Selfish Mining Under General Stochastic Rewards

Maryam Bahrani
Ritual, Vancouver, BC, Canada

Michael Neuder
Ethereum Foundation, New York, NY, USA

S. Matthew Weinberg[1]
Princeton University, NJ, USA

Abstract

Selfish miners selectively withhold blocks to earn disproportionately high revenue. The vast majority of the selfish mining literature focuses exclusively on block rewards. [7] is a notable exception, observing that similar strategic behavior is profitable in a zero-block-reward regime (the endgame for Bitcoin's quadrennial halving schedule) if miners are compensated with transaction fees alone. Neither model fully captures miner incentives today. The block reward remains 3.125 BTC, yet some blocks yield significantly higher revenue. For example, congestion during the launch of the Babylon protocol in August 2024 caused transaction fees to spike from 0.14 BTC to 9.52 BTC, a 68× increase in fees within two blocks.

Our results are both practical and theoretical. Of practical interest, we study selfish mining profitability under a combined reward function that more accurately models miner incentives. This analysis enables us to make quantitative claims about protocol risk (e.g., the mining power at which a selfish strategy becomes profitable is reduced by 22% when optimizing over the combined reward function versus block rewards alone) and qualitative observations (e.g., a miner considering both block rewards and transaction fees will mine more or less aggressively respectively than if they cared about either alone). These practical results follow from our novel model and methodology, which constitute our theoretical contributions. We model general, time-accruing stochastic rewards in the Nakamoto Consensus Game, which requires explicit treatment of difficult adjustment and randomness; we characterize reward function structure through a set of properties (e.g., that rewards accrue only as a function of time since the parent block). We present a new methodology to analytically calculate expected selfish miner rewards under a broad class of stochastic reward functions and validate our method numerically by comparing it with the existing literature and simulating the combined reward sources directly.

2012 ACM Subject Classification Applied computing → Electronic commerce; Security and privacy → Distributed systems security

Keywords and phrases Proof-of-Work, Selfish Mining, MEV

Digital Object Identifier 10.4230/LIPIcs.AFT.2025.20

Related Version *Full Version*: https://arxiv.org/abs/2502.20360 [3]

Funding *S. Matthew Weinberg*: Supported by NSF CAREER Award CCF-1942497.

1 Introduction

Blockchain consensus mechanisms rely on incentives to coordinate behavior. To remain safe and live, crypto-economic systems require a majority (as in Proof-of-Work) or a super-majority (as in Proof-of-Stake) of participants to adopt the protocol-specified (sometimes referred to as "honest") actions. Selfish mining [10] first demonstrated that this honest behavior

[1] During Professor Weinberg's development of this paper, he participated as an expert witness on behalf of the State of Texas in ongoing litigation against Google (the "Google Litigation").

might not be incentive compatible for the rational miner who could earn a disproportionately large fraction of block rewards by selectively delaying the publication of their blocks. In the ensuing decade, a rich literature around strategic behavior in consensus protocols developed (e.g., in Ethereum Proof-of-Stake [27, 31, 24]). The vast majority of this literature focuses on strategies that optimize for the portion of the protocol-assigned rewards earned by the agent. These rewards, sometimes referred to as "protocol issuance" or "consensus rewards," have historically accounted for nearly all of the value in consensus participation; this is no longer true.

As modern blockchains gain usage and facilitate more significant economic activity, their decentralized applications generate revenue. Consensus participants can collect some of this revenue through the block producer's ability to arbitrarily re-order, insert, and delete transactions when they are elected leader; [8] introduces this concept as Miner/Maximal Extractable Value (abbr. MEV). MEV has been studied theoretically and measured empirically, leading to significant changes in blockchain design. Ethereum best exemplifies this, as over 90% of its blocks are built using a public, open-outcry block-building auction. The motivation for this auction is grounded in the notion of "fairness" of validator rewards. By creating a transparent market for buying and selling transaction orderings, each consensus participant should earn about the same amount of MEV – a principle originally encoded into consensus rewards, which are proportional to investment (measured in either work or stake).

A separate line of literature studies strategic behavior in decentralized finance (abbr. DeFi), which represents another source of rewards generated at the application layer. For example, loss-versus-rebalancing [21] (abbr. LVR) measures the amount of loss incurred by liquidity providers in decentralized exchanges as arbitrageurs balance the price of the decentralized exchange against an infinitely deep centralized exchange. These losses are precisely the profit available to those performing the arbitrage. This model completely abstracts the block creation and consensus processes, only considering the profits available to traders. In reality, the block producer has the final say over the transactions in their block, resulting in a large portion of this value flowing back to the consensus participants themselves.

The perspectives of the selfish mining, MEV, and DeFi literatures are incomplete in isolation. The co-mingling of revenue across the consensus and application layers necessitates a more precise model of rewards and their impact on strategic behavior, as demonstrated in the following real-world examples.

▶ **Example 1** (The launch of Bablyon)**.** On August 22, 2024, the Babylon [33] protocol launched on Bitcoin. The launch allowed `BTC` tokens to be "locked" through a transaction processed on the chain. With a cap of 1000 `BTC`, demand for transaction inclusion spiked as people rushed to be among the first to lock their tokens. This congestion led to a 68× increase in transaction fee revenue from 0.138 to 9.515 `BTC` between parent and child blocks `857909`, `857910`; over the four block range of `857908` to `857911`, the fee revenue increased by 500× from 0.031 to 15.551 `BTC` [19]. This immense growth in transaction fees persisted for only seven blocks, with an average per-block fee revenue of 9.64 `BTC`, after which the protocol reached its cap and fees returned to baseline levels. For those seven blocks, the block reward of 3.125 `BTC`, which normally represents nearly the entire source of miner revenue, was only 25% of the rewards claimed. Despite the limited scope of Bitcoin applications, Babylon exemplifies how non-protocol-specified rewards can dramatically distort miner incentives.

▶ **Example 2** (The "Low-Carb Crusader")**.** Proof-of-Stake differs from Proof-of-Work in that it requires stakers to explicitly lock up capital to participate in the system. While Proof-of-Work is limited only to incentivizing miners with positive rewards, Proof-of-Stake

enforces a subset of the protocol rules through the credible threat of destroying the capital owned by a misbehaving staker. Historically, this stick has served as an effective deterrent, but on April 2, 2023, an attacker referred to as the "Low-Carb Crusader" exploited a piece of infrastructure in the Ethereum protocol motivated by application layer-generated rewards. By tricking a server facilitating the block building auction referenced above, the attacker accessed private transaction data, which they exploited to 20 million `USD` of MEV [9]. In the Ethereum specification, this behavior violated the rules and thus was subject to a slashing penalty of 1 `ETH` (2600 USD at current prices) levied against the attacker's stake. Clearly, the consensus reward and penalty mechanism could not account for this magnitude of profit arising from the application layer. This example demonstrates the risk facing consensus mechanisms, where non-honest behaviors are incentivized with multi-million dollar exogenous rewards originating from the application layer.

These examples shows how the economic value generated in the application layer bleeds into the consensus layer rewards; see Appendix A.1 in the extended version of the paper [3] for a discussion on "timing games," which is another source of revenue for consensus participants (particularly in Proof-of-Stake).[2] To fully understand consensus incentives, a more general model for rewards is needed. In particular, a more accurate view of rewards would capture the aggregate incentives for following a specific strategy under many distinct revenue streams. The present work was motivated by that reality and takes the first step toward modeling general stochastic rewards in longest-chain protocols.

1.1 Related work

Combining the proportion of block rewards and the linear-in-time transaction fee models of [10] and [7] was the initial motivation for this work. We build upon their Markov Chains to analyze expected attacker rewards and study the β-cutoff strategies for selfish mining. As previously noted, neither work captures the state of the world in 2025; the fundamental question of "how vulnerable is Bitcoin to Selfish Mining now?" remains unanswered and of interest to the research community; this work seeks to address this gap in the literature. [37] demonstrates how large "whale transaction" fees in conjunction with the standard block rewards may result in attacker profitability at lower hashrates. They use reinforcement learning to approximate the optimal policy and profit for attackers. We also model these rewards as granting bonus value to blocks depending on the outcome of a Bernoulli trial. Our framework (Section 4) accommodates much more general rewards, and our instantiation (Section 5) includes a third source – linear-in-time transaction fees. Further, we analytically solve for the profitability of strategies rather than approximating them.

The selfish mining literature has grown extensively in the past dozen years; see [13] for a recent survey. [23, 30, 18] generalized the basic selfish mining strategy to broader strategy spaces. [34] studied the effect of the relative sizes of block rewards and transaction fees and the impact on miners' decisions on when to mine; [14] extended that analysis and show that if all miners are rationale, the equilibrium hash rates will be far below the maximal capacity. [5] demonstrated that longest chain Proof-of-Stake protocols would also be vulnerable to selfish mining – a result instantiated through numerous selfish strategies in various staking protocols: [27, 31, 24] in Ethereum, [12, 11] in Algorand's cryptographic self-selection, [25, 26] in Tezos. We extend our model of the Nakamoto Consensus Game from [4], which studies the detectability of selfish mining in Proof-of-Work.

