Strategic Mining in Proof-of-Stake with Practical Random Election

***Abstract*—The security of blockchain systems relies on the honest majority assumption. However, strategic mining threatens this assumption, because selfish miners can gain more block rewards than honest miners by attacks such as withholding blocks. Due to its significant implication, blockchain mining games have been studied in PoW and PoS under various settings using different methods. Nonetheless, this paper argues that the practical limitation of random beacons has not been exploited in strategic mining in PoS blockchains.**

Current PoS blockchains use random beacons to randomly select validators for each slot. However, the randomness is usually fixed for multiple slots, due to the latency of distributed random beacon protocols. This means that validators actually know the exact election results for future slots within an epoch, which contrasts with the Markov process models in previous analysis.

In this paper, we formalize this "Lookahead Property" and present Lookahead-Aware Mining (LAM), a deterministic optimal strategy that exploits the predictability of epoch-based leader schedules. By mapping the mining process to a Weighted Interval Scheduling problem on the difference of prefix sums, we derive an $O(T \log T)$ algorithm that allows adversaries to compute the exact optimal attack strategy in real-time. Our simulations demonstrate that LAM enables miners with as little as 5% stake to outperform honest mining, significantly lowering the security threshold compared to probabilistic MDP-based models which require >30% stake to be profitable.

I. Introduction

The security of blockchain consensus protocols relies on a super-majority of honest miners or validators. For example, the Bitcoin consensus protocol, based on Proof-of-Work (PoW) and the longest-chain fork selection rule, requires that more than 50% of the hash power be controlled by honest miners. Proof-of-Stake (PoS) protocols also assume that honest validators own more than 50% of the total stake. If the honest majority assumption does not hold, disastrous attacks might happen, including double spending and history reversion. Even if malicious miners cannot afford to become the super-majority, blockchain security is further undermined by selfish mining attacks. Eyal and Sirer [1] showed that in PoW, a miner with $\alpha < 1/2$ hash power can publish more than α fraction of blocks in the finalized chain by strategically withholding blocks. This unfair gain allows them to grow relatively richer over time, potentially exceeding the 50% threshold.

Proof-of-Stake protocols have gained popularity due to their energy efficiency, as evidenced by Ethereum's transition to PoS via "The Merge". Strategic mining in PoS has also been studied; for instance, Ferreira et al. [2] showed that a malicious validator with $\approx 31\%$ of the total stake can outperform honest mining under assumptions of perfect "per-slot" randomness. However, practical PoS systems often deviate from this theoretical ideal. For security and efficiency reasons, decentralized random beacons (e.g., RANDAO in Ethereum 2.0 [3]) emit fresh random numbers only once per epoch. Consequently, the leader schedule is fixed for the entire epoch (e.g., 32 slots in Ethereum). This introduces what we term the **Lookahead Property**: validators know exactly which future slots belong to them for the duration of the epoch. In contrast to PoW (where the next leader is strictly probabilistic), this deterministic lookahead allows malicious validators to plan sophisticated fork attacks with arbitrarily small stake.

Our Contribution. Our main contribution is to formalize and analyze the impact of the Lookahead Property on the security of Proof-of-Stake protocols. We acknowledge that Sarenche et al. [4] have comprehensively solved this problem, presenting optimal algorithms for both semi-predictable and fully predictable settings. Their use of Reinforcement Learning provides additional capabilities for complex scenarios. This work, initiated in December 2023, represents an independent formulation of the optimal strategy for the single-epoch setting. Specifically:

1) We formalize the **System Model** for PoS with epoch-based randomness, modeling the adversary's view as a known binary string of leadership slots.
2) We present the **Lookahead-Aware Mining (LAM)** algorithm, a deterministic optimal strategy that maps the mining problem to Weighted Interval Scheduling. This allows efficient computation of the optimal attack strategy in $O(T \log T)$ time using Fenwick Trees.
3) We demonstrate through simulation that LAM significantly outperforms standard MDP-based strategies. We show that an adversary with as little as 5% stake can achieve profitable deviations from honest mining, lowering the security threshold compared to previous probabilistic models.

