseteq A(C(n))$). A strategy is Timeserving if when played against FRONTIER, with probability 1, all PublishSet(V', E') actions it takes are Timeserving.*

It is easy to see that FRONTIER is itself Timeserving: it publishes at most a single block at a time, and that block is the new unique longest chain. We first argue that there exists an optimal strategy against FRONTIER which is also Timeserving.

Theorem 4.2 (Timeserving). *For any strategy π, there is a strategy $\tilde{\pi}$ that is Timeserving, takes a valid action at every step, and satisfies $\textsc{Rev}_\gamma^{(n)}(\tilde{\pi}) = \textsc{Rev}_\gamma^{(n)}(\pi)$ for all γ and $n \in \mathbb{N}$.*

We now state three basic properties of Timeserving strategies.

Observation 4.3. *If π is Timeserving, then*

(i) Whenever π publishes blocks, it publishes a single path. Formally, whenever π takes action PublishSet(V', E') in round n, with $V' = \{b_1, \ldots, b_k\}$ ($b_i < b_{i+1}$ for all i), then E' contains an edge $b_{i+1} \to b_i$ for all $i \in [k-1]$, and an edge $b_1 \to b$ for some $b \in V$.
(ii) There are never two leaves of the same height. Formally, for all leaves $q \neq \tilde{q} \in V$, $h(q) \neq h(\tilde{q})$.
(iii) Whenever π forks, it publishes at least two blocks. Formally, whenever π takes the action PublishSet(V', E') which removes the old longest chain from the longest path, then $|V'| \geq 2$.

Observation 4.3 gives us some nice structure about Timeserving strategies (and Theorem 4.2 asserts that it is w.l.o.g. to study such strategies). In particular, we only need to consider strategies which publish a single path at a time. Formally, we may w.l.o.g. replace the action PublishSet(V', E') with the action:

Definition 4.4 (PublishPath). *Taking action PublishPath(V', u) with $u \in V$ and $V' \subseteq \mathcal{U}$ is equivalent to taking action PublishSet(V', E'), where E' contains an edge from the minimum element of V' to u, and an edge from v to the largest element of V' strictly less than v, for all other $v \in V'$.*

4.2 Step 2: Orderly

Section 4.1 provides structure on *when* we may assume blocks are announced, but it does not yet provide structure on *which blocks* are announced. Specifically, for all we know right now it could still be that when a strategy chooses to take action PublishPath(V', u), and $|V'| = k$, the precise k blocks it chooses to publish matter (e.g. in state $B_{2,0}$ it could choose to publish $2 \to 0$ versus $1 \to 0$). Our next reduction shows that it is without loss to consider only strategies which are *Orderly*, and always publish the earliest legal blocks. Intuitively, this gives the strategy more flexibility later on. For simplicity of notation, we introduce the terms $\min^{(k)}\{S\} \subseteq S$ to refer to the $\min\{k, |S|\}$ smallest elements in S and $\max^{(k)}\{S\} \subseteq S$ to refer to the $\min\{k, |S|\}$ largest elements in S.

Definition 4.5 (Orderly). *The action PublishPath(V', u) is* Orderly *if $V' = \min^{(|V'|)}(\mathcal{U} \cap (u, \infty))$. That is, an action is Orderly if it publishes the smallest $|V'|$ blocks it could have possibly published on top of u. A strategy is Orderly if when played against FRONTIER, with probability 1, all PublishPath$(\cdot, \cdot)$ actions it takes are Orderly.*

Theorem 4.6 (Orderly). *Let π be any Timeserving strategy. Then there is a valid, Timeserving, Orderly strategy $\tilde{\pi}$ that satisfies $\textsc{Rev}_\gamma^{(n)}(\tilde{\pi}) = \textsc{Rev}_\gamma^{(n)}(\pi)$ for all γ and $n \in \mathbb{N}$.*

We conclude this section by noting that, after restricting attention to Orderly strategies, we can further replace the action PublishPath(V', u) with the action:

Definition 4.7 (Publish). *Taking action Publish(k, u) with $k \in \mathbb{N}_+$ and $u \in V$ is equivalent to taking the action PublishPath$(\min^{(k)}(\mathcal{U} \cap (u, \infty)), u)$.*

4.3 Step 3: Longest Chain Mining

We now have structure on *when* blocks are published, and *which* blocks are published, but not yet on *where* those blocks are published. Specifically, an *orphaned chain* is a path in Tree that used to be part of the longest path $A(C)$ but was overtaken by another path. Intuitively, a chain can only be orphaned by Miner 1 and if Miner 1 is playing according to an optimal strategy, publishing blocks which build on top of orphaned chains should be sub-optimal. We define a strategy as *Longest Chain Mining* if it never publishes on top of a block in an orphaned chain.

Definition 4.8 (Longest Chain Mining). *Action Publish(k, u) is* Longest Chain Mining *(LCM) if $u \in A(C)$ is a block in the longest path. That is, an action is LCM if it builds on top of some block within the longest path (not necessarily the leaf). A strategy is LCM if, with probability 1, every Publish action it takes against FRONTIER is LCM.*

Previous work on Proof-of-Work mining games [15, 21] assume all strategies are LCM. For Proof-of-Stake mining games, Theorem 4.9 proves that it is w.l.o.g. to assume an LCM strategy.

Theorem 4.9 (LCM). *Let π be any Timeserving, Orderly strategy. Then there is a $\tilde{\pi}$ that is Timeserving, Orderly, LCM, takes a valid action at every step, and satisfies $\textsc{Rev}_\gamma^{(n)}(\tilde{\pi}) \geq \textsc{Rev}_\gamma^{(n)}(\pi)$ for all γ and $n \in \mathbb{N}$.*

4.4 Step 4: Trimmed

With Theorems 4.2, 4.6 and 4.9, we can immediately conclude that there exists an optimal strategy satisfying several structural properties. We wrap up by showing one final property, and will show that there exists an optimal strategy which is *Trimmed*.

Definition 4.10 (Trimmed Action). *Action Publish(k, v) is* Trimmed *if $v \in A(C)$ and whenever v is not* the *longest chain (that is, $v \neq C$), and u is the unique node in $A(C)$ with an edge to v, then u was created by Miner 2 (that is, $u \in T_2$).*

Put another way, every Publish(k, v) either builds on top of the longest chain (in which case it is vacuously Trimmed), or kicks out the successors of v. In the latter case, an action is Trimmed if and only if the *minimum* successor of v was created by Miner 2.

Definition 4.11 (Trimmed Strategy). *A strategy is* Trimmed *if every action it takes is either Wait or Trimmed.*

We now conclude our main theorem of this section.

Theorem 4.12 (Trimming). *For all strategies π, there is a Trimmed strategy $\tilde{\pi}$ that take valid actions in every step, and $\textsc{Rev}_\gamma^{(n)}(\tilde{\pi}) \geq \textsc{Rev}_\gamma^{(n)}(\pi)$ for all γ and $n \in \mathbb{N}$.*

5 TRIMMING THE STATE SPACE

So far, we have greatly simplified strategies which we need to consider. However, we still have not even established that there exists an optimal strategy which is *recurrent*. That is, for all we know so far, the optimal strategy might need to store not only the entire longest chain, but also all blocks which have ever been published, and all unpublished blocks which they ever created. The goal in this section is to establish that an optimal strategy exists which is recurrent: it will eventually (with probability 1) reach a "checkpoint" which the strategy treats as a new genesis block that will never be overridden.

5.1 Checkpoints and Weak Recurrence

We iteratively define a sequence of blocks $P_0, P_1, \ldots$ in the longest path $A(C)$ to be checkpoints as follows.