[2] The appendix is excluded from the proceedings version of this paper due to the page limit – see the extended version [3] for the full appendices. Hereafter, we simply refer to the appendices directly without referencing the full version to reduce redundancy.

MEV is one of the most relevant topics existing blockchains are reckoning with; we focus how MEV impacts consensus mechanisms. [8] coined the term and introduced many of the key properties of MEV in permissionless systems. [36] systematized MEV strategies and proposed mitigations. [2, 6, 16] focused on the centralizing nature of MEV and how Ethereum's block building market is implemented through "Proposer-Builder Separation." [28, 32] studied timing games and their impact on consensus. [35, 29] empirically analyzed Ethereum block builders and how the market structure has evolved. We also draw on the DeFi literature when considering application-generated revenue for consensus participants. We focus on arbitrage profits as captured in LVR [21]. [20] extends the original model to capture trading fees.

1.2 Summary of results

We partition our results into two sets: practical and theoretical.

Despite the analysis of selfish mining under block rewards and transaction fees alone being several years old ([10, 7]), there remains a glaring hole in the literature to study selfish behavior under the combined rewards. Quoting from [13], a recent selfish mining SoK, "we find that only 3 works include transaction fees in their modeling; 2 consider both block rewards and transaction fees." As described in Section 1.1, [37] model transaction fees as "whale transactions" instead of linear-in-time; [15] approximate transaction fees using the average amount of time in each block; thus, they simply increase the size of the fixed block rewards. **Our first contribution is an analysis of selfish mining under the combined model of block and transaction fee rewards; this practical contribution helps paint a more accurate picture of the selfish mining in Bitcoin under a realistic aggregate reward function.**

Section 5.3 contains numerical results and discussion (see Figure 2) for the basic combination of fees and block rewards, along with other aggregate reward functions. Critically, as demonstrated in Figure 3, the protocol risk depends greatly on the reward model. For example, we show that the threshold at which an attack becomes profitable decreases by 22% when considering the two rewards together instead of only block rewards. Additionally, our plots allow us to make qualitative observations about miner behavior under different reward schemes. For example, a miner considering both block rewards and transaction fees will mine more or less aggressively, respectively, than if they cared about either alone. We confirm these analytical results through simulations (Figure 7 in the full version [3]) and by directly comparing them to existing literature (Appendices G and H in [3]).

To derive the aforementioned practical results, we develop a set of theoretical results that may be of independent interest. **We present (i) a model of the Nakamoto Consensus Game with general stochastic reward sources, (ii) a novel methodology to analytically solve for a selfish miner's rewards, and (iii) a natural set of reward function properties.** Section 2 describes the new structure we impose on the NCG and how general, time-accruing reward sources interact with difficulty adjustment, which we must explicitly account for. Further, unlike previous work,[3] our reward functions can be stochastic. Namely, we study a much more general class of *static* rewards (Definition 6), which we define as functions that accrue randomly and independently only as a function of time since the parent block. Calculating attacker profits under these general reward sources

[3] With the sole exception of [37], which studies a narrow set of random rewards – see Section 1.1 for discussion

requires the novel, path-counting technique presented in Section 4. The practical results (Section 5) described in the previous paragraph follow as a corollary since the combination of block and transaction fee rewards is static.

Lastly, we characterize a natural set of reward function properties motivated by existing blockchains (Section 3 and Appendix B in the full version [3]). We illustrate these properties through two extensive case studies. Section 3.1 examines transaction fees and describes how different assumptions about block size, transaction patience, and arrival rate manifest in very different reward functions captured by our properties. Appendix B.2 in the full version [3] focuses on arbitrager profits under various assumptions about price trajectories and leader-election mechanisms.

2 Preliminaries and model

We start by defining a stylized model of Proof-of-Work mining with general stochastic rewards. This necessitates some crucial differences between our model and previous selfish mining literature. For example, general rewards can be sensitive to specific inter-block times, requiring explicit modeling of difficulty adjustment. Section 2.2 discusses these differences in detail.

2.1 Nakamoto Consensus Game with general rewards

Let M denote the set of n miners, where miner $m \in M$ has hashrate α_m.

Views

At any time t, there is a public *view* V_t, consisting of the "state" of the blockchain known to all miners at time t. This view includes all blocks that have already been broadcast, their creation times, and the identity[4] of their creators in M. It also includes the content of each block, which contains enough information to compute the values of all variables and account balances in every block across forks. For each block B in a view, we have $\texttt{Timestamp}(B)$, the time[5] that the block was produced.

At any time t, there is also a private view V_t^m for each miner m that includes V_t and potentially some additional blocks m knows about that are unknown to all other miners (e.g., a private fork). We assume that miners don't selectively exclude a subset of miners when they broadcast, and all broadcasting happens instantaneously (e.g., no eclipse attacks [17]). As a result, V_t^m will only include V_t and any blocks mined by m that have not yet been broadcast (along with their contents).

General Rewards

Miners are rewarded for creating blocks on the eventual longest chain in the form of block rewards (a fixed value issued once per block), fees from included transactions, and potentially additional revenue stemming from their monopolistic control over the content of the block (MEV). The size of this reward can be different across blocks and might be stochastic. We abstractly model these rewards as a function R.

[4] Real-world blockchains are often pseudonymous, and the "identities" of miners refer to their public keys.
[5] Timestamp here refers to the actual creation time of the block, rather than a reported time stated by the miner.

Fix a time t, a view V, a block B in V, and a miner m. We use r to capture any exogenous randomness that could impact the value of blocks that a miner creates (e.g., the launch of a protocol that could create large amounts of congestion and resultingly higher transaction fees as in Example 1). We denote by $\mathcal{B}^m(t, V, B, r)$ the set of *valid* blocks that m can create. Because not all views are achievable under a specific realization of the randomness r, when we invoke a view V together with r, we implicitly restrict r such that V is realizable.

▶ **Definition 3** (Reward Function)**.** *A* reward function R^m *for miner* m *takes as input a time* t*, a view* V*, a block* B *in* V*, randomness* r*, as well as a block* $B' \in \mathcal{B}^m(t, V, B, r)$*, and outputs a real number,*

$$R^m(t, V, B, r, B') \to \mathbb{R}.$$

The output of R^m can be interpreted as the amount of reward collected by m for creating a block B' that extends B in V at time t given randomness r, assuming B' ends up on the eventual longest chain.

We allow different miners to have different reward functions to keep the model general. This per-miner reward can capture miner heterogeneity (e.g., from private order flow or better trading strategies). For the properties we define in Section 3 and the selfish mining analysis in Sections 4 and 5, however, we restrict our study to *miner-independent* (see Definition 4 below) reward functions.

Miner Strategies

Each miner m has a strategy that takes as input a time t, a view V_t^m, and the reward $R^m(t, V_t^m, B, r, B')$ for extending each block $B \in V_t^m$ by a valid block $B' \in \mathcal{B}^m(t, V_t^m, B, r)$, and outputs

- a block $B \in V_t^m$ to mine on,
- contents of the next block $B' \in \mathcal{B}^m(t, V_t^m, B, r)$, and
- a (potentially empty) subset of blocks in $V_t^m \setminus V_t$ to broadcast.

For each miner m, we denote by $\texttt{Next}(m, t, V_t^m, r)$ the first time after (or equal to) t that m broadcasts a block assuming their private view remains V_t^m, and by

$$\texttt{Next\_Broadcaster}(t, r) := \arg\min_{m \in M} \{\texttt{Next}(m, t, V_t^m, r)\},$$

the identity of the next miner to broadcast after (or at t), breaking ties arbitrarily. We use these functions to determine the ordering of broadcasters as the game progresses (see details in [3], Appendix – Algorithm 1).

Note that miner strategies cannot directly observe the randomness r but might indirectly depend on it through the realizations of R^m and $\mathcal{B}^m(t, V_t^m, B, r)$, all of which take as input the same randomness r. While we focus on deterministic miner strategies in this paper, our model can easily be extended to account for randomized behavior.

Nakamoto Consensus Game (NCG)

The Nakamoto Consensus Game describes how views evolve given a fixed set of miner strategies. We model the game after difficulty has already been adjusted according to these strategies, resulting in a stable orphan rate λ,[6] and we normalize time so that the average block time is 1. We let time 0 refer to a point after which the difficulty of mining puzzles remains constant. We further assume that miners only extend blocks created after time 0.

[6] See Section 2.2 for extended discussion and [22] for a more comprehensive overview of difficulty adjustment is used in the Bitcoin protocol.

Prior to the game, we draw the following random variables independently:[7]

- *Miner selection* – A sequence of miners $\vec{m} \in M^{\mathbb{N}}$, where m_i is the creator of the i^{th} block. For each i, m_i is selected independently such that it equals $m \in M$ with probability $\alpha_m / \sum_{j=1}^{n} \alpha_j$.
- *Block times* – A sequence of block creation times $\vec{t} \in \mathbb{R}^{\mathbb{N}}$, where $t_0 := 0$, and the duration $t_j - t_{j-1}$ for $j \geq 1$ is drawn i.i.d. from an exponential distribution with rate $1/(1-\lambda)$.
- *Remaining randomness* – The randomness r.