In practice, mining rewards include block rewards and transaction fees. For simplicity, we focus on maximizing the share of valid blocks (relative revenue), which directly correlates with total rewards in most inflation-based reward schemes.

II. Related Work

A. Selfish Mining in Proof-of-Work

Selfish mining was first formalized by Eyal and Sirer [1] in the context of Bitcoin. They demonstrated that a miner with $> 33\%$ of the hash power could profit by withholding blocks. This line of research modeled mining as a Markov

Decision Process (MDP) [5], considering network latency and propagation delays. In these PoW settings, the leader selection is purely probabilistic; miners only know the probability distribution of the next leader.

B. Strategic Mining in Proof-of-Stake

Early PoS analysis [2] extended the MDP framework to PoS, assuming an ideal "per-slot" randomness where the leader for slot t is revealed only at slot t. Under this assumption, the strategic landscape is similar to PoW, with adversaries requiring $\approx 31\%$ stake to profit. However, modern PoS protocols sacrifice perfect per-slot unpredictability for efficiency, utilizing epoch-based randomness which creates a window of predictability.

C. Predictable Randomness and Lookahead

The impact of "proposer predictability" has been comprehensively analyzed by Sarenche et al. [4] in concurrent work. They formalized the settings of both semi-predictable and fully predictable leader schedules. Crucially, they presented optimal algorithms for the predictable epoch setting and further employed Deep Reinforcement Learning (DRL) to find strategies in more complex environments where exact optimization is intractable. Our work, developed independently in late 2023, focuses on the deterministic optimal algorithm (LAM) for the single-epoch case. Our derivation via Weighted Interval Scheduling aligns with the optimal strategies established in [4], serving as an independent verification of their theoretical bounds for adversary revenue.

D. Randomness Generation in Distributed Systems

Secure distributed random number generation (DRNG) schemes like RANDAO [6] or VDFs [7] focus on unbiased generation. However, once the seed is generated, if it applies to a long epoch, the lookahead vulnerability persists regardless of the generation method's quality.

III. System Model

We consider a Proof-of-Stake (PoS) blockchain system where time is divided into discrete units called *slots*. A fixed number of slots T constitutes an *epoch*.

A. Consensus and Randomness

Leader Selection.. In each slot $t \geq 1$, a single validator is selected as the *slot leader* to propose a block. The selection is performed by a randomized functionality based on the validator's stake. Let α be the stake ratio of a strategic adversary $\mathcal{A}$, and $\beta = 1 - \alpha$ be the stake ratio of the honest majority $\mathcal{H}$. For any slot t, the leader L_t is $\mathcal{A}$ with probability α and $\mathcal{H}$ with probability β.

Epoch-Based Randomness.. Existing PoS protocols (e.g., Ethereum 2.0, Cardano) typically update the randomness source for leader selection only at epoch boundaries due to the latency of distributed random beacons or VDFs. This leads to the **Lookahead Property**:

Definition 1 (Lookahead Property)**.** *At the beginning of an epoch k, all validators know the sequence of leaders for all slots in epoch k.*

This contrasts with previous theoretical models where the leader of slot t is revealed only at t.

B. Game Abstraction

We model the mining process in a single epoch as a game played over a binary string $S \in \{0,1\}^T$.

- $S[t] = 0$: The leader of slot t is the Adversary $\mathcal{A}$.
- $S[t] = 1$: The leader of slot t is the Honest miner $\mathcal{H}$.

Ideally, the blockchain grows by one block per slot, including all honest and adversarial blocks. The honest miner $\mathcal{H}$ follows the standard protocol: always publish a block when selected, extending the longest known chain. The adversary $\mathcal{A}$ can deviate by withholding blocks and releasing them later to create forks.