Definition 5.1 (Checkpoints). *Based on the current state, checkpoints are iteratively defined as follows.*

- *The first checkpoint, P_0, is the genesis block.*
- *If P_{i-1} is undefined, then P_i is undefined as well.*
- *If P_{i-1} is defined, then v is a* potential i^{th} checkpoint *if:*
  - $v > P_{i-1}$.
  - $v \in A(C)$.
  - *Among blocks that Miner* 1 *created between P_{i-1} and v (including v, not including P_{i-1}), more are in the longest chain than unpublished. That is, $|A(C) \cap (P_{i-1}, v] \cap T_1| \geq |\mathcal{U}_1 \cap (P_{i-1}, v]|$.*
- *If there are no potential i^{th} checkpoints, then P_i is undefined.*
- *Else, then P_i is defined to be the minimum potential i^{th} checkpoint.*

Note that each P_i is again a random variable, meaning that a priori P_i might change over time, including from undefined to defined. For example, $P_i(n)$ would denote the i^{th} checkpoint, as defined by the state after the conclusion of the n^{th} round (we will later prove that there exists an optimal strategy which never changes or undefines P_i once it is defined. But this will be a result, and not a definition).

Example 5.2. *Consider the state in Figure 7 where blocks 0, 1, 5, and 7 are the checkpoints. By definition, block 0 is the base-case and is always a checkpoint. Block 1 is a checkpoint because Miner 1 has no unpublished blocks in the interval* $(0, 1]$. *Block 2 is unpublished and thus not a checkpoint. Block 3 is not a checkpoint because Miner 1 has one unpublished block in the interval* $(1, 3]$ *and zero published block in the path from* $3 \to 1$ *(not counting block 1). From a similar reasoning, blocks 4 and 6 are not checkpoints. Block 5 is a checkpoint because Miner 1 has one unpublished block in the interval* $(1, 5]$ *and one block in the path* $5 \to 4 \to 3 \to 1$ *(not counting block 1). Block 7 is a checkpoint because Miner 1 has no unpublished blocks in the interval* $(5, 7]$.

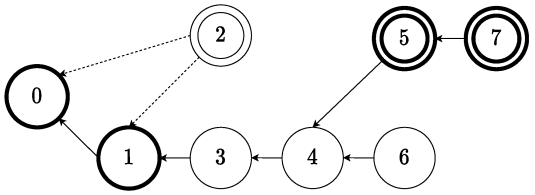

Fig. 7. Example of a state and its checkpoints. Blocks with thicker lines denote blocks that are checkpoints and blocks with thinner lines denote blocks that are not checkpoints. From Definition 5.1, blocks 0, 1, 5, and 7 are checkpoints.

The main result of this section is stated below, and claims that there exists an optimal strategy which treats checkpoints like the genesis block.

Definition 5.3 (Checkpoint Recurrent). *A strategy π is* Checkpoint Recurrent *if when π is played against FRONTIER:*

- *For all $i \in \mathbb{N}$, if P_i changes from undefined to defined, P_i never changes again (in particular, this implies that once P_i is defined, it remains in $A(C)$ forever).*
- *Immediately when P_i becomes defined, neither player has any unpublished blocks $> P_i$.*

When bullet one is satisfied, checkpoints are never overridden. Given that bullet one holds, bullet two implies that immediately when P_i is defined, it is essentially a genesis block (because bullet one holds, no unpublished blocks $< P_i$ can ever enter the longest chain. If bullet two also holds, then there are no unpublished blocks $> P_i$, so there are no relevant unpublished blocks, and P_i is in the longest chain forever, just like the genesis block in round 0). This implies that when optimizing over Checkpoint Recurrent strategies, it suffices to consider only strategies that reset its state space whenever a new checkpoint is defined. That is, whenever a new checkpoint is defined Miner 1 capitulates to state B_0.

Theorem 5.4 (Weak-Recurrence). *There exists an optimal strategy which is checkpoint recurrent.*

The weak-recurrence theorem provides a useful tool to reduce the state space of optimal strategies; however, it does not say how often (if ever) the block tree reaches a new checkpoint. Fortunately, each new checkpoint give us important information about the payoff of a strategic miner: *if the block tree never reaches a checkpoint, then at all times Miner 1 has at least half of their blocks unpublished.* Next, we check such strategies are not better than FRONTIER before diving into the proof of the weak-recurrence theorem.