Initially, there is some public view V_0 but no hidden blocks, so $V_0^m = V_0$ for all $m \in M$, where $V_0 := \{B_0\}$ is the view containing a single genesis block B_0 such that $\texttt{Timestamp}(B_0) = 0$. Starting with $j = 1$ (the variable used to index the miners $\vec{m}$ and block times $\vec{t}$) and $t = 0$, we check if there are new blocks to broadcast before updating the block that each miner is building on based on the contents of the pre-determined strategy. See Appendix C.1 in [3] for the procedure to carry out the NCG and for a note on ensuring uniqueness of the longest chain.

2.2 Notes on model

We briefly summarize how we model difficulty adjustment below. See Appendix C.2 in [3] for an extended comparison of our model to previous work and the role of independence in the randomness of rewards.

Difficulty adjustment

In practice, mining involves solving computational puzzles with adjustable difficulty. Since miners can enter (or exit) permissionlessly, the total hashrate of all miners can vary over time, resulting in varying block production rates. The protocol varies the difficulty of these puzzles based on timestamps of recent blocks, targeting a fixed average inter-block time. In Bitcoin, the difficulty updates once every difficulty *epoch* (2016 blocks/roughly every two weeks assuming ten-minute block times) by the *difficulty adjustment algorithm (DAA)*. The difficulty of extending any blocks is the same within an epoch, except for forks across the epoch boundary. Note also that forks are rarely longer than a few blocks, so this represents an insignificant fraction of the blocks in an epoch.

Fixing a set of miner strategies, one can compute the expected fraction of blocks per epoch that do not end up on the longest chain. We assume the difficulty adjusts based on this expected value (rather than directly modeling per-epoch updates described above) and calculate the profitability of various strategies under this new difficulty. Specifically, we calculate the expected orphan rate λ (Lemma 16), which implies the difficulty-adjusted rate of block production is $1/(1-\lambda)$. This corresponds to blocks on the longest chain growing at an average rate of 1.

3 Reward functions: properties and examples

Recall that miner strategies take as input the amount of reward available for extending each existing block at time t, as specified by the reward function R, and make decisions about where to mine, what to include, and what to broadcast accordingly. This section defines a set of natural properties that reward functions might have. In Section 3.1, we

[7] See Appendix C.2 in [3] for a discussion of why we can assume independence.

motivate these properties with an extensive case study on transaction fees, one of the primary revenue sources observed empirically to date. See Appendices A.2, B, B.1, and B.2 in [3] for additional properties and a second case study on LVR, a prominent source of revenue on chains with significant DEX volume.

While we define these properties in the context of the NCG in this paper, we believe their applicability extends far beyond Proof-of-Work and selfish mining. Our framework can be used to characterize rewards and their implications for the incentives of consensus participants across blockchain protocols.

Recall that in the NCG, given a set of miner strategies, three independent random variables $\vec{t}, \vec{m}, r$ are drawn and used to compute a set of views V_t^m for all miners m and all times t. Let $\mathcal{V}_t^m$ be the support V_t^m, meaning the set of views achievable at time t for *some* realization of $\vec{t}, \vec{m}, r$. Initially, $\mathcal{V}_0^m = \{V_0\}$ for all m, where $V_0 := \{B_0\}$ is the view containing a single genesis block B_0 such that $\texttt{Timestamp}(B_0) = 0$. Miner strategies in the NCG take the realization of a reward function as input. That is, at time t, miner m sees the reward $R^m(t, V_t^m, B, r, B')$ for extending each block $B \in V_t^m$ by a valid block $B' \in \mathcal{B}^m(t, V_t^m, B, r)$.

A miner-independent reward function yields the same value for the block regardless of who created it. This corresponds to a setting where all miners have access to the same set of rewards (e.g., the common value setting), and thus, we drop the superscript m. In practice, some reward sources may be heterogeneous between block producers (e.g., from private order flow or from differing abilities to extract MEV [1]). All reward functions considered in this paper will be miner-independent, but the properties can be readily generalized by tracking the subset of miners with access to each reward source. See Section 6 for a discussion of extending this work.

▶ **Definition 4** (Miner-Independent Rewards). *A reward function R is* miner-independent *if for all times t, all miners have the same set of valid views, the same set of valid blocks extending each block in those views, and equal rewards from any such valid block.*[8] *Formally, R is miner-independent if for all t, and all $m, m' \in M$,*

- $\mathcal{V}_t^m = \mathcal{V}_t^{m'}$,
- *for all $V \in \mathcal{V}_t^m$, all blocks B in V, and all r, we have $\mathcal{B}^m(t, V, B, r) = \mathcal{B}^{m'}(t, V, B, r)$,*
- *for all $V \in \mathcal{V}_t^m$, all r, all parent blocks B in V, and all valid blocks $B' \in \mathcal{B}^m(t, V, B, r)$, we have $R^m(t, V, B, r, B') = R^{m'}(t, V, B, r, B')$.*

We can also characterize reward functions that grow according to the same distribution without depending on the chain's history. The following property limits the dependence of R on the view. Intuitively, it says that the only relevant information in the view that affects the amount of reward in a block is the *timestamp of its parent.*

▶ **Definition 5** (View-Independent Rewards). *A reward function R is* view-independent *if for all times $t' < t$, any two views $V_1, V_2 \in \mathcal{V}_{t'}$ such that $\texttt{Timestamp}(B_1) = \texttt{Timestamp}(B_2) = t'$ for some blocks $B_1 \in V_1, B_2 \in V_2$, we have:*

- *for all r, the set of valid blocks extending B_1 at t in V_1 is the same as the set of valid blocks extending B_2 at t in V_2, $\mathcal{B}(t, V_1, B_1, r) = \mathcal{B}(t, V_2, B_2, r)$,*[9] *and*

[8] Technically, since blocks include information about their creator, it would be more accurate to say that there is a bijection between the set of valid views/blocks for any pair of miners. We overlook this formality to simplify notation.

[9] Recall that when we invoke a view and randomness together as inputs to a function, we implicitly assume that the randomness could give rise to the view.

- *for every valid block* $B' \in \mathcal{B}(t, V_1, B_1, r)$*, we have*

$$\Pr_{r,\vec{t},\vec{m}|V_1}[R(t, V_1, B_1, r, B') = x] = \Pr_{r,\vec{t},\vec{m}|V_2}[R(t, V_2, B_2, r, B') = x]$$

  for all x*.*

Note that fixing a view V_1 (resp. V_2) can update the distribution of the $r, \vec{t}, \vec{m}$. We use the subscript $r, \vec{t}, \vec{m}|V_i$ to refer to the posterior distribution of these random variables conditioned on V_1, V_2. For example, block rewards are view-independent (within the same four year halving window) because each block earns the same fixed reward from the protocol. Example 7 below is a non-example that demonstrates how transaction fees that are not fully claimed by a parent block (e.g., arising from finite block sizes) are not view-independent because the reward of the resulting child block depends on the amount of unclaimed transaction fees.

View-independence already limits the dependence of R on the view to the timestamp of the parent block. We next define a subset of view-independent rewards where the dependence on view is limited to the length of *elapsed time since the parent block* (and is the same regardless of the exact parent block timestamp).

▶ **Definition 6** (Static Rewards). *A reward function* R *is* static *if for all* $\Delta > 0$*, all times* t_1, t_2 *and views* $V_1 \in \mathcal{V}_{t_1}$ *and* $V_2 \in \mathcal{V}_{t_2}$ *such that* $\texttt{Timestamp}(B_1) = t_1 - \Delta$ *and* $\texttt{Timestamp}(B_2) = t_2 - \Delta$*, we have:*

- *for all* r*, the set of valid blocks extending* B_1 *at* t_1 *in* V_1 *is the same as the set of valid blocks extending* B_2 *at* t_2 *in* V_2*,* $\mathcal{B}(t_1, V_1, B_1, r) = \mathcal{B}(t_2, V_2, B_2, r)$*, and*
- *for all valid blocks* $B' \in \mathcal{B}(t_1, V_1, B_1, r)$*, we have*

$$\Pr_{r,\vec{t},\vec{m}|V_1}[R(t_1, V_1, B_1, r, B') = x] = \Pr_{r,\vec{t},\vec{m}|V_2}[R(t_2, V_2, B_2, r, B') = x]$$

  for all x*.*

Example 9 below highlights that transaction fees are static using the [7] model with constant arrival rate and infinite block sizes. See appendix Examples 37 and 38 in [3], which demonstrate the conditions under which LVR is or is not static.

3.1 Properties of transaction fees

To illustrate the value of the aforementioned properties of reward functions, we perform an extensive case study on transaction fees (see Appendix B.2 in [3] for a similar study but on LVR). We consider the relevant properties that arise from different assumptions about block sizes, user patience levels, and accrual rate of transactions. These examples aim to justify the properties we focus on in Section 3 and motivate Sections 4 and 5, which measure attacker revenue under multiple static reward sources.

