Latency Makes You Look Honest: Fully Undetectable Selfish Mining in Bitcoin

A

Thesis

Presented to
the faculty of the School of Engineering and Applied Science
University of Virginia

in partial fulfillment
of the requirements for the degree

Master of Science

by

Varun Vejalla

December 2025

APPROVAL SHEET

This

Thesis

is submitted in partial fulfillment of the requirements
for the degree of

Master of Science

Author: Varun Vejalla

This Thesis has been read and approved by the examing committee:

Advisor: Matheus Ferreira

Advisor:

Committee Member: Wei-Kai Lin

Committee Member: Jack Doerner

Committee Member:

Committee Member:

Committee Member:

Committee Member:

Accepted for the School of Engineering and Applied Science:

Jennifer L. West, School of Engineering and Applied Science

December 2025

Abstract

Eyal and Sirer [2014] showed that a strategic Bitcoin miner can increase their share of Bitcoin rewards by deviating from the intended protocol through a *selfish mining* strategy. However, their strategy is statistically detectable, as there are unusually long forks which a system with natural network delays wouldn't produce. The exchange rate hypothesis has been put in place to justify why Selfish Mining does not occur in practice. Under this hypothesis detectable deviations would harm the price of Bitcoin. While detectable attackers may get a higher payoff in terms of BTC, it may be a lower payoff in terms of USD.

Bahrani and Weinberg [2024] have demonstrated that selfish mining strategies that are undetectable when looking at the *shape* of the blockchain are available. We take this further by showing that even in the presence of global clocks and precise measurements of end-to-end network delays, there is an undetectable selfish mining strategy. We accomplish this by showing that if the adversary adds a small delay to all messages it sends or receives, a sufficiently large miner would obtain unfair profits. This is surprising since our strategy is passive in the sense that miners are not adaptive and commit to time to release or receive blocks.

We develop a selfish mining strategy that is provably statistically undetectable under a realistic network model: block generation follows a Poisson process, blocks propagate with pre-defined latency distributions, and forks are resolved arbitrarily by miners. We show that an attacker who controls more than one-third of the total hash power can strictly increase their long-run share of blocks in the main chain while producing a distribution of blocks that is indistinguishable from that of an honest miner with slightly worse latency. Moreover, no augmentation of per-block metadata (headers, timestamps, signatures, etc.) can prevent this advantage in our model. Our results suggest that protocol changes limited to block metadata are insufficient to eliminate profitable, undetectable selfish mining under realistic network delays.

Contents

1 Introduction

Bitcoin, introduced in 2008 by an individual or group under the pseudonym of Satoshi Nakamoto [3], is the first widely adopted decentralized cryptocurrency. Its main innovation is a mechanism by which a globally distributed network of mutually distrustful participants can agree on a single append-only ledger of transactions, the blockchain, without relying on any central authority. This problem, decentralized consensus, is made challenging by network latency, arbitrary message delays, and potentially adversarial participants.

1.1 Blockchain Structure

At a high-level, the blockchain is a public data structure that is hard to manipulate. It consists of a sequence of blocks. Each block contains a collection of transactions along with a cryptographic reference (a hash) to a preceding parent block. This hash pointer ensures that modifying a past block would require recomputing every subsequent block, making the ledger immutable (or at least, hard to change) under standard cryptographic assumptions. Because blocks reference earlier blocks, the overall structure formed by all blocks every created is a tree. The Bitcoin protocol says that the longest chain in this tree is the authoritative ledger. Specifically, nodes can adopt any longest valid chain they are aware of.

Although the longest chain isn't necessarily unique at any given moment, especially right after blocks get created, the protocol is designed so that, under honest behavior and mild synchronization assumptions, temporary divergences eventually resolve. A block will, with probability one, either appear in the longest chain indefinitely or eventually be permanently excluded from all longest chains. This is part of what creates the incentives in the system: miners aim to produce blocks that ultimately become part of the longest chain, as only those blocks yield rewards.

Note that even now, it is possible that there is a long hidden chain that a group of miners with a large portion of hash power and very bad latency has been maintaining since the creation of bitcoin. As such, we do not know for sure that the current visible longest chain is actually the longest chain.

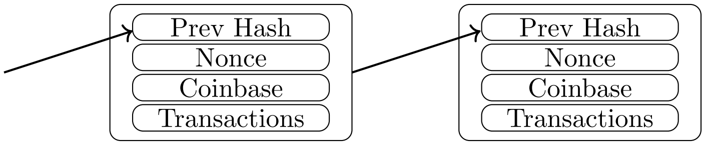


Figure 1: Connection between blocks. The hash of each block is included in the subsequent block.

1.2 Proof-of-Work and Block Creation

A central component of Bitcoin's decentralized consensus protocol is the proof-of-work (PoW) mechanism, which regulates the creation of new blocks and defends the system against adversarial attacks. Participants, called miners, compete to produce the next block by solving a computational problem that depends on how much work the miners put in. In the case of bitcoin and many other cryptocurrencies with a PoW mechanism, this is to find an integer nonce value such that evaluating a cryptographic hash function on that value along with the rest of the block header results in a small hash value. Formally, a block is considered valid if

$$\texttt{hash(block\_header)<target}.$$

Assuming that finding a valid nonce is as hard as if the hash function was some random black-box function, the only viable strategy for finding a valid nonce is brute-force search. In bitcoin, the target threshold is recalculated every 2016 blocks to ensure, on average, the network produces one block every ten minutes, regardless of the total computational power participating. If a target adjustment like this did not exist, blocks would be created much more frequently now than at the start of bitcoin in 2009 because more and more hash power is involved.