Proposition 5.5. *If π is checkpoint recurrent and P_1 is never defined, then $\textsc{Rev}(\pi) \leq \textsc{Rev}(\textit{FRONTIER})$.*

As a first step toward proving upper bounds in the revenue, we will require a simply but useful fact about rate of growth of the block tree.

Lemma 5.6 (Minimum Growth Rate). *For any mining game starting at state $X_0 = B_0$,*

$$\liminf_{n\to\infty} \frac{h(C(X_n))}{n} \geq 1 - \alpha, \quad \textit{with probability 1.} \tag{5.1}$$

Corollary 5.7. *For any optimal strategy π, $\textsc{Rev}(\textit{FRONTIER}) = \alpha \leq \textsc{Rev}(\pi) \leq \frac{\alpha}{1-\alpha}$.*

Both Lemma 5.6 and Corollary 5.7 will be useful to prove Proposition 5.5. We need to understand one property of checkpoints, and then we can complete the proof. Intuitively, Proposition 5.8 and Corollary 5.9 just apply the definition of checkpoints to relate the number of blocks that Miner 1 has unpublished vs. published in the longest path.

Proposition 5.8. *For all $v \in A(C)$,*

(i) If v is a checkpoint, then for all checkpoints $P_i > v$, $|A(C) \cap (v, P_i] \cap T_1| \geq |\mathcal{U} \cap (v, P_i)|$.

(ii) If v is not a checkpoint and $P_i = \max\{P_j : P_j < v\}$, then $|A(C) \cap (P_i, v] \cap T_1| < |\mathcal{U} \cap (P_i, v]|$.
(iii) If v is not a checkpoint, then for all checkpoints $P_i > v$, $|A(C) \cap (v, P_i] \cap T_1| > |\mathcal{U} \cap (v, P_i]|$.

Corollary 5.9. *Suppose $v \in A(C)$ is not a checkpoint and let P_i be the highest checkpoint below v. Then Miner 1 publishes less than half of all blocks they created from time $P_i + 1$ to v. That is, $|A(C) \cap (P_i, v] \cap T_1| < \frac{|T_1 \cap (P_i, v]|}{2}$.*

Proof. Bullet (ii), Proposition 5.8, implies

$$|A(C) \cap (P_i, v] \cap T_1| + |\mathcal{U} \cap (P_i, v]| > 2|A(C) \cap (P_i, v] \cap T_1|.$$

Suppose for contradiction $|A(C) \cap (P_i, v] \cap T_1| \geq \frac{|T_1 \cap (P_i, v)|}{2}$. Then, the number of unpublished blocks plus blocks in the longest path would be strictly bigger than the number of blocks Miner 1 created, a contradiction. □

Proof of Proposition 5.5. From the strong law of large numbers, Corollary 5.9 and Lemma 5.6,

$$\limsup_{n\to\infty} \frac{|A(C(X_n)) \cap T_1|}{|A(C(X_n))|} \leq \limsup_{n\to\infty} \frac{\frac{|T_1 \cap (0,n]|}{2n}}{\frac{|A(C(X_n))|}{n}} \leq \frac{\limsup_{n\to\infty} \frac{|T_1 \cap (0,n]|}{2n}}{\liminf_{n\to\infty} \frac{|A(C(X_n))|}{n}} \leq \frac{\alpha/2}{(1-\alpha)} \leq \alpha.$$

where the last inequality uses the fact $\alpha \leq 1/2$. Intuitively, if Miner 1 never reaches a checkpoint, then they are publishing at most $\alpha n/2$ blocks in expectation by round n (and clearly at most $\alpha n/2$ of these can be in the longest path). But Lemma 5.6 asserts that there are at least $(1-\alpha)n$ blocks in expectation in the longest path by round n. Therefore, the fraction produced by Miner 1 cannot be too high (and in particular, honesty would have been better in expectation). □

We prove Theorem 5.4 in two steps, Section 5.2 and Section 5.3.

5.2 Step 1: Checkpoint Preserving

The first step is to show the existence of an optimal strategy that never forks a checkpoint. For that, we will give an explicitly procedure f to transform any strategy π that could fork checkpoints into another strategy $f(\pi)$ that does not fork checkpoints satisfying $\textsc{Rev}(f(\pi)) \geq \textsc{Rev}(\pi)$.