Transaction fees

Users pay transaction fees to interact with blockchains. A mempool collects transactions as they arrive, and its state at all times is captured in our model through the realization of the randomness r. Consider transactions as infinitely divisible,[10] belonging to the same

[10] We could instead consider transactions as heterogeneous in size (e.g., as in Ethereum where transactions consume different amounts of gas) or exclusive to miners (e.g., from private order flow), but the additional complexity doesn't add anything to the qualitative observations and is thus elided.

mempool,[10] and specifying a fee. A valid block B' mined at time t and extending a parent block B can include any transactions in the mempool at t that are not already included in $\texttt{Chain}(B)$. The corresponding reward function for a valid candidate block is the sum of the fees paid by the transactions it includes.

We call users *patient* if their transactions remain valid until they are eventually included in a later block. We shorthand transactions originating from patient users as *patient transactions*.

As demonstrated in the following example, we cannot claim any further structure on the patient-user transaction fee reward function without restricting the set of valid blocks.

▶ **Example 7** (Patient transaction fees may be view-dependent)**.** Consider two blocks B_1, B_2 with the same timestamp t' and with the same parent mined at t. B_1 claims all transaction fees arriving in $[t, t']$, while B_2 claims none. The rewards of maximizing candidate blocks B_1', B_2' built on B_1, B_2 respectively, are different, as B_2' can claim more transaction fees than B_1'.

The key observation is that miners may not claim the complete set of available transactions, thus impacting the claimable rewards of descendant blocks in that view (for more formalism, see Lemma 30 in the appendix of [3]). Alternatively, consider the case where each block can include all transactions (e.g., infinite block size as in [7]). If we additionally restrict the set of views for each miner $\mathcal{V}_{t'}^m$, we *can* make the following stronger claim.

▶ **Example 8** (Patient transaction fees are view-independent if blocks are infinite capacity and fully-claiming)**.** Assume blocks have infinite capacity and restrict views to only include blocks that contain all available transaction fees at the time of mining. Then, the distribution of rewards for B' built at time t on parent block B_1 or B_2, which have the same timestamp t', is the same. Namely, the reward is the sum of patient transaction fees arriving in the interval $[t', t]$.

Here, view-independence arises from the mempool fully emptying after each block is created. Thus, the reward function only depends on newly arriving transaction fees after the parent block is mined. Importantly, this reward function *may not be static* (which is a stronger condition than view independence) because the transaction fee arrival rate may not be homogeneous over time. For example, some hours of the day (such as trading hours in Asia time zones) might result in higher transaction fee arrivals. Assuming a constant transaction arrival rate, we can further establish staticness.

▶ **Example 9** ([7]'s model of transaction fees is static)**.** Assume 1 unit of patient transaction fees arrive per unit of time, blocks have infinite capacity, and all blocks in the view claim all available transaction fees (as in [7]). A block B' extending B at time $\texttt{Timestamp}(B) + \Delta$ can claim any reward in $[0, \Delta]$. Therefore, this reward function is static.

While the previous example considers *deterministic* transaction fee arrivals (1 unit of fees per unit of time), the same claim holds if the arrival rate is a random function of r (but still identically distributed over time). Constant accrual, in addition to the mempool clearing, results in the reward function being independent of the timestamp of the parent block, making it static.

Until now, we have only considered patient users. In contrast, consider *impatient users*, who submit transactions that are only valid for the next block produced (e.g., by checking the height of the block they are included in before executing). We similarly shorthand these as *impatient transactions*.

▶ **Example 10** (Identically distributed, impatient transaction fees are static)**.** Assuming the impatient transactions arrive according to a fixed distribution over time since the parent block, this reward function is static because the mempool clears after each block.

Note that the mempool clearing after each block was necessary for both Examples 9 and 10 to be static. However, the clearing came about differently – infinite block sizes in the former and impatient users in the latter. The mempool clearing is a *sufficient* condition for staticness if the distribution of rewards doesn't depend on global clock time.

Varying the assumptions on block size and user patience allows us to describe reward functions under differing models of congestion; we now consider transaction fees that are high regardless of the block size. This *contentious transaction* model is motivated by the launch of Babylon (Example 1). Transaction fees may spike because there is immense demand not just for inclusion in a block but also for a specific ordering (e.g., needing to be one of the first 100 transactions of a particular type).

▶ **Example 11** (Bernoulli rewards are static)**.** Consider contentious transaction fees modeled as independent Bernoulli trials that occur once per block height, resulting in a constant random reward of size E with probability p. This is a static reward function.

In Section 5, we study a variant of selfish mining under a combined reward function that includes Bernoulli rewards, linear-in-time transaction fees as in Example 9, and block rewards. This combined reward function is static, which is crucial to the tractability of that analysis. See Section 1.1 for a discussion on the similarities between our model of Bernoulli rewards and that of [37]. See Appendix B.1in [3] for examples pertaining to the properties defined in Appendix B and Appendix B.2 for an extended case study on LVR.

These examples showcase the properties we ascribe to general reward functions in Section 3. While these case studies allow us to demonstrate View-Independence (Definition 5) and Staticness (Definition 6) in familiar settings, they do not cover all MEV types. As mentioned in Section 6, we see characterizing the complete set of properties and applying them to other forms of MEV (e.g., sandwiches and liquidations) as a key direction for future work. With these properties in place, we now focus on calculating expected attacker profits from performing β-cutoff selfish mining strategies under general static reward functions.

4 Selfish mining with static rewards

Sections 2 and 3 presented our model of general stochastic rewards and created a structure around these reward functions. The subsequent sections study a specific set of miner strategies to analyze their profitability and feasibility under general static rewards (Definition 6). We examine $\beta-$cutoff selfish mining strategies [7], in which the attacker determines whether or not to hide their blocks based on the amount of reward realized during the mining process.

4.1 Mining strategies in the NCG

In the NCG defined in Section 2, miners make three decisions at each time t:

1. which block to extend,
2. the contents of their next mined block, and
3. which blocks to broadcast.

Based on these decisions, we define the protocol-prescribed mining as honest.

▶ **Definition 12** (Honest mining). *The honest mining strategy is defined as,*

1. *mine on the longest chain,*
2. *claim all available rewards, and*
3. *publish every block immediately.*

In words, the honest miners always follow the longest chain and immediately share any block they find with the rest of the network. If the remainder of the network is honest, the rewards that an honest miner, i, controlling α_i fraction of the hash power is proportional to their mining power.

[10] and [7] demonstrate that selfish mining is profitable for miners (even under various tie-breaking schemes) when considering *only* block rewards or *only* transaction fees that are linear-in-time respectively. [7] also introduced β-cutoff selfish mining strategies, in which the attacker mines selfishly as long as the rewards they earn on their hidden block are sufficiently small. If their rewards are larger than a threshold β, they instead broadcast immediately to avoid losing the valuable block.

▶ **Definition 13** (β-cutoff selfish mining [7]). *If there is no private chain, the attacker follows the rules:*

1. *mine on the public longest chain,*
2. *claim all available rewards, and*
3. *withhold any block found where the time since parent is less than β (create a private chain).*

The third step above creates the private chain for the attacker; they transition into the following rules (same as original selfish mining):

1. *mine on the private chain,*
2. *claim all available rewards, and*
3. *withhold any block found unless an honest block is found and the difference in length between the public chain and the private chain is ≤ 1.*

This strategy differs from pure selfish mining only in Step 3 under no private chain, where the attacker decides whether or not to publish based on the rewards captured in the block. Note that the strategies we consider claim all available rewards; miners could instead choose to intentionally leave some rewards on the table to incentivize subsequent miners to build on their chain ("undercutting" [7]). See Section 6 for discussion on extending our framework to a broader class of miner strategies.

Given a static reward function, we want to determine the per-unit-time expected attacker rewards from following the β-cutoff strategy as in Definition 13. We develop a new technique based on a Markov Chain similar to Figure 13 in [7] and Figure 1 in [10].

▶ **Definition 14** (β-cutoff Markov Chain). *Consider the NCG where the $1-\alpha$ of the mining power follows the honest strategy and α follows the β-cutoff strategy. Then define* `State i` *for $i \geq 1$ where the attacker has a hidden chain i blocks longer than the public chain. Let* `State 0` *denote the attacker having no hidden blocks and* `State 0'` *denote the race state between the honest and attacker forks each of length 1. Let* `State 0"` *denote the state immediately after the attacker publishes their private chain.*

Figure 1 depicts this Markov Chain. We now derive the transition probabilities using a general, static reward function. When considering static reward sources, notice that R is only a function of the time since the parent block was mined; we hereafter denote this static reward source as $R(t)$, where t is the time since the parent block. This simplification allows