The randomized search results in a natural stochastic model for block creation. Each hash attempt is an independent Bernoulli trial with small success probability, and miners perform these trials continuously at rates proportional to their computational resources. In the limit, this process is well-approximated by a continuous-time Poisson process: If a miner controls a fraction α of the network's total hash power, then the times where they successfully find blocks form a Poisson process with rate $\alpha\lambda$, where λ is the global block creation rate. One big advantage of such a model is that the gaps between mining events follows an exponential distribution, which is memoryless.

1.3 Payoffs and Rewards

Miners in proof-of-work cryptocurrencies are incentivized to contribute computational power through block rewards. When a miner successfully creates a valid block, they also include a special transaction, called the coinbase transaction. This grants newly created cryptocurrency to the block's creator. In addition to that block creation reward, miners also collect any transaction fees associated with the transactions they include in the block. Throughout this work, we ignore transaction fees.

In an ideal setting, the block reward would only be valid if the block eventually becomes part of the longest chain. However, this is impossible to check or know ahead of time in a real-world system. For example, even now, all the created blocks in bitcoin could be invalid if there was a long hidden chain. Instead, Bitcoin implements a coinbase maturity role, which says the newly created coins in a block are not valid until at least 100 blocks have been built on top of it. This delay is good enough for practical purposes.

Thus, a miner's long-term payoff is determined not by how many blocks they produce in total, but how many of them end up in the longest chain. The maturity rule ensures miners are incentivized to extend the longest chain.

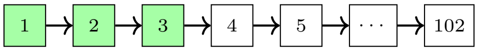


Figure 2: The coinbase maturity rule. Coinbase rewards become spendable only after 100 confirmations. In this example, the reward associated with block 1 is already spendable (green), while block 4's reward (white) is not.

1.4 Forks and Honest Miner Behavior

In the Bitcoin protocol, an honest miner follows two basic rules.

1. They always attempt to extend the longest valid chain they are aware of. If a unique longest chain exists, they mine on its deepest block. If multiple chains of equal length are visible, they may choose among them arbitrarily.

2. An honest miner immediately broadcasts any block they create, making it available to the rest of the network as soon as possible. Honest miners never withhold blocks strategically.

Even under fully honest behavior, the global blockchain does not necessarily remain a single chain at all times. If there is a nonzero latency by which information propagates across the network, two miners may create a block at nearly the same time, before either learns of the other's block. In such cases, both blocks reference the same parent, yielding a fork: two blocks at the same depth. This is an unavoidable property of decentralized block creation.

Forks are eventually resolved through continued mining. As soon as one branch receives additional blocks and becomes longer than the others, miners observing this will switch to the now-longest branch. Blocks on losing branches become stale. They do not contribute to the eventual canonical longest chain and their coinbase rewards never mature. These forks are especially important in the analysis of selfish mining attacks, as adversarial miners may attempt to exploit the same mechanisms to induce forks deliberately.

1.5 Selfish Mining

While the Bitcoin protocol is designed so that honest miners are rewarded for extending the longest chain, it is possible for a miner to increase their payoff by deviating strategically from honest behavior. A line of work beginning with Eyal and Sirer [2], and later extended in [5, 1], demonstrates that a miner with

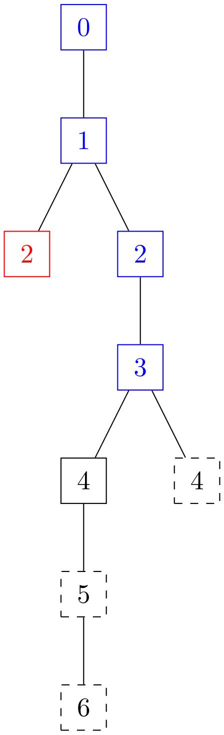


Figure 3: Example segment of the blockchain. Solid blocks are public; dashed blocks represent blocks not yet propagated to the entire network. Blue blocks lie on a longest chain if all miners are honest; red blocks cannot become part of any longest chain; black blocks may or may not end up in the eventual canonical chain, depending on how forks resolve.

sufficient computational power can obtain a disproportionate share of block rewards through selfish mining. Moreover, under stronger conditions, a miner with arbitrarily small computational power can also profit. In these strategies, the miner deliberately withholds blocks they have created and selectively reveals them only when doing so is advantageous.

By withholding blocks, the selfish miner can privately extend their own chain without immediately informing the rest of the network. When honest miners produce competing blocks, the selfish miner may then release some or all of their hidden blocks in order to force a fork or to cause honest miners to abandon their work. This behavior has two key effects:

1. Fork amplification. Strategic withholding increases the frequency of depths at which multiple competing blocks are created, thereby decreasing the expected growth rate of the public longest chain.
2. Biased fork resolution. When the selfish miner chooses the timing of block publication, they can arrange to win more fork races than their hash power alone would suggest. Over many rounds, this leads to a larger fraction of their blocks surviving into the eventual longest chain.

Since a miner's payoff is determined by the long-run fraction of blocks they contribute to the longest chain, these two effects can combine to produce a net profit: the miner may increase the numerator (blocks they win) and decrease the denominator (the chain's growth rate). Whether such a strategy is profitable depends on both the selfish miner's hash power and the behavior of honest miners - particularly how honest miners break ties when multiple longest chains are available.