Definition 5.10 (Finality). *A block $q \in A(C)$ reaches* finality *with respect to strategy π if, with probability 1, π takes no action that removes q from longest path.*

Definition 5.11 (Checkpoint Preserving). *A strategy π is* checkpoint preserving *if whenever a new checkpoint P_i is defined, P_i reaches finality with respect to π.*

Theorem 5.12. *For every strategy π, there is a trimmed, checkpoint preserving strategy $f(\pi)$ with $\textsc{Rev}(f(\pi)) \geq \textsc{Rev}(\pi)$.*

5.3 Step 2: Opportunistic

We have shown the existence of an optimal strategy that would never fork the longest chain C when it becomes a checkpoint. However, to be checkpoint recurrent, we must also show Miner 1 has no unpublished blocks bigger than C when the longest chain is a checkpoint. The converse can only happen when Miner 1 is about to take action PublishPath(Q, v) and max Q will reach finality with respect to Miner 1's strategy, but Miner 1 would leave an unpublished block $q >$ max Q. The intuition is that Miner 1 can wait instead of publishing Q pointing to v in current round. If Miner 1 creates the next block, Miner 1 can publish Q pointing to v as before. If Miner 2 creates the next block, Miner 1 can still take action PublishPath$(Q \cup \{q\}, v)$ adding $Q \cup \{q\}$ to the longest path.

Definition 5.13 (Opportunistic). *Let π be a strategy and let B be a state. Action PublishPath(Q, v) is* opportunistic *with respect to B and π if*

- *PublishPath*(Q, v) *is a valid action at state* B.
- *If* π *takes action PublishPath*(Q, v) *where* max Q *reaches finality with respect to* π *(Definition 5.10), then* $Q = \mathcal{U}(B) \cap (v, \infty)$.

Strategy π *is* opportunistic *if at all states* B, π *waits or takes an opportunistic action with respect to* B *and* π.

Theorem 5.14. *For any strategy* π*, there is a valid, trimmed, checkpoint preserving and opportunistic strategy* $f(\pi)$ *with* $\textsc{Rev}(f(\pi)) \geq \textsc{Rev}(\pi)$.

Proof of Theorem 5.4. Theorem 5.14 directly implies the weak-recurrence theorem since a checkpoint preserving and opportunistic strategy is also checkpoint recurrent. □

5.4 Step 3: Strong Recurrence

So far, we have shown that there exist an optimal strategy that is checkpoint recurrent. That is, once we reach a state X_t where $C(X_t)$ is a checkpoint, Miner 1 capitulates to state B_0. Next, we will aim for a stronger result.

Theorem 5.15 (Strong recurrence). *There exists an optimal checkpoint recurrent and positive recurrent strategy.*

For a proof sketch, observe the Weak Recurrence Theorem implies there exists an optimal strategy π that is checkpoint recurrent. We will assume π is not positive-recurrent (i.e., the expected time $\mathbb{E}[\tau]$ to define a new checkpoints is infinite) and derive that $\textsc{Rev}(\pi) \leq \alpha = \textsc{Rev}(\text{FRONTIER})$. The case where Miner 1 never defines checkpoint P_1 – i.e., $\tau = \infty$ with probability 1 in Proposition 5.5 – give us intuition why the claim should hold to the more general case where $\mathbb{E}[\tau] = \infty$. Once we proof $\textsc{Rev}(\pi) \leq \textsc{Rev}(\text{FRONTIER})$, we just observe FRONTIER is checkpoint and positive recurrent. Thus there exists an optimal checkpoint and positive recurrent strategy.

6 NASH EQUILIBRIUM

We briefly give intuition behind our second main result, which leverages Theorem 5.15 to lower bound α^{PoS}.

Theorem 6.1. *For* $\alpha \leq 0.308$*, FRONTIER is an optimal strategy for Miner 1 when Miner 2 follows FRONTIER.*

We defer the proof to Appendix **??**. The main idea behind the proof is to show that Nothing-at-Stake Selfish Mining is almost optimal when $\alpha < 1/3$. The following proof-sketch highlights the main insights of the proof.