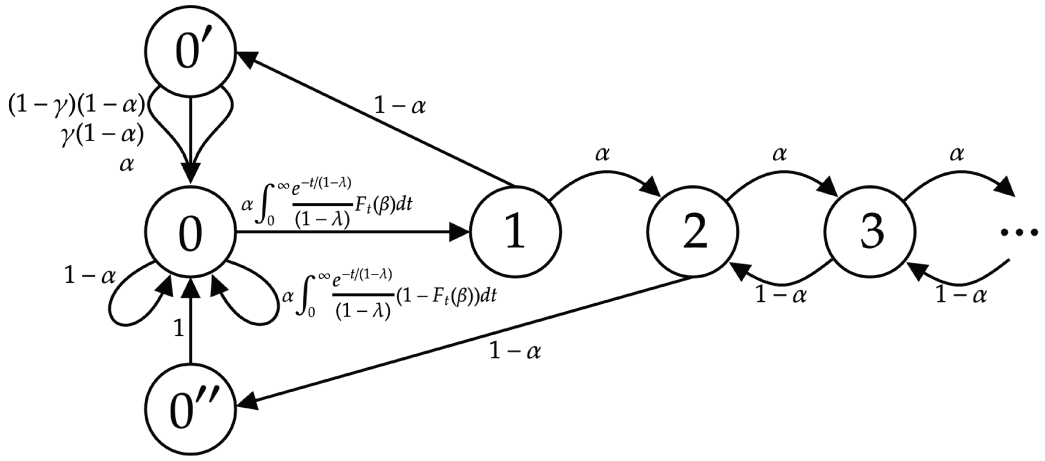

Figure 1 The Markov Chain capturing the $\beta-$cutoff strategy for miners deciding whether to publish blocks depending on the size of the static reward. $F_t(\beta)$ is the CDF of the rewards given time t since the parent block, $\Pr[R(t) \leq \beta]$. The rate of the chain is $1/(1-\lambda)$, which explicitly captures the difficulty adjustment that results from a specific β-cutoff strategy.

us to compute the probability of transitioning from `State 0` $\to$ `State 1` by comparing the expected amount of rewards earned in `State 0` conditioned on those rewards being less than β (the cutoff threshold for publishing the block in `State 0`).

▶ **Definition 15** (Static Reward CDF & PDF). *For a static reward source R and randomness r, let $F_t(x)$ denote the CDF of the reward function indexed by time t,*

$$F_t(x) = \Pr_r[R(t) \leq x].$$

Similarly, let $F'_t(x)$ denote the PDF of the reward function,

$$F'_t(x) = \Pr_r[R(t) = x].$$

To calculate the probability of withholding the block, we integrate the probability distribution of the time until the next block multiplied by the CDF of the rewards at each time.

$$\Pr[\texttt{State 0} \to \texttt{State 1}] = \alpha \int_0^\infty \underbrace{\frac{e^{-t/(1-\lambda)}}{(1-\lambda)}}_{\text{density of time}} \cdot \underbrace{F_t(\beta)}_{\substack{\text{rewards} < \beta \\ \text{by time } t}} dt \tag{1}$$

Intuitively, given a reward source R, this value tells us how likely it is that the rewards within an attacker block are less than β. Notice that the density function of the exponential depends on a rate parameter $1/(1-\lambda)$ (as discussed in Section 2.1), where λ the explicitly calculated orphan block rate calculated as a function of β to account for difficulty adjustment. See Lemma 16 for its derivation. Conversely, given an attacker block we can also calculate the probability that the attacker publishes the block immediately if the block rewards are be greater than β,

$$\Pr[\texttt{State 0} \to \texttt{State 0} \wedge \text{attacker block}] = \alpha \int_0^\infty \underbrace{\frac{e^{-t/(1-\lambda)}}{(1-\lambda)}}_{\text{density of time}} \cdot \underbrace{(1 - F_t(\beta))}_{\substack{\text{rewards} \geq \beta \\ \text{by time } t}} dt \tag{2}$$

With Equations (1) and (2), we construct the entire Markov chain in Figure 1. Note that it differs from Figure 1 in [10] and Figure 13 in [7], only in the transition probabilities from `State 0` calculated above for general static reward sources (Equations (1) and (2)). As in previous work, γ is the tie-breaking rate dictating the fraction of honest miners who mine on the attacker block after it is published, and there is a race of length-1 forks (in `State 1`). This parameter doesn't impact the β-cutoff itself and only affects the probability that the attacker fork wins the tie. See [3], Appendix E.1, for the calculation of the stationary distribution of this Markov Chain.

With the stationary distribution, we can explicitly solve for the proportion of orphan blocks, $\lambda \in [0, 1]$, which in turn gives us the difficulty-adjusted rate of the Poisson process of the transitions in the Markov Chain as $1/(1-\lambda)$. This rate is *faster* than the rate of canonical blocks (normalized to 1) because the orphaning process causes a reduction in difficulty.

▶ **Lemma 16** (Calculating λ)**.** *Let λ measure the probability that a block produced in the Markov Chain is orphaned. Then,*

$$\lambda = p_1(1-\alpha)\left(1 + \frac{\alpha}{1-2\alpha}\right).$$

Proof in Appendix D.1 of the full version [3]. With λ, the new block production rate is $1/(1-\lambda)$. This is the rate at which blocks are found by any miner (i.e., the rate of transitioning between states in the Markov Chain; Figure 1) assuming a constant hash rate and results in the canonical chain blocks being produced at a rate of 1.

4.2 Expected attacker rewards

The stationary distribution alone is incomplete. To determine the attacker profit for a given cutoff strategy, we calculate their expected profit from each state and multiply those values by the stationary distribution of the Markov Chain to determine the expected profit per unit of time.

▶ **Definition 17** (Per-state attacker rewards, f_i)**.** *Let f_i denote the expected reward of a canonicalized attacker block mined in `State i`.*

To calculate this value, we need to find the expected value of the reward function by integrating the time distribution over the possible paths that include an attacker block claiming rewards arriving during `State i`. We first enumerate all possible paths that result in a canonical attacker block from `State i`; we then integrate the reward function over each path. The following example demonstrates this technique, and we generalize it in [3], Lemma 40.

▶ **Example 18** (`State 3` paths)**.** Consider the rewards arriving after the attacker has a lead of length three. These rewards can be canonicalized in four different ways:

1. the attacker finds the next block, extending their lead to four,
2. the honest parties find the next block, then the attacker finds the subsequent,
3. the honest parties find the next two blocks, causing the attacker to publish their hidden chain, and then the attacker finds the first block after publishing,
4. the honest parties find the next two blocks, causing the attacker to publish their hidden chain, and then the honest parties find the first block after that.

We can succinctly represent these four outcomes using the strings, `A, HA, HHA, HHH`, where `H` & `A` denote honest and attacker blocks, respectively. This example prompts the definition of attacker paths.

▶ **Definition 19** (Attacker paths). *Given* `State i` *for all* $i \geq 2$*, there are* i *distinct paths resulting in the attacker capturing rewards accrued in that state. The paths are enumerated as the string* `(H*)A`*, where* `H` *&* `A` *denote honest and attacker blocks respectively and* `H` *is repeated* $0, 1, \ldots i-1$ *times.*

Continuing our `State 3` example, we now calculate the expected reward from each attacker path; adding these together is precisely the value of interest, f_3.

▶ **Example 20** (f_3 continued). Consider the three attacker paths of `State 3`: `A, HA, HHA`. These paths have lengths 1,2,3 and occur with probabilities $\alpha, (1-\alpha)\alpha, (1-\alpha)^2\alpha$, respectively. Thus, we calculate the expected reward as,

$$f_3 = \underbrace{\alpha \int_0^\infty \frac{e^{-t/(1-\lambda)}}{(1-\lambda)} \mathbb{E}_r[R(t)]\, dt}_{\texttt{A}} + \underbrace{(1-\alpha)\alpha \int_0^\infty \frac{te^{-t/(1-\lambda)}}{(1-\lambda)^2} \mathbb{E}_r[R(t)]\, dt}_{\texttt{HA}}$$
$$+ \underbrace{(1-\alpha)^2\alpha \int_0^\infty \frac{t^2 e^{-t/(1-\lambda)}}{2(1-\lambda)^3} \mathbb{E}_r[R(t)]\, dt}_{\texttt{HHA}}$$

Each of these expressions can be viewed as the product of three independent sources of randomness. The coefficients of the integrals are the probabilities of each path determined by the winning miner, which depends on $\vec{m}$. The first expression in the integrand is the PDF of the Erlang Distribution, which measures the sum of i.i.d. exponential random variables (all with rate $1/(1-\lambda)$) to determine the amount of time of the path, which depends on $\vec{t}$. The second expression in the integrand is the expected value over all remaining randomness, r, of the reward function at time t. See Appendix E.2 in [3] for the remaining f_i calculations. Combining the stationary distribution values, p_i, with the per-state expected rewards, f_i, we can calculate the full expected attacker reward.

▶ **Theorem 21.** *The attacker's expected reward is,*

$$\textit{ATTACKER REWARD} = f_0 p_0 + f_1 p_1 + \alpha \sum_{i=2}^{\infty} f_i p_{i-1}.$$

Proof. For `State 0` and `State 1`, we multiply the stationary distribution probability by the expected per-state attacker reward to calculate the contribution to the full attacker reward. For `State i`, $i \geq 2$, we need to avoid double counting the contributions from each state (e.g., you can transition to `State 3` from either `State 2` *or* `State 4`). To account for this we only consider the probability of arriving in each state from the $i-1$ state, which occurs with probability αp_{i-1}. Thus, for each state, we add the contribution to the total attacker reward as $\alpha f_i p_{i-1}$. The resulting value tells us the expected attacker reward per unit time of following a β-cutoff strategy under the static reward function and as a function of α, β, γ. ◀

5 Selfish mining with three reward sources

Selfish mining strategies were analyzed with *just* transaction fees and *just* block rewards in [10, 7], respectively. With the more general notion of miner rewards as defined in Section 2, a similarly general analysis is required to describe the profitability of selfish mining under

different reward schedules. The methodology of path counting and integrating the general reward function established in Section 4 works for any static reward functions. We now instantiate a specific aggregate reward function, which more accurately captures complete miner incentives as they exist in Bitcoin today. This combined reward function, which we denote $\hat{R}$, is composed of (1) a fixed block reward of size C, (2) a linear-in-time transaction fee reward, and (3) an "extra" reward of size E awarded to a block based on the outcome of a Bernoulli trial with probability p. Note that this new reward function considers the sum of each of these rewards, a more representative model of how miners are rewarded in reality rather than considering each of the rewards in isolation. For more straightforward examples of applying the path-counting technique to single-source reward functions, see Appendix H in [3] for only considering block rewards as in [10] and Appendix G for only considering transaction fees as in [7].