1.6 Detecting Selfish Mining

The problem with many of these selfish mining attacks is that they are statistically detectable. There is some behavior in the blockchain that is not explainable by natural latency. For example, there may be more stale blocks than expected.

This has been proposed as an explanation for why selfish mining either does not happen or has simply not been detected. Even though a selfish miner may get more block rewards with some detectable selfish mining attack, the fact it's detectable may result in the value of the cryptocurrency itself decreasing. For example, they may get more bitcoin, but less USD.

However, this does not preclude undetectable selfish mining from happening. That is the main result of this paper. We show that under certain fairly reasonable conditions, a selfish miner with greater than 1/3 of the total hash power can achieve a profit in a fully undetectable way.

Of note is that in some PoW cryptocurrencies, the largest mining pools already have hash powers above 1/3 of the total hash power.

2 Related Work

Here, we talk about past selfish mining attacks and the effect of latency on Bitcoin in more detail.

2.1 Eyal and Sirer, 2013

Eyal and Sirer [2] introduced the first selfish mining attack in 2013. Their strategy has the selfish miners keeping the mined blocks private until the honest moners are just about to catch up again. Lifted almost verbatim from their paper, it is:

Algorithm 1 Selfish-Mine (Eyal & Sirer)

```
on Init
    public chain ← publickly known blocks
    private chain ← publicly known blocks
    $privateBranchLen \leftarrow 0$
    Mine at the head of the private chain.
on My pool finds a block
    $\Delta_{prev} \leftarrow$ |private chain| − |public chain|
    append new block to privateChain
    $privateBranchLen \leftarrow privateBranchLen + 1$
    if $\Delta_{prev} = 0$ and $privateBranchLen = 2$          ▷ Was tie with branch of length 1
        publish all of the private chain
        $privateBranchLen \leftarrow 0$
    Mine at the new head of the private chain
on Others found a block
    $\Delta_{prev} \leftarrow$ |private chain| − |public chain|
    append new block to public chain
    if $\Delta_{prev} = 0$ then
        private chain ← public chain                          ▷ They win
        $privateBranchLen \leftarrow 0$
    else if $\Delta_{prev} = 1$ then
        public last block of the private chain                ▷ Now same length. Try our luck
    else if $\Delta_{prev} = 2$ then
        publish all of the private chain                      ▷ Pool wins due to the lead of 1
        $privateBranchLen \leftarrow 0$
    else                                                      ▷ $\Delta_{prev} > 2$
        publish first unpublished block in private block
    Mine at the head of the private chain.
```

Eyal and Sirer make the assumption that a selfish miner is fairly well-connected. Because of this, when the selfish miner releases their blocks as needed, the selfish miner knows that the honest miners will at least consider those blocks. This results in the fork with the honest miners' blocks often being just behind the fork with the selfish miner's blocks. But since the selfish miner has less hash power than the honest miners, the honest miners will eventually catch up. Right when this happens, the selfish miner will publish the last block of their chain.

To analyze the payoff, they consider the Markov chain with states being the chain lengths. They then look at transitions and the long-term payoffs. Their main results are that a selfish miner that is infinitely well-connected can get a payoff higher than they should be getting (i.e. higher than their proportion of hash power) no matter what their hash power is when honest miners are using the first-seen rule as the tiebreaking rule. And even if they are poorly-connected, a selfish miner with at least 1/3 of the total hash power can be profitable.

Even a minor gain in payoff can be problematic. A rational miner that was previously honest might choose to join the selfish mining pool if they are able to. This can keep happening, leading to centralization in that mining pool, even though the original goal of bitcoin was to have a decentralized cryptocurrency.

Other works use Markov chains as well. However, using a Markov chain with countably many states is impossible for our problem. We assume continuous time, and the "selfish" miners will be acting fully consistently with honest miners with some latency, so there are uncountably many states. Although we could use a countable representation of time (e.g. a round-based model like in [1]), using continuous time is both more realistic and actually leads to easier analysis, albeit using different techniques.

2.2 Undetectable Selfish Mining

A natural critique of classical selfish mining is that it is statistically detectable. As implemented in [2] and other works, a selfish miner changes the distribution of stale blocks. The attacker's strategic withholding creates patterns of forks that are inconsistent with those produced by natural network latency. With natural network latency, the probabilities that the blocks at two subsequent heights are forked should be independent. However, with Eyal and Sirer's selfish mining attack, if there is a fork at some height, the next height is more likely to have a fork as well. This correlation makes it so that an observer can detect that selfish mining, or some other suspicious behavior, is happening, even if they may not be able to tell who exactly is doing it.

Bahrani and Weinberg [1] create a selfish mining attack that is designed to remove this correlation. Their goal is to produce a blockchain whose shape is indistinguishable from one with a network of only honest miners with higher latency. In their attack, the probability of each height being single or forked is independent of previous information. Thus, any statistical test based only on the tree structure of the blockchain (e.g. counts of stale blocks, correlations between fork events at different heights) simply cannot distinguish their attack from a world of only honest miners with worse network conditions.

Here, we only present the attack where all honest miners tiebreak in favor of the attacker. Their strategy is defined in two-steps: a labeling strategy decides whether a height should become a single height (only one block created) or a pair height (both attacker and honest miner create a block), and a broadcast strategy that implements those labels.

When a block at height $h-1$ gets created, the attacker computes the probability P_h that they will be the first miner to make a block at height h. Then the labeling strategy is:

- If the honest minrs make the first block at height h, label h as single.
- If the attacker makes the first block at height h, label it as pair with probability β/P_h, and single otherwise.