Selfish Mining is optimal when Miner 2 creates the first block (when $\alpha < 1/3$*).* We know that there is an optimal checkpoint recurrent strategy, Theorem 5.15. Therefore, it is optimal for Miner 1 to capitulate to state B_0 if Miner 2 creates and publishes $1 \longrightarrow 0$ (which is exactly what selfish mining does).

Selfish Mining is optimal after Miner 1 creates and withholds blocks 1 and 2. Starting from state $B_{2,0}$, selfish mining will wait until the first time step τ when the lead decrease to a single block to fork all of Miner 2 blocks. We show that waiting until time τ is indeed optimal for any value of α (which is not surprising). Less obvious is why Miner 1 must publishes all his blocks at time τ when they still have a lead of a single block. Indeed we should not expect this to be optimal for all values of α. If Miner 1 waits at time τ and creates the next block, they will again have a lead of two blocks and can resume to "selfish mine". Here, we shown that Miner 1 creates $\frac{\alpha}{1-2\alpha}$ blocks on expectation from the time they have a lead of two blocks to the moment the lead decreases to a single block.

This quantity can be arbitrarily large but it is at most 1 when $\alpha < 1/3$ so it is a risky action for Miner 1. That is because if (instead) Miner 2 creates next block, there is a tie (Miner 1 does not have enough blocks to fork the longest chain) and we can show that the probability Miner 1 will ever publish any blocks created before time τ is at most $\frac{\alpha}{1-\alpha}$. Formally, we will prove Miner 1 maximizes rewards by waiting until time τ and immediately publishing all unpublished blocks (which is what selfish mining does).

There is little window to improve Selfish Mining when Miner 1 creates and withhold block 1 and Miner 2 publishes $2 \to 0$. Winning the tie-breaking at state $B_{1,1}$ is another source of revenue for Miner 1. In fact, the only improvement that Nothing-at-Stake Selfish Mining provides over standard Selfish Mining is increasing the probability that Miner 1 wins the tie-breaking between blocks 1 and 2. From a similar argument from previous bullet, we can show the probability of adding block 1 to the longest path is at most $\frac{\alpha}{1-\alpha}$. Next, we observe Miner 1 has no more advantage of being the creator of the block at height $\ell \geq 2$ at state $B_{1,1}$ than being the creator of the block at height $\ell - 1$ at state B_0. We formalize this intuition by showing that, by ignoring blocks 1 and 2, any action taken on a state reachable from $B_{1,1}$ can be converted into an action for an state reachable from B_0. The only advantage state $B_{1,1}$ provides over state B_0 is that Miner 1 has a probability (of at most $\frac{\alpha}{1-\alpha}$) of adding block 1 to the longest path.

Wrapping up. From the discussion above, state $B_{1,1}$ is the only state where we could possible search for a better strategy than Nothing-at-Stake Selfish Mining when $\alpha < 1/3$, but there is little window to improve Miner 1's action at state $B_{1,1}$. As a result, we will obtain FRONTIER is optimal when $\alpha \leq 0.308$ as desired.

7 CONCLUSION

We study miner incentives in longest-chain proof-of-stake protocols with perfect external randomness. We show both that such protocols are strictly more vulnerable to manipulation than those based on proof-of-work (Theorem 3.4), but also that it is a Nash equilibrium for all miners to follow the longest-chain protocol as long as no miner has more than ≈ 0.308 of the total stake (Theorem 6.1). Our main technical results characterize potentially optimal strategies in a complex, infinite-state MDP (Theorem 5.15). Our work motivates several natural open problems:

- Theorem 5.15, combined with the analysis in Theorem 6.1, provides strong structure on optimal strategies. It is therefore conceivable that a simulation-based approach with MDP solvers (as in [21]) could estimate α^{PoS} to high precision.
- Our Theorem 6.1 provides a reduction from incentive-compatible longest-chain proof-of-stake protocols to designing a randomness beacon and a slashing protocol. Clearly, it is important for future work to construct these primitives, although these are well-known and ambitious open problems. In our setting, it is further important to understand what are the *minimal* assumptions on a randomness beacon or slashing protocol necessary to leverage Theorem 6.1.