5.1 Rewards #1 & #2: block rewards and transaction fees

Each block that a miner produces earns a "fixed block reward" of magnitude C, which is paid directly to the miner as the first transaction in a block. We consider the block reward fixed.[11]

▶ Remark 22 (Block rewards are static). Block rewards are a constant function that doesn't depend on t,

$$R(t) = C. \tag{3}$$

As such, they are static because each block reward is identically distributed no matter the timestamp of the parent block.

The miners are also paid through the contents of the block they create. In particular, the transactions themselves specify a fee[12] to be paid to the miner for including the transaction in the block. As in [7], we start by assuming transaction fees arrive at a deterministic rate and are fully claimable by any subsequent block. See [34] for empirical measurements justifying the linear-in-time transaction fee rewards.

▶ Remark 23 (Deterministic transaction fees with fully claiming blocks are static). Using the [7] definition of fixed-rate transaction fee arrival, we have

$$R(t) = t. \tag{4}$$

This reward is static, as it is deterministic and the same for all blocks (only depending on timestamp of the parent block).

5.2 Rewards #3: non-deterministic extra rewards

We also introduce a third type of reward to our model, motivated by the reality that some blocks have much higher transaction fee revenue than others due to contention. [37] use a similar model to capture high-fee-paying transactions in addition to block rewards; see Section 1.1 for further discussion. Consider, for example, that a new type of transaction can

[11] The Bitcoin block reward is cut in half every four years, which impacts the relative size of the block reward compared to other reward sources. Our model considers the strategies available to miners within the same block reward period.

[12] In Bitcoin, the UTXO model defines a set of inputs and outputs for a transaction. Any balance that doesn't specify an output is claimable by the miner.

become available at a specific block height, and only a fixed amount of those transactions are valid (e.g., the first 10,000 transactions that purchase a specific NFT). To get their transaction included, participants submit bids specifying the fee they will pay to the block producer for higher-priority inclusion (assuming transactions are ordered by fee). This contention for block space leads to much higher revenue for the miner (who serves as the auctioneer) because even assuming infinite block sizes, the finite nature of the transaction type induces the competition (sometimes referred to as a "priority gas auction" [8]). We model this reward as a fixed size "extra reward" of magnitude E available to a miner of a block with probability p (a Bernoulli trial) and independent of time. We refer to this reward function as "Bernoulli rewards."

▶ Remark 24 (Bernoulli rewards are static). Bernoulli rewards are static because each block has the same distribution of rewards according to the outcome of the trial,

$$R(t) = \begin{cases} E & \text{if } X = 1 \\ 0 & \text{otherwise,} \end{cases} \quad \text{where } X \sim \text{Bernoulli}(p). \tag{5}$$

Note that this model doesn't allow for the "predictability" of these Bernoulli rewards. Since miners may know a priori what block height a new set of transactions will arrive at, miners' strategy space would be different than the standard selfish mining strategies we explore below. See Section 6 for more discussion.

▶ **Definition 25** (Reward function instantiation, $\hat{R}$)**.** *Combining the three reward sources (Equations* (3)*–*(5)*), we have the full reward function, which we denote as* $\hat{R}$*,*

$$\hat{R}(t) = C + t + E \cdot \mathbb{1}[X = 1],\ X \sim \textit{Bernoulli}(p). \tag{6}$$

Recall that the path-counting technique defined in Section 4 applies to any static reward function. Since $\hat{R}$ is the sum of three independent, static rewards sources, it is static itself, and thus, we can analyze it. Under $\hat{R}$, we seek to calculate the attacker reward (Theorem 21). Following the structure above, we define the Markov Chain as a function of $\hat{R}$, which induces a stationary distribution p_i before explicitly calculating the per-state attacker reward f_i. For the derivation of the stationary distribution, an instantiation of the general technique in Section 4.1, and the Markov chain under this combined reward function, see [3], Appendix E.3. For the derivation of the expected attacker rewards, an instantiation of the general technique in Section 4.2, see Appendix E.4 in [3].

5.3 Numerical results and discussion

Linear-in-time transaction fees and block rewards

Figure 2 analyzes the simple combination of the linear-in-time transaction fee rewards and block rewards. As expected, the attacker rewards under this combined function interpolates between the two extremes. `Selfish` (in red) shows the percentage of the block rewards collected when always hiding in `State 0` (which is exactly the reward in [10] – see Appendix H in [3] for the full derivation). $\beta-$`cutoff (linear)` (in blue) shows the percentage of the linear-in-time transaction fees collected on the attacker chain when choosing β to maximize this ratio (which is exactly the reward in [7] – see Appendix G in [3] for the full derivation). $\beta-$`cutoff (linear + block)` (in tan) shows the attacker's reward when considering both reward sources together. One interpretation of Figure 2 examines how different reward regimes can lead to dramatically different conclusions regarding the "risk of attack" a

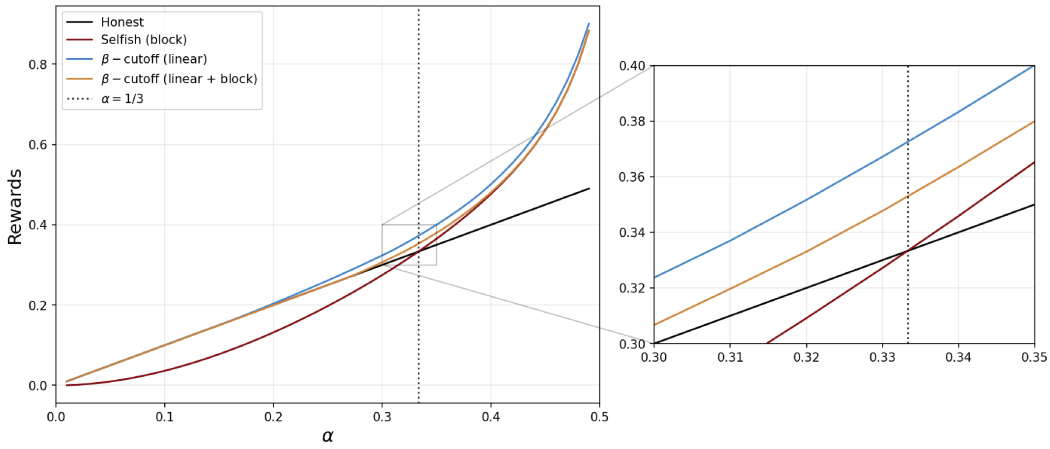

Figure 2 The attacker rewards as a function of α under different metrics of rewards. We consider miners who optimize for block rewards, linear-in-time rewards, or a combination of both. Considering both rewards together paints a more realistic picture of the protocol risk.

protocol faces. In this case, the selfish miner who only optimizes for the ratio of block rewards is not profitable until $\alpha = 1/3$. On the other hand, if we only consider the fraction of linear-in-time transaction fees capturable by a β-cutoff selfish miner, the story looks much worse. In particular, that miner becomes profitable around $\alpha = 0.15$. Considering both rewards results in a more measured conclusion, where the strategy becomes profitable around $\alpha = 0.25$. By varying the relative size of the block reward compared to the per-unit linear-in-time transaction fees, we can thus fully capture the dynamics of both reward models by interpolating between the two strategies, which consider the sub-rewards in isolation. Additionally, this figure can be interpreted qualitatively. We see that the attacker considering both rewards (tan) behaves *less aggressively* than the linear optimizing attacker (blue) for $\alpha \in [0.15, 0.25]$, as the optimal reward in that range is equivalent to honest. Conversely, for $\alpha \in [0.3, 0.33]$, a pure selfish mining strategy would not be profitable; thus, the attacker considering both rewards would be *more aggressive* than the block-reward maximizing miner (who would choose to mine honestly).

Profitability thresholds

Figure 3 shows the value of α at which various strategies become profitable under different reward sources as a function of γ. This extends Figure 3 of [10] to include more strategies. For each γ, we consider the optimal β cutoff for an attacker, maximizing block, linear, and total rewards, respectively. For each candidate α, we check if the optimal β results in a total reward that exceeds the benchmark of the honest performance under that reward function (i.e., the proportional block rewards from honest mining). We find the lowest candidate α such that the rewards exceed the benchmark and identify that as the profitability threshold. Intuitively, this is the fraction of the mining power needed to perform this strategy profitably.