This makes it so that each height is independently a pair height with probability β.

The broadcasting strategy is:

- If h is labeled single and the attacker made a block at height h, broadcast that block immediately after the parent block is broadcast.
- If h is labeled pair, broadcast the attacker's block immediately after the honest block at that height is broadcast.

Although this attack is undetectable when considering only the shape of the blockchain, it is not undetectable when considering the times that blocks arrive. For example, in the pair heights, from the perspective of an outside observer, the two blocks would become visible at exactly the same time. Under any realistic network latency model, this should almost never happen. The attacker would be revealing blocks with timing patterns that are inconsistent with natural latency.

This distinction between tree-shape undetectability and full undetectability motivates this work, where we develop an attack that is indistinguishable from honest mining even when considering block arrival times.

2.3 Sakurai and Shudo, 2025

Although the other works and the rest of this work looked at adversarial manipulation, latency can influence miner profitability even when all miners behave honestly. Sakurai and Shudo [4] looked at mining fairness, which is the requirement that a miner's long-run share of block rewards is proportion to its computational power. Any deviation is called "The Rich Get Richer" (TRGR) effect and is harmful to decentralization.

Miners with higher hashrate get more proportional rewards, so a rational honest miner should join that centralized group. This would compound the effect, leading to a positive feedback loop.

They show the following:

1. A miner's profit rate increases linearly with its hashrate, with a slope dependent on propagation delays/latency.

2. As delays decrease, mining fairness improves. Larger delays amplify the TRGR effect.

3. The break-even threshold for profitability equals the sum of squares hashrate proportions, which grows as the hash power becomes more concentrated. Centralization makes it harder for small miners to remain profitable.

They also look at different tie-breaking rules like first-seen, random, and last-generated, but find that the main factor in mining fairness is block propagation delay and the the block generation time. Their work is relevant to ours because our attack is essentially just to simulate an honest miner.

Taken together, these works highlight three ideas: selfish mining can be profitable (Eyal and Sirer), selfish mining can in principle be disguised if one considers only the shape of the blockchain (Bahrani and Weinberg), and latency alone can distort mining fairness even in fully honest systems (Sakurai and Shudo). The present work is different in crucial ways.

1. We seek a selfish mining strategy that is fully undetectable, not only from the tree structure of the blockchain but also when block arrival times are taken into account. Bahrani and Weinberg's attack, while shape-undetectable, has detectable timing patterns.

2. We work in continuous time, which better reflects how Bitcoin actually operates and enables a more precise analysis of latency and block races. The two previously discussed works rely on round-based (discrete-time) models.

3. We have selfish miners that behave exactly like honest miners with increased latency. The profitability question reduces to analyzing how latency skews payoffs among otherwise honest miners. In our setting, the attacker can tiebreak however they would like in case of forks though.

3 Background

Although Markov chains are often defined only on finite or countable spaces, we consider Markov chains on arbitrary measure spaces. We consider Markov chains on $\mathbb{N}$ (countable) and $\mathbb{R}$ (uncountable, but a measure space via the Lebesgue measure).

3.1 Markov Chains on Measurable Spaces

Let $(X, \mathcal{F}, \mu)$ be a measure space. We abuse notation when the the σ-algebra nd measure are clear, and just say that X is the measure space. A sequence of random variables $(S_0, S_1, \ldots)$ taking values in X is called a *Markov chain* if it satisfies the *Markov property*:

$$\Pr\left[S_{n+1} \in A \mid S_n = x, S_{n-1}, \ldots, S_0\right] = \Pr\left[S_{n+1} \in A \mid S_n = x\right]$$

for all $n \geq 0$, all $x \in X$, and all measurable sets $A \in \mathcal{F}$. That is, the future evolution of the chain does not depend on the past, only on the current state.

The evolution of such a chain is specified by a transition kernel. A Markov kernel with equal source and target is a map

$$K : X \times \mathcal{F} \to [0, 1]$$

with the following properties:

1. For every fixed $F_0 \in \mathcal{F}$, the map $x \mapsto K(x, \mathcal{F}_0)$ is $\mathcal{F}$-measurable.
2. For every fixed $x_0 \in X$, the map $F \mapsto K(x_0, F)$ is a probability measure on $(X, \mathcal{F})$.

Put simply, there is a probability measure $K(dy|x) : F \mapsto K(x, F)$ on $(X, \mathcal{F})$ for each $x \in X$ such that for every measurable set $F \in \mathcal{F}$, the map $x \mapsto K(x, F)$ is measurable with respect to the measure of X. This describes the distribution of the next state given that the current state is x.

If the distribution of S_n is ψ, then the distribution of S_{n+1} is given by ψK, defined by

$$(\psi K)(A) = \int_X K(x, A) d\psi(x).$$

Iterating this relation yields $S_n \sim S_0 K^n$, where K^n is the n-step transition kernel.

Markov chains on measure spaces arise naturally in this work when considering the dynamics of systems where the states have real-valued parameters. We will often be considering the difference in mining times between the honest players and the selfish players as the state. The evolution of this quantity from one block height to the next is captured by a Markov kernel on $\mathbb{R}$. We also consider Markov chains on $\mathbb{N}$, which are often simpler.

3.2 Markov Kernels on $\mathbb{R}$ and Density Transitions

In many settings, including ours, the transition kernel admits a density with respect to the Lebesgue measure. That is, for each $x \in \mathbb{R}$, we have

$$K(x, dy) = k(x, y)dy$$

for some nonnegative function $k : \mathbb{R}^2 \to [0, \infty)$ that is a probability distribution for each $x \in \mathbb{R}$. In this case, the distribution of the next state is obtained by integrating against the density: if $X_n = x$, then X_{n+1} has density $y \mapsto k(x, y)$.