C. Objective Function

The goal of the adversary is to maximize their *relative revenue* in the final canonical chain $\mathcal{C}_{final}$. Let $n_{\mathcal{A}}(\mathcal{C})$ and $n_{\mathcal{H}}(\mathcal{C})$ be the number of blocks owned by $\mathcal{A}$ and $\mathcal{H}$ in chain $\mathcal{C}$, respectively. The payoff is defined as:

$$U(\mathcal{C}_{final}) = \frac{n_{\mathcal{A}}(\mathcal{C}_{final})}{n_{\mathcal{A}}(\mathcal{C}_{final}) + n_{\mathcal{H}}(\mathcal{C}_{final})} \tag{1}$$

Since maximizing this ratio is analytically complex, we often tackle the equivalent problem of maximizing $n_{\mathcal{A}} - \frac{\alpha}{1-\alpha} n_{\mathcal{H}}$ or simply minimizing $n_{\mathcal{H}}$ while keeping $n_{\mathcal{A}}$ constant (since $\mathcal{A}$ can typically include all their own blocks in a successful attack). In our single-epoch analysis with Lookahead, we focus on maximizing the number of *excluded* honest blocks.

IV. Lookahead-Aware Mining (LAM)

In this section, we present the **Lookahead-Aware Mining (LAM)** algorithm. We first define the optimization problem for a single epoch where the leader schedule is fully known. We then map this problem to the Weighted Interval Scheduling problem. While a naive solution runs in $O(T^2)$, we show that the specific structure of our problem allows for an optimization to $O(T \log T)$, enabling efficient calculation even for large epochs where T is in the thousands.

A. Single-Epoch Optimization

Consider a single epoch of length T. Let $S \in \{0,1\}^T$ be the leader schedule, where $S[t] = 0$ implies the leader is the adversary $\mathcal{A}$, and $S[t] = 1$ implies the leader is the honest validator $\mathcal{H}$.

The adversary's goal is to construct a private chain that, when released, reorganizes the public chain to exclude as many honest blocks as possible. A specific attack consists of a fork starting at slot t_{start} and ending at slot t_{end}. To successfully reorganize the honest chain during this interval, the adversary must produce a fork that is strictly longer than the honest fork in the same interval.

Definition 2 (Attackable Interval)**.** *An interval $I = [i, j]$ (with $1 \leq i < j \leq T$) is **attackable** if the number of adversarial slots in I is strictly greater than the number of honest slots in I:*

$$\texttt{count}(S[i \dots j], 0) > \texttt{count}(S[i \dots j], 1) \tag{2}$$

If an interval I is attackable, the adversary can withhold all blocks during this interval and release a fork at slot j. The **weight** (gain) of such an attack is the number of honest blocks excluded:

$$w(I) = \texttt{count}(S[i \dots j], 1) \tag{3}$$

B. The LAM Algorithm

The optimal strategy for the adversary is to select a set of disjoint attackable intervals $\mathcal{I}_{opt} = \{I_1, I_2, \dots, I_m\}$ such that the total weight $\sum_{I \in \mathcal{I}_{opt}} w(I)$ is maximized.

Naive $O(T^2)$ ***DP.*** Let $\mathsf{DP}[j]$ be the maximum weight achievable considering slots up to j.

$$\mathsf{DP}[j] = \max\Big(\mathsf{DP}[j-1], \max_{\substack{i<j \\ I=[i,j] \text{ attackable}}} \{\mathsf{DP}[i-1] + w([i,j])\}\Big) \tag{4}$$

Identifying all attackable intervals takes $O(T^2)$.