For the pure selfish miner (in green), we see that the profitability thresholds of $1/3, 0.3, 0.25$ for $\gamma = 0, 0.25, 0.5$ are identical to [10]. When considering just linear and block rewards (in blue) and the total rewards (linear + block + bernoulli) (in pink), we see that for all values of γ, the profitability threshold decreases significantly. For example, at $\gamma = 0$, the profitability threshold is reduced from $1/3 \to 0.26 \to 0.18$ (reductions of 22% and 31% respectively) when considering the different reward sources. Similarly, at $\gamma = 0.5$, the profitability threshold is reduced from $0.25 \to 0.18 \to 0.09$ (reductions of 28% and 50% respectively).

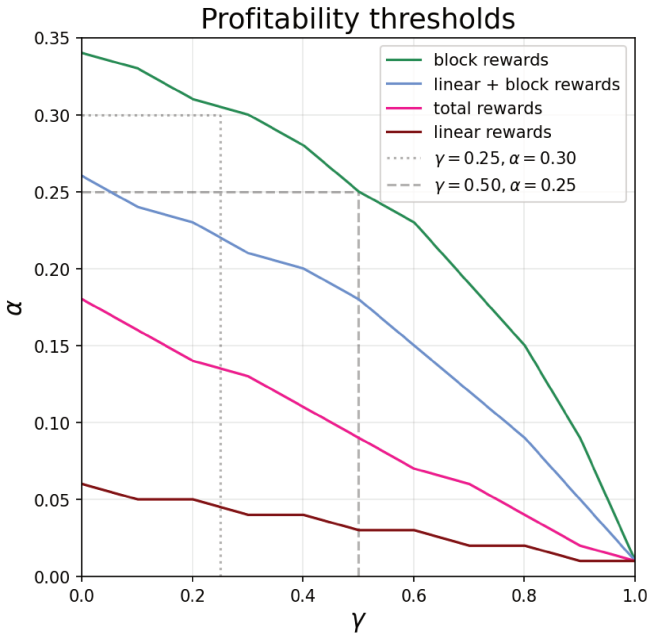

Figure 3 Demonstrating the α at which each strategy becomes profitable over honest as a function of γ. This extends Figure 3 from [10] to include more strategies. Each respective strategy considers profitability when only measuring a subset of the total rewards. For example, `linear + block rewards` (in blue) denotes a $\beta-$cutoff strategy for α profitable if, when selecting β to maximize the sum of linear and block rewards, the expected attacker reward exceeds 2α.

Interestingly, the attacker that only considers linear-in-time transaction fees (shown in red) is profitable (measured against just linear-in-time rewards) for nearly all values of α. While this may seem concerning, we believe that only linear-in-time rewards are not a large enough component of the current Bitcoin incentive structure to warrant concern. The aggregate view of the rewards (e.g., total shown in pink) more accurately represents rewards as they exist in Bitcoin today. These results only consider the case where all three components are approximately equal in magnitude. See Section 6 for a discussion on how choosing the relative sizes of these different reward sources based on empirical values would be a valuable application of our technique.

Measuring Bernoulli reward

Figure 4 examines the profitability of two other mining strategies: optimizing β for Bernoulli rewards (in green) versus optimizing β for the sum of linear, block, and Bernoulli rewards (in tan). Again, the combined rewards interpolate between the Bernoulli and the block-optimizing miners. For the miner maximizing over all three rewards, we normalize them each to have an expected value of 1 per block. The miner who only considers Bernoulli rewards (in green) is always profitable and significantly outperforms honest when $\alpha \geq 0.1$.

Bernoulli rewards aim to capture the volatility of MEV, which depends greatly on exogenous randomnesss (e.g., global capital market volatility). As the block reward continues on it's halving schedule and if transaction fees for Bitcoin transfers remain relatively low, MEV may come to dominate the miner incentives in the long-term. The fact that a miner becomes profitable under MEV rewards at such a low value of α may be a canary in the

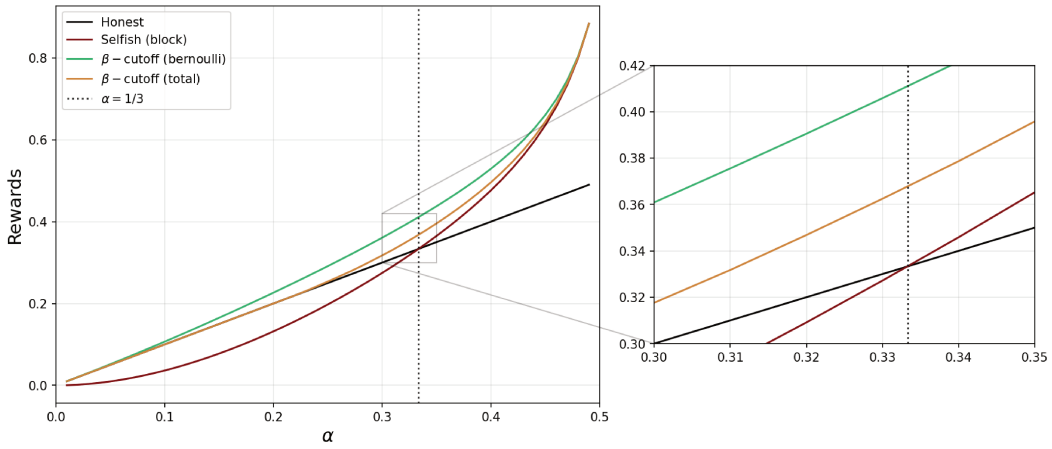

Figure 4 The attacker rewards as a function of α under different metrics of rewards. We consider miners who optimize for block rewards, Bernoulli rewards, and the full $\hat{R}$ containing block, Bernoulli, and linear rewards. Note that the Bernoulli reward-optimizing attacker is profitable for all values of α and meaningfully deviates from honest for $\alpha > 0.1$.

coal mine for the endgame analysis of selfish mining under MEV. We see understanding how Bitcoin MEV may play out under increased financial activity secured by the chain (e.g., through Bitcoin L2s that need to post data to the base chain itself) as one of the most important avenues for future work.

See Appendix F in [3] for more numerical examples and simulation results confirming the accuracy of the technique.

6 Conclusion and future work

We hope this work is a starting point for a more complete picture of participants' incentives in permissionless consensus mechanisms. **Our practical contributions of analyzing selfish mining under a more representative reward model in Section 5** fills a long-standing gap in the selfish mining literature. **Our modeling contribution of general rewards in the NCG and characterizing properties of observed reward sources in Sections 2 and 3** serves as the basis for a richer theoretical treatment of MEV and its implications for consensus. **Our methodological contribution of presenting a path-counting technique to calculate expected attacker profit under general static rewards in Section 4** provides a turn-key solution for future work to analyze selfish mining under newly identified reward sources. To conclude, we outline many potential future research directions.

Applying our methodology more broadly. Our reward instantiation in Section 5 is a reasonably realistic model of reward sources in the Bitcoin blockchain today. The methodology and instantiation represent a significant step in understanding the risk of selfish mining in the presence of multi-faceted rewards, especially since prior work generally considered one reward source at a time. However, empirical analysis may strengthen our results by forming a more nuanced understanding of these rewards in practice (e.g., measuring the relative size and probability of different MEV events). Note that our methodology still applies to any static reward sources that can be analytically calculated using the path-counting technique presented in Section 4. Beyond explicitly using our methodology, our technique has relatively straightforward extensions that can reach beyond static rewards and $\beta-$cutoff strategies.

Extending our methodology. There are several natural extensions to our methods. For example, considering the profitability of β-cutoff selfish mining under non-static reward functions is feasible. Such reward functions depend on additional information not captured in the states of the Markov Chain. However, suppose that the additional information is exogenous to the chain and independent of views. In that case, it is possible to augment the state space of the Markov Chain to include this information. Sme reward sources exhibit "periodic" behavior; in the case of Babylon (Example 1), elevated rewards persisted for a seven-block period. A simple modification of the Markov chain would allow the rewards to depend on whether the system was in a "high" versus "low" regime.

Another extension is to study MDP-based optimal strategies as in [30] rather than β-cutoff selfish mining. [37] demonstrate the impact of changing the reward function on optimal selfish mining profits when considering the combination of block rewards and occasional "whale" transactions, and they note that the resulting large state spaces were intractable with traditional MDP solving tooling and required machine learning. Considering how to more succinctly represent multi-reward state spaces or using the Deep RL approach with more combinatorial rewards are both promising directions. While we consider strategies that make decisions based on the realization of rewards in the *current* block, the broader MDP strategy space can be future-looking; for example, an attacker may want to start creating a hidden chain of several blocks in advance of an anticipated large reward (e.g., from an NFT drop occurring at a specific block height).

A complete picture of consensus incentives. As demonstrated in Examples 1 and 2, modern blockchains have faced and will continue to face distortion of consensus incentives from the application layer handling growing amounts of economic activity. Section 3 is a first step at modeling properties of general reward functions, but applying these properties to MEV beyond the transaction fee (Section 3.1) and LVR (Appendix B.2 in [3]) case studies remains open. It would be interesting to derive a set of necessary and sufficient properties to fully taxonomize MEV. Studying heterogeneity of reward sources was out of the scope of this work, but expanding the properties of reward functions when block producers have highly different rewards realizations is another key future direction.