Working with kernels on $\mathbb{R}$ allows the model to incorporate continuous latency distributions and exponential block-generation times, neither of which can be represented nicely in a round-based model.

3.3 Stationary Distributions and Steady States

An important distribution when analyzing Markov chains, whether on finite, countable, or general measure spaces, is a stationary distribution. A probability measure π on the measure space X is called a stationary distribution for the Markov kernel K if

$$\pi = \pi K.$$

Equivalently, for every measurable set A, we have

$$\pi(A) = \int_X K(x, A) d\pi(x).$$

This essentially means that if we have some distribution giving the current state, then the next state also has the same distribution. A useful equation when considering the expected values of $\varphi(x)$, where x is drawn from the steady state, is

$$\int_X \varphi(y) d\pi(y) = \iint_{X^2} \varphi(y) K(x, y) d\pi(x) dy.$$

Stationary distributions are linked to the long-run behavior of a Markov chain. If $S_0 \sim \psi$ and $S_n \sim \psi K^n$, then any limit

$$\psi K^n \xrightarrow[n\to\infty]{} \pi$$

must be a stationary distribution if the limit exists. To be clear, this limit does not always exist, but when it does, it must be a stationary distribution.

Stationary distributions essentially capture the idea that the initial condition becomes less and less important. After many transitions, the distribution of the state only depends on the dynamics given by K, not on the starting point. In this paper, the stationary distribution of a certain real-valued variable will give the limiting distribution of the difference in mining times for each player for each height.

3.4 Long-Run Payoffs and Limiting Win Probabilities

Each miner's revenue is determined not by the total number of blocks they create, but by the number of their blocks that ultimately remain in the longest chain. Because forks occur, some blocks eventually become stale and earn no reward. We therefore define the payoff in terms of the longest chain.

Let:

- n be the total number of blocks created.
- D_n be the depth after n total blocks have been created.
- L_n^A be the total number of blocks created by miner A that are guaranteed to be in the longest chain.

Definition 1 (Payoff)**.** The payoff of miner A is

$$\mathcal{P}^A = \mathbb{E}\left[\liminf_{n\to\infty} \frac{L_n^A}{D_n}\right].$$

We make the following assumptions:

1. Given that we are currently in a state with all blocks public and no fork, the evolution of the game is independent of what happened previously. We cannot have some strategy that changes with time.
2. Given that we are currently in a state with all blocks public and no fork, the expected number of blocks created until we are in such a state again is finite.

Under these assumptions, consider rounds separated by those times when all blocks are public and there are no forks. We have

$$\frac{L_n^A}{n} \xrightarrow{n} A', \frac{D_n}{n} \xrightarrow{n} D'$$

almost surely, where A', D' are the respective expected values.

So under these assumptions, we have

$$\mathcal{P}^A = \mathbb{E}\left[\lim_{n\to\infty} \frac{L_n^A}{D_n}\right]$$

since the limit exists.

If there were no forks, the payoff would just be the proportion of total blocks they create. Forks complicate this because they do not contribute to the depth. To analyze the payoff, it's useful to work with respect to the depth rather than the total number of created blocks.

For each depth $d \geq 1$, define the indicator

$$I_d^A = \mathbf{1}\{\text{miner } A \text{ wins depth } d\}$$

and

$$P_d^A = \mathbb{E}\left[I_d^A\right] = \Pr[A \text{ wins depth } d].$$

If the longest chain has depth D, then the number of blocks won by player A would be

$$\sum_{d=1}^{D} I_d^A.$$

Note that we can consider any subsequence in the payoff since the limit exists. The depth-indexed payoff is then

$$\mathbb{E}\left[\lim_D \frac{1}{D}\sum_{d=1}^{D} I_d^A\right] = \lim_D \frac{1}{D}\sum_{d=1}^{D} P_d^A.$$

Here we use the previous assumptions and linearity of expectation to justify this.

If $\lim_d P_d^A$ exists, then $\mathcal{P}^A = \lim_d P_d^A$.

4 Model

The goal of this work is to look at fully undetectable selfish mining attacks. In order to do that, we first formalize what "undetectable" means and then describe our model.

4.1 Undetectability with Latency Simulation

A selfish mining strategy is called undetectable if the sequence of events it generates is statistically indistuignshable from a world in which all miners are honest but possibly experience higher network delays.

Definition 2 (Undetectability)**.** A strategy for miner S is undetectable if, for every algorithm that has real-time access to the public blockchain, the distribution of the mining game is identical to a distribution generated by a system in which all miners follow the honest protocol, except that their network latencies may be larger than in the base model.

In the model of Bahrani and Weinberg [1], the "detector" algorithms did not have real-time access. Instead, they could only look at the shape of the blockchain. With our model, we have statistics like:

1. Independence of the gap in arrival times between subsequent blocks
2. The gap in arrival times for fork blocks
3. If miners are required to include timestamps, the latency can be identified.

Although there may be a broader class of undetectable attacks available, the attack we will consider for the selfish miner is just to mimic an honest miner with higher latency. We define the behavior of honest miners under arbitrary latency distributions, and then define selfish behavior as the behavior of an honest miner with strategically chosen latency distribution. In this way, all deviations are fully explainable as network delays.