Optimal $O(T \log T)$ ***Algorithm.*** We can optimize the recurrence. Let $P_0[t]$ and $P_1[t]$ be the prefix counts of adversarial and honest slots respectively. An interval $[i, j]$ is attackable iff:

$$\begin{aligned} &(P_0[j] - P_0[i-1]) > (P_1[j] - P_1[i-1]) \\ \iff &(P_0[j] - P_1[j]) > (P_0[i-1] - P_1[i-1]) \end{aligned}$$

Let $D[t] = P_0[t] - P_1[t]$. The condition becomes $D[j] > D[i-1]$. The weight is $w([i, j]) = P_1[j] - P_1[i-1]$. Substituting into the recurrence:

$$\mathsf{DP}[j] = \max\Big(\mathsf{DP}[j-1], P_1[j] + \max_{i \text{ s.t. } D[i-1] < D[j]} \{\mathsf{DP}[i-1] - P_1[i-1]\}\Big) \tag{5}$$

This is a Range Maximum Query problem. As we iterate j from 1 to T, we want to query $\max\{\mathsf{DP}[k] - P_1[k]\}$ for all k where $D[k] < D[j]$. Since values of $D[k]$ are integers in $[-T, T]$, we can use a Fenwick Tree (Binary Indexed Tree) or Segment Tree over the domain of D values to support point updates and prefix maximum queries in $O(\log T)$.

C. Multi-Epoch Comparison

In a realistic setting, the adversary only has knowledge of the leader schedule for the current epoch (and possibly the immediate next epoch depending on the protocol), but not for the indefinite future. Therefore, LAM is executed independently for each epoch. However, to evaluate the performance of LAM, we define a theoretical **Global Optimal** strategy. This strategy assumes the adversary has omniscience over all K future epochs. We can simulate this by concatenating the schedules of all K epochs and running the $O(KT \log(KT))$ algorithm on the combined sequence. This provides an upper bound on achievable revenue.

Input: Schedule S, Length T
Calculate P_1 and D. Offset D to be positive.
Initialize Fenwick Tree BIT with $-\infty$
Add $(\mathsf{DP}[0] - P_1[0])$ at index $D[0]$ in BIT
for $j \leftarrow 1$ to T **do**
  $\mathsf{DP}[j] \leftarrow \mathsf{DP}[j-1]$
  $M \leftarrow \texttt{QueryMax}(BIT, D[j]-1)$
  if $M \neq -\infty$ **then**
    $\mathsf{DP}[j] \leftarrow \max(\mathsf{DP}[j], M + P_1[j])$
  end if
  $\texttt{Update}(BIT, D[j], \mathsf{DP}[j] - P_1[j])$
end for
Return $\mathsf{DP}[T]$

Algorithm 1: Fast LAM ($O(T \log T)$)

V. Experiments

In this section, we estimate the payoff gain achieved by our Lookahead-Aware Mining (LAM) strategy through simulation. We compare our results with the standard MDP-based strategy proposed by Ferreira et al. [2], which assumes no knowledge of future leader elections (per-slot randomness).

A. Experimental Setup

We simulate a process of 10 epochs, with each epoch consisting of $T = 100$ slots. The adversary $\mathcal{A}$ controls a fraction α of the total stake, while the honest validators $\mathcal{H}$ control the remaining $\beta = 1 - \alpha$. We vary α from 0.05 to 0.48. For each α, we run $10,000$ independent simulations to estimate the expected relative revenue.

We compare three strategies:

- **LAM (Ours)**: The adversary uses the *Fast LAM* algorithm (Sec. IV) within each epoch independently. Since the adversary only knows the leader schedule for the current epoch, the optimization is reset at each epoch boundary.
- **Global Optimal**: An idealized strategy where the adversary has knowledge of leader schedules across all 10 epochs simultaneously (simulated by concatenating schedules and running interval scheduling on the full sequence). This serves as a theoretical upper bound.
- **MDP Strategy (No Lookahead)**: The optimal policy for PoS with per-slot randomness [2]. In this model, the adversary probabilistically decides to withhold or publish based on the current state (lengths of private and public chains), assuming the next leader is chosen with probability α.

B. Results

The results are summarized in Table I. The values represent the *relative revenue* of the adversary, defined as the fraction of blocks in the final canonical chain that were mined by $\mathcal{A}$.