REFERENCES

[1] Nick Arnosti and S. Matthew Weinberg. 2019. Bitcoin: A Natural Oligopoly. In *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA (LIPIcs, Vol. 124)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 5:1–5:1.

[2] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. 2018. Verifiable delay functions. In *Annual international cryptology conference*. Springer, 757–788.

[3] Joseph Bonneau, Jeremy Clark, and Steven Goldfeder. 2015. On Bitcoin as a public randomness source. *IACR Cryptol. ePrint Arch.* 2015 (2015), 1015.

[4] Jonah Brown-Cohen, Arvind Narayanan, Alexandros Psomas, and S Matthew Weinberg. 2019. Formal barriers to longest-chain proof-of-stake protocols. In *Proceedings of the 2019 ACM Conference on Economics and Computation*. 459–473.

[5] Miles Carlsten, Harry A. Kalodner, S. Matthew Weinberg, and Arvind Narayanan. 2016. On the Instability of Bitcoin Without the Block Reward. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*. ACM, 154–167.

[6] Xi Chen, Christos H. Papadimitriou, and Tim Roughgarden. 2019. An Axiomatic Approach to Block Rewards. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies, AFT 2019, Zurich, Switzerland, October 21-23, 2019*. ACM, 124–131.

[7] Jeremy Clark and Urs Hengartner. 2010. On the Use of Financial Data as a Random Beacon. *EVT/WOTE* 89 (2010).

[8] Phil Daian, Rafael Pass, and Elaine Shi. 2019. 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 (Lecture Notes in Computer Science, Vol. 11598)*. Springer, 23–41.

[9] Ittay Eyal and Emin Gün Sirer. 2014. Majority is not enough: Bitcoin mining is vulnerable. In *International conference on financial cryptography and data security*. Springer, 436–454.

[10] Matheus V. X. Ferreira, Daniel J. Moroz, David C. Parkes, and Mitchell Stern. 2021. Dynamic Posted-Price Mechanisms for the Blockchain Transaction-Fee Market. arXiv:2103.14144 [cs.GT]

[11] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. 2017. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles*. 51–68.

[12] LM Goodman. 2014. Tezos: A self-amending crypto-ledger position paper. *Aug* 3 (2014), 2014.

[13] Gur Huberman, Jacob Leshno, and Ciamac Moallemi. 2020. Monopoly without a Monopolist: An Economic Analysis of the Bitcoin Payment System. *Review of Economic Studies* (2020).

[14] John Kelsey, Luís TAN Brandão, Rene Peralta, and Harold Booth. 2019. *A reference for randomness beacons: Format and protocol version 2*. Technical Report. National Institute of Standards and Technology.

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

[16] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. 2017. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual International Cryptology Conference*. Springer, 357–388.

[17] Jacob Leshno and Philipp Strack. 2020. Bitcoin: An Impossibility Theorem for Proof-of-Work based Protocols. *American Economics Review: Insights* (2020).

[18] Satoshi Nakamoto. 2007. *Bitcoin: A peer-to-peer electronic cash system*. Technical Report.

[19] Michael Neuder, Daniel J. Moroz, Rithvik Rao, and David C. Parkes. 2021. Selfish Behavior in the Tezos Proof-of-Stake Protocol. *Cryptoeconomic Systems* 0, 1 (5 4 2021). https://doi.org/10.21428/58320208.27350920 https://cryptoeconomicsystems.pubpub.org/pub/neuder-selfish-behavior-tezos.

[20] Michael O Rabin. 1983. Transaction protection by beacons. *J. Comput. System Sci.* 27, 2 (1983), 256–267.

[21] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. 2016. Optimal selfish mining strategies in bitcoin. In *International Conference on Financial Cryptography and Data Security*. Springer, 515–532.

[22] Gavin Wood et al. 2014. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper* 151, 2014 (2014), 1–32.