The rest of this model explains how miners behave under latency, how blocks propagate, and how chain selection occurs. In this section we formalize the network assumptions and assumptions about tie-breaking that we use in our analysis. The model specifies:

1. How blocks are created
2. When miners learn about different blocks
3. How miners select which chain to extend

In order to simplify this, we treat the honest miners as a group of *atomic* miners. That is, we assume they are a group of infinite miners each with infinitesimal hash power, but with non-infinitesimal hash power altogether.

4.2 Players, Hash Power, and Block Creation

We consider two miners: a (potentially) selfish miner S and an honest miner H. The selfish miner control a fraction $\alpha \in (0, 1)$ of the total hash power, while the honest miner has the remaining $1 - \alpha$ of the hash power.

We model block creation in continuous time. Each miner generates blocks according to a Poisson process with rates given by their respective hash powers. Then the inter-arrival times

$$T_S \sim \mathrm{Exp}(\alpha), \qquad T_H \sim \mathrm{Exp}(1 - \alpha)$$

are exponentially distributed and independent for each block. Overall, a block is created according to Exp(1). That is, a "unit" of time is normalized to be the block creation rate. In Bitcoin, this corresponds to 10 minutes.

When a miner is chosen to create a block, they make it according to a chain they think is longest.

4.3 Latency and Block Propagation

When a block is created by one miner, it is not necessarily learned immediately by the other. We model block propagation delays using nonnegative random variables:

$$L_{AB} : \text{delay for a block created by } A \text{ to reach } B.$$

These delays are assumed to have support in $[0, \infty)$ and to be drawn independently for each block. Note that L_{SH} and L_{HS} do not necessarily have to have the same distribution, although we may restrict that as well.

Suppose we are in the case of $L_{SH} = L_{HS} = L_{HH} = 0$ almost surely. That is, there is no real-world latency. The objective of the selfish miner is to simulate some latency L_{SH}, L_{HS}. The strategy for them to do so is simple. When receiving a block at time t, they will ignore it until time ℓ_{HS} passes, where $\ell_{HS} \sim L_{HS}$. At the end of that time, they will update their view. When making a block at time t, they will make it honestly according to their view. But they will wait until $\ell_{SH} \sim L_{SH}$ time passes before broadcasting it to the honest pool. This attack is entirely indistinguishable from an attack with the latency they are trying to simulate.

4.4 Chain Selection and Tie-Breaking

Miners maintain a local view of all blocks they have received. Their rules are as follows:

1. Extend a longest visible chain. A miner always attempts to mine on a chain of longest length in its local view.
2. Tiebreaking.
   (a) The selfish miner will always choose their own fork in case of ties.
   (b) The honest miner can choose whichever fork they like. Although there are infinitely many allowed policies, we say that they choose the selfish miner's fork $\gamma \in [0, 1]$ fraction of the time.

4.5 Orphan Blocks

Because latency may be probabilistic, it is entirely possible for a miner to receive a block whose parent/ancestor has not yet arrived. There are two natural ways to model this:

1. Immediate acceptance (used most of the time in our analysis). A miner treats a block as genuine even if its parent has not yet arrived. This simplifies our analysis.
2. Parent-first/waiting (the results in the low-latency regime extend to this model). A miner waits until missing ancestors arrive before accepting the block. Most of our methodology also extends to this model, although we do not consider it in as much detail.

4.6 Summary

- Mining occurs in continuous time with exponential gaps in block mining.
- There is a nonnegative random latency distribution for each ordered pair of miners.
- We restrict to two miners.
- Honest miners broadcast immediately, while the selfish miner broadcasts according to L_{SH}.
- Miners follow the longest-chain rule with their respective tie-breaking rules.
- We generally use the immediate-acceptance model when missing parent blocks.

This fully specifies the behavior of miners and the network. The mining game can be expressed only by the latency distributions, hash powers, and tie-breaking γ parameters. Thus, this attack is undetectable.

5 Results

This section presents our main theoretical contributions. There are two complementary lines of analysis:

1. Low-latency first-order payoff analysis. This involves a small perturbation around zero latency, although it is not particularly useful for finding a closed form.

2. General analysis using a Markov kernel and steady-state distribution. Although it is valid for arbitrary latency distributions, it is hard to analyze what happens in the low-latency regime.

Each part is self-contained, and together, they give a good understanding of undetectable selfish mining profitability.

5.1 Low-Latency Behavior and First-Order Profitability

The goal here is to analyze the regime where latency is small and get a first-order expansion (with respect to latency) of the payoff. This gives a simple sufficient condition for profitability. Specifically, we are trying to find $\mathcal{P}^S(L) \approx \alpha + \frac{d\mathcal{P}^S}{dL}|_{L=0}L$, where the constant-term is α because that is the payoff when there is no latency. Here, L is just some measure of the overall latency in the system. A selfish miner can be profitable with arbitrarily small latency if and only if $\frac{d\mathcal{P}^S}{dL}|_{L=0} > 0$.

Theorem 1. *A selfish miner can be profitable when $\alpha > \frac{1-\gamma}{2-\gamma}$, where γ is the probability that the honest miner mining on the selfish fork in case of the honest miner observing a tie. Moreover, they can be profitable even if the latency they are simulating is limited to be symmetric, deterministic, and bounded.*

The overall idea is that we can consider the game in terms of rounds where all blocks are public again. Because latency is assumed to be bounded, we will almost surely have a time in the future when all blocks are public again (e.g. when the gap between mining times is greater than this bound). We can set up a game that has the same first-order behavior and consider the number of blocks that were created before blocks were public again.

5.1.1 Simplified Race

Let $a_{s,h}$ denote the probability that, given all blocks are currently public, exactly s selfish blocks and h honest blocks are made before the next time that all blocks are public again and at least one block has been made. For small L, only the events with $s + h \leq 2$ contribute to the first-order term. So we restrict our attention mainly to the outcomes

$$(1, 0), (0, 1), (2, 0), (1, 1), (0, 2).$$

When $L = 0$,

$$a_{1,0} = \alpha, \qquad a_{0,1} = 1 - \alpha, \qquad a_{s,h} = 0 \text{ for all } s + h \geq 2.$$

Define derivatives $a'_{s,h} = \frac{d}{dL}\big|_{L=0} a_{s,h}$.

5.1.2 Events with Three or More Blocks

As $s + h$ grows, it becomes more complicated to say what happens. In fact, even just from $a_{2,1}$ and $a_{1,2}$, we do not have enough information to say who wins the current fork and what the new fork will be. But we have the following to simplify it:

Lemma 2. $\sum_{s+h\geq 3} a_{s,h}(1 + s + h) = O(L^2)$.

Proof. Once the first block gets created, we require at least two more blocks to get made in a time window that is on the order of L. But gaps in mining times follow exponential distributions. The probability of both getting made in $O(L)$ time is $O(L^2)$. □ □

So we can assign these blocks without worrying about distorting the first-order terms. We just give them all to the honest miner.

Pr	S gain	H gain	New state
$a_{1,0}$	$n+1$	0	0
$a_{0,1}$	0	$n+1$	0
$a_{2,0}$	$n+2$	0	0
$\gamma a_{1,1}$	$n+1$	0	0
$v := (1-\gamma)a_{1,1}$	0	0	$n+1$
$a_{0,2}$	0	$n+2$	0
$a_{s,h},\ s+h \geq 3$	0	$n+h$	0

5.1.3 Induced Markov Chain

Let `fork(k)` denote the state where there is currently a public fork with k blocks on each one.

Suppose we are currently in the `fork(n)` state. Then we have Table ?? giving the transition probabilities, gains, and new state for each player.

Of note here are the entries with $a_{1,1}$. These are the entries that make the fork grow. With some probability, the honest miner will then choose the selfish miner's fork, while with remaining probability, the two players keep fighting, but with one more block at stake.

In order to simplify the calculations and make them stateless, we can then group the process into "resolving rounds"

$$\texttt{fork(0)} \to \texttt{fork(1)} \to \cdots \to \texttt{fork(n)} \to \texttt{fork(0)}.$$

In each of these larger rounds, the fork is getting resolved. Note that the fork length only grows in a single case, and it grows by 1 every one of those times. n is then geometric with parameter

$$v = (1-\gamma)a_{1,1}.$$

The expected fork is given by

$$N = \frac{1}{1-v} - 1.$$

Recall that the limiting value for the ratio of counting random variables $S, S+H$ is given by the ratio of the expected values of their increments.

then have

$$\begin{aligned}\frac{S}{S+H} &\approx \frac{a_{1,0}(N+1) + a_{2,0}(N+2) + \gamma a_{1,1}(N+1)}{(a_{1,0}+a_{0,1}+\gamma a_{1,1})(N+1) + (a_{2,0}+a_{0,2})(N+2)} \\ &= \frac{a_{1,0}(N+1) + a_{2,0}(N+2) + \gamma a_{1,1}(N+1)}{(a_{1,0}+a_{0,1}+\gamma a_{1,1})(N+1) + (a_{2,0}+a_{0,2})(N+2)}\end{aligned}$$

To a first-order approximation in L (which corresponds to a first-order approximation in $a_{1,1}$), we have $N \approx (1-\gamma)a_{1,1}$. We can go through the (somewhat tedious) steps and find that

$$\mathcal{P}^S \approx \alpha + \mathcal{P}_1 L + O(L^2), \qquad \mathcal{P}_1 = \gamma(1-\alpha)a'_{1,1} + (1-\alpha)(a'_{1,0} + 2a'_{2,0}) - \alpha(a'_{0,1} + 2a'_{0,2}) \tag{1}$$

We require $\mathcal{P}_1$ to be positive in order to be profitable near zero latency. Up to now, the results hold for any setting in which we consider forking or block withholding as a proxy of latency, and where the probability of creating three or more blocks until all blocks are public is on the order of $O(L^2)$. Notably, this also applies to the setting considered by Bahrani and Weinberg. They came up with an attack that has $\mathcal{P}_1 > 0$ even when $\gamma = 0$ (i.e. when the honest miners are never choosing the selfish fork) and $\alpha < 1/2$. Under this restriction, we would need

$$(1-\alpha)(a'_{1,0} + 2a'_{2,0}) > \alpha(a'_{0,1} + 2a'_{0,2}).$$

Unfortunately, we do not get as strong of a result as them. Where we differ is by the coupling of these derivative terms to each other. For example, if the selfish miner wants to manipulate $a'_{1,0}$, they also end up manipulating $a'_{2,0}$ in a very constrained way.

5.1.4 Derivative terms with given latency distributions

Because $a'_{2,0} = a'_{2,0} + \frac{d}{dL}\big|_{L=0} \sum_{s\geq 2, s+h\geq 3} a_{s,h}$, we consider S_2 as the probability that the selfish miner makes the first two blocks and makes the second one before the honest miner sees the first one. Similarly, define H_2 as the probability that the honest miner makes the first two blocks and makes the second one before the selfish miner sees the first one. Define T_2 as the probability that the first two are one by each player, and both make them before seeing the other one. For these three events, we do not care about what happens afterward since it simplifies the expressions and is not needed for first-order analysis.

We have the following expressions that arise due to the exponential nature of the mining game:

$$a_{1,0} = \alpha\, \mathbb{E}[\exp(-L_{SH})], \qquad a_{0,1} = (1-\alpha)\, \mathbb{E}[\exp(-L_{HS})] \tag{1}$$

$$S_2 = \alpha^2 \left(1 - \mathbb{E}[\exp(-L_{SH})]\right), \quad H_2 = (1-\alpha)^2 \left(1 - \mathbb{E}[\exp(-L_{HS})]\right) \tag{2}$$

$$T_2 = \alpha(1-\alpha) \left(2 - \mathbb{E}[\exp(-L_{SH})] - \mathbb{E}[\exp(-L_{HS})]\right) \tag{3}$$

I will only explain S_2 and T_2 for conciseness. S_2 is given by the probability that the first two mining events are the selfish miner, and the second one is close to the first. The probability of the first two mining events being the selfish miner is α^2. Conditioned on the first event, the probability that the latency is greater than the gap in mining times, which follows an exponential distribution, is given by $1 - \mathbb{E}[\exp(-L_{SH})]$. Similarly, for T_2, we could have either the honest miner being the first one or the selfish miner being the first one. The requirement for the gap in mining times being less than the latency is the same as in other situations, but we add them together since either miner could have been the first one.

If we consider the first-order approximations, noting that L_{SH}, L_{HS} are nonnegative distributions, we get

$$a'_{1,0} = -\alpha \frac{\partial \mathbb{E}[L_{SH}]}{\partial L}, \qquad a_{0,1} = -(1-\alpha) \frac{\partial \mathbb{E}[L_{HS}]}{\partial L} \tag{4}$$

$$S'_2 = \alpha^2 \frac{\partial \mathbb{E}[L_{SH}]}{\partial L}, \quad H'_2 = (1-\alpha)^2 \frac{\partial \mathbb{E}[L_{HS}]}{\partial L} \tag{5}$$

$$T'_2 = \alpha(1-\alpha) \frac{\partial \mathbb{E}[L_{SH} + L_{HS}]}{\partial L} \tag{6}$$

where the derivatives are evaluated with respect to L at $L = 0$.

So we get the first-order term of

$$\mathcal{P}_1 = \gamma(1-\alpha)\alpha(1-\alpha) \frac{\partial \mathbb{E}[L_{SH} + L_{HS}]}{\partial L} \tag{7}$$

$$+ (1-\alpha)(-\alpha + 2\alpha^2) \frac{\partial \mathbb{E}[L_{SH}]}{\partial L} \tag{8}$$

$$- \alpha(-(1-\alpha) + 2(1-\alpha)^2) \frac{\partial \mathbb{E}[L_{HS}]}{\partial L} \tag{9}$$

$$\tag{10}$$

This simplifies further to

$$\mathcal{P}_1 = \alpha(1-\alpha) \left(\gamma(1-\alpha) - 1 + 2\alpha\right) \frac{\partial \mathbb{E}[L_{SH} + L_{HS}]}{\partial L}\Bigg|_{L=0}$$

Since $\frac{\partial \mathbb{E}[L_{SH}+L_{HS}]}{\partial L}\Big|_{L=0} > 0$, the requirement to be profitable is simply

$$\gamma(1-\alpha) - 1 + 2\alpha > 0 \iff \alpha > \frac{1-\gamma}{2-\gamma}$$

There are a few points to highlight here:

1. If $\gamma = 1$ (i.e. the honest miners always choose the selfish fork), a selfish miner can be profitable with arbitrarily small hash power.

2. If $\gamma = 1/2$, a selfish miner can be profitable with 1/3 the total hash power.
3. If $\gamma = 0$, a selfish miner can be profitable with 1/2 the total hash power. However, this is a trivial bound since that is the known threshold. For any $r \in [\alpha, 1)$, a selfish miner can find latencies such that their payoff is r by just delaying blocks for an extremely long time.

This isn't to say these bounds are absolutely tight however. This was only considering strategies where the selfish miner was just simulating an honest miner's latency. It is possible that a profitable and undetectable strategy exists with lower α values.

5.2 True Payoff with Probabilistic Latency

Let us recall again the payoff. We have a lemma that lets us simplify the payoff.

Lemma 3. *If $\lim_{d\to\infty} P_d^A$ exists, then $\mathcal{P}(A) = \lim_{d\to\infty} P_d^A$, and the* $\liminf$ *in the definition of payoff is equal to the limit.*

Proof. By linearity of expectation, we have

$$\mathbb{E}\left[\frac{1}{D}\sum_{d=1}^{D} P_d^A\right] = \frac{1}{D}\sum_{d=1}^{D}\mathbb{E}\left[P_d^A\right] = \frac{1}{D}\sum_{d=1}^{D} P_d^A$$

It is just an exercise in analysis from here. The average of the first n terms of a sequence converges to the same limit as the sequence itself if the sequence converges. □ □

The idea is that if we can find the limiting probability, then that is the payoff for a given miner.

Another lemma we will use is one regarding the probabilities of winning each depth given some set of information. Note that the probability of winning each depth can be written in terms of some minimal set of information. For example, if miners use a "first-seen" rule, then