C. Analysis

The experiments demonstrate that the LAM strategy significantly outperforms the MDP baseline. Notably, the MDP

TABLE I
RELATIVE REVENUE OF ADVERSARY WITH VARYING STAKE α

Stake α	LAM (Ours)	Global Opt.	MDP (Baseline)
0.05	0.0503	0.0503	0.0105
0.10	0.1019	0.1020	0.0360
0.15	0.1566	0.1566	0.0606
0.20	0.2174	0.2179	0.1311
0.25	0.2838	0.2864	0.1929
0.30	0.3736	0.3736	0.2783
0.32	0.4066	0.4092	0.3140
0.34	0.4450	0.4480	0.3329
0.36	0.4952	0.4986	0.4029
0.38	0.5413	0.5452	0.4385
0.40	0.5988	0.6061	0.4868
0.43	0.6891	0.6969	0.5508
0.46	0.7850	0.8214	0.6959
0.48	0.8759	0.9006	0.7613

strategy requires an adversary to hold $\approx 33\%$ stake to gain any advantage over honest mining (where relative revenue equals stake). Below this threshold, the MDP strategy actually performs *worse* than honest mining (e.g., at $\alpha = 0.25$, MDP revenue is $0.1929 < 0.25$). This confirms that blindly applying probabilistic strategies in a system with fixed lookahead is detrimental.

In contrast, with Lookahead-Aware Mining, an adversary with as little as 5% stake can achieve marginal gains, and an adversary with 20% stake achieves a relative revenue of $\approx 21.7\%$, a clear advantage over the honest expected return of 20%.

D. Computational Efficiency

The adoption of the $O(T \log T)$ Fast LAM algorithm is crucial for practical feasibility. For an epoch of $T = 100$, the optimal strategy is computed in negligible time. This efficiency enables the adversary to re-evaluate strategies instantly as the network state evolves, a capability not present in computationally intensive solvers like Deep Reinforcement Learning. Furthermore, the performance of LAM is extremely close to the Global Optimal UB, confirming that the greedy epoch-by-epoch application of LAM is sufficient for practical attacks.

VI. CONCLUSION

This paper investigates the impact of epoch-based randomness on the security of Proof-of-Stake blockchains. While previous analyses often assume per-slot unpredictability, we highlight that practical protocols (e.g., Ethereum, Cardano) inadvertently grant validators a "Loopahead Property" by fixing the leader schedule for an entire epoch.

We formalized this setting and introduced **Lookahead-Aware Mining (LAM)**, an optimal strategy for the single-epoch mining game. By mapping the problem to weighted interval scheduling, we derived a deterministic algorithm that allows an adversary to maximize the disruption of the honest chain. Our simulations confirm that LAM outperforms standard strategies derived from probabilistic models, enabling adversaries with as little as 5% stake to gain a disproportionate advantage.

These findings suggest that the trade-off between randomness efficiency (long epochs) and security (predictability) is sharper than previously understood. Practically, this implies that PoS protocols should either shorten the lookahead window (e.g., via faster VDFs or single-slot finality) or adopt incentive mechanisms that specifically penalize the withholding behaviors characteristic of interval attacks.

REFERENCES

[1] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," in *FC*, 2014, pp. 436–454.

[2] M. V. X. Ferreira and S. M. Weinberg, "Proof-of-stake mining games with perfect randomness," in *EC*, 2021, pp. 433–453.

[3] "Ethereum proof-of-stake (pos)," https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/, 2024.

[4] R. Sarenche, S. Nikova, and B. Preneel, "Deep selfish proposing in longest-chain proof-of-stake protocols," in *FC*, 2024, available at https://eprint.iacr.org/2024/622.

[5] A. Sapirshtein, Y. Sompolinsky, and A. Zohar, "Optimal selfish mining strategies in bitcoin," in *FC*, 2016, pp. 515–532.

[6] "RANDAO," https://github.com/randao/randao, 2019.

[7] D. Boneh, S. Eskandarian, L. Hanzlik, and N. Greco, "Single secret leader election," in *AFT*, 2020, pp. 12–24.