References

1 Maryam Bahrani, Pranav Garimidi, and Tim Roughgarden. Centralization in block-building and proposer-builder separation. In *International Conference on Financial Cryptography and Data Security*, pages 331–349. Springer, 2024. doi:10.1007/978-3-031-78676-1_19.

2 Maryam Bahrani, Pranav Garimidi, and Tim Roughgarden. Transaction fee mechanism design with active block producers. In *International Conference on Financial Cryptography and Data Security*, pages 85–90. Springer, 2024. doi:10.1007/978-3-031-69231-4_6.

3 Maryam Bahrani, Michael Neuder, and S Matthew Weinberg. Selfish mining under general stochastic rewards. *arXiv preprint arXiv:2502.20360*, 2025. doi:10.48550/arXiv.2502.20360.

4 Maryam Bahrani and S Matthew Weinberg. Undetectable selfish mining. In *Proceedings of the 25th ACM Conference on Economics and Computation*, pages 1017–1044, 2024. doi:10.1145/3670865.3673485.

5 Jonah Brown-Cohen, Arvind Narayanan, Alexandros Psomas, and S Matthew Weinberg. Formal barriers to longest-chain proof-of-stake protocols. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, pages 459–473, 2019. doi:10.1145/3328526.3329567.

6 Agostino Capponi, Ruizhe Jia, and Sveinn Olafsson. Proposer-builder separation, payment for order flows, and centralization in blockchain. *Payment for Order Flows, and Centralization in Blockchain (February 12, 2024)*, 2024.

7 Miles Carlsten, Harry Kalodner, S Matthew Weinberg, and Arvind Narayanan. On the instability of bitcoin without the block reward. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 154–167, 2016. doi:10.1145/2976749.2978408.

8 Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 910–927, 2020. doi:10.1109/SP40000.2020.00040.

9 Francesco D'Amato and Michael Neuder. Equivocation attacks in mev-boost and epbs. https://ethresear.ch/t/equivocation-attacks-in-mev-boost-and-epbs/15338, 2023. Accessed: 2025-02-05.

10 Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. *Communications of the ACM*, 61(7):95–102, 2018. doi:10.1145/3212998.

11 Matheus VX Ferreira, Aadityan Ganesh, Jack Hourigan, Hannah Huh, S Matthew Weinberg, and Catherine Yu. Computing optimal manipulations in cryptographic self-selection proof-of-stake protocols. In *Proceedings of the 25th ACM Conference on Economics and Computation*, pages 676–702, 2024.

12 Matheus VX Ferreira, Ye Lin Sally Hahn, S Matthew Weinberg, and Catherine Yu. Optimal strategic mining against cryptographic self-selection in proof-of-stake. In *Proceedings of the 23rd ACM Conference on Economics and Computation*, pages 89–114, 2022.

13 Colin Finkbeiner, Mohamed E Najd, Julia Guskind, and Ghada Almashaqbeh. Sok: Time to be selfless?! demystifying the landscape of selfish mining strategies and models. *Cryptology ePrint Archive*, 2025.

14 Guy Goren and Alexander Spiegelman. Mind the mining. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, pages 475–487, 2019. doi:10.1145/3328526.3329566.

15 Cyril Grunspan and Ricardo Pérez-Marco. On profitability of selfish mining. *arXiv preprint arXiv:1805.08281*, 2018. arXiv:1805.08281.

16 Tivas Gupta, Mallesh M Pai, and Max Resnick. The centralizing effects of private order flow on proposer-builder separation. *arXiv preprint*, 2023. arXiv:2305.19150.

17 Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. Eclipse attacks on bitcoin's peer-to-peer network. In *24th USENIX security symposium (USENIX security 15)*, pages 129–144, 2015. URL: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/heilman.

18 Charlie Hou, Mingxun Zhou, Yan Ji, Phil Daian, Florian Tramer, Giulia Fanti, and Ari Juels. Squirrl: Automating attack analysis on blockchain incentive mechanisms with deep reinforcement learning. *arXiv preprint*, 2019. arXiv:1912.01798.

19 mempool.space. Bitcoin block 000000000000000000 25c7d9798e97c8f5d8502b03f4bd6b99c365991c5f03b, 2025. URL: https://mempool.space/block/000000000000000000025c7d9798e97c8f5d8502b03f4bd6b99c365991c5f03b.

20 Jason Milionis, Ciamac C Moallemi, and Tim Roughgarden. Automated market making and arbitrage profits in the presence of fees. *arXiv preprint arXiv:2305.14604*, 2023.

21 Jason Milionis, Ciamac C Moallemi, Tim Roughgarden, and Anthony Lee Zhang. Automated market making and loss-versus-rebalancing. *arXiv preprint arXiv:2208.06046*, 2022.

22 Arvind Narayanan. *Bitcoin and cryptocurrency technologies: a comprehensive introduction*. Princeton University Press, 2016.

23 Kartik Nayak, Srijan Kumar, Andrew Miller, and Elaine Shi. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 305–320. IEEE, 2016. doi:10.1109/EUROSP.2016.32.

24 Joachim Neu, Ertem Nusret Tas, and David Tse. Two more attacks on proof-of-stake ghost/ethereum. In *Proceedings of the 2022 ACM Workshop on Developments in Consensus*, pages 43–52, 2022. doi:10.1145/3560829.3563560.

25 Michael Neuder, Daniel J Moroz, Rithvik Rao, and David C Parkes. Selfish behavior in the tezos proof-of-stake protocol. *arXiv preprint arXiv:1912.02954*, 2019. arXiv:1912.02954.

26 Michael Neuder, Daniel J Moroz, Rithvik Rao, and David C Parkes. Defending against malicious reorgs in tezos proof-of-stake. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, pages 46–58, 2020. doi:10.1145/3419614.3423265.

27 Michael Neuder, Daniel J Moroz, Rithvik Rao, and David C Parkes. Low-cost attacks on ethereum 2.0 by sub-1/3 stakeholders. *arXiv preprint arXiv:2102.02247*, 2021. arXiv:2102.02247.

28 Burak Öz, Benjamin Kraner, Nicolò Vallarano, Bingle Stegmann Kruger, Florian Matthes, and Claudio Juan Tessone. Time moves faster when there is nothing you anticipate: The role of time in mev rewards. In *Proceedings of the 2023 Workshop on Decentralized Finance and Security*, pages 1–8, 2023. doi:10.1145/3605768.3623563.

29 Burak Öz, Danning Sui, Thomas Thiery, and Florian Matthes. Who wins ethereum block building auctions and why? *arXiv preprint arXiv:2407.13931*, 2024. doi:10.48550/arXiv.2407.13931.

30 Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. Optimal selfish mining strategies in bitcoin. In *Financial Cryptography and Data Security: 20th International Conference, FC 2016, Christ Church, Barbados, February 22–26, 2016, Revised Selected Papers 20*, pages 515–532. Springer, 2017. doi:10.1007/978-3-662-54970-4_30.

31 Caspar Schwarz-Schilling, Joachim Neu, Barnabé Monnot, Aditya Asgaonkar, Ertem Nusret Tas, and David Tse. Three attacks on proof-of-stake ethereum. In *International Conference on Financial Cryptography and Data Security*, pages 560–576. Springer, 2022. doi:10.1007/978-3-031-18283-9_28.

32 Caspar Schwarz-Schilling, Fahad Saleh, Thomas Thiery, Jennifer Pan, Nihar Shah, and Barnabé Monnot. Time is money: Strategic timing games in proof-of-stake protocols. *arXiv preprint arXiv:2305.09032*, 2023. doi:10.48550/arXiv.2305.09032.

33 Ertem Nusret Tas, David Tse, Fangyu Gai, Sreeram Kannan, Mohammad Ali Maddah-Ali, and Fisher Yu. Bitcoin-enhanced proof-of-stake security: Possibilities and impossibilities. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 126–145. IEEE, 2023. doi:10.1109/SP46215.2023.10179426.

34 Itay Tsabary and Ittay Eyal. The gap game. In *Proceedings of the 2018 ACM SIGSAC conference on Computer and Communications Security*, pages 713–728, 2018. doi:10.1145/3243734.3243737.

35 Sen Yang, Kartik Nayak, and Fan Zhang. Decentralization of ethereum's builder market. *arXiv preprint arXiv:2405.01329*, 2024. doi:10.48550/arXiv.2405.01329.

36 Sen Yang, Fan Zhang, Ken Huang, Xi Chen, Youwei Yang, and Feng Zhu. Sok: Mev countermeasures: Theory and practice. *arXiv preprint arXiv:2212.05111*, 2022. doi:10.48550/arXiv.2212.05111.

37 Roi Bar Zur, Ameer Abu-Hanna, Ittay Eyal, and Aviv Tamar. Werlman: To tackle whale (transactions), go deep (RL). In *44th IEEE Symposium on Security and Privacy, SP 2023, San Francisco, CA, USA, May 21-25, 2023*, pages 93–110. IEEE, 2023. doi:10.1109/SP46215.2023.10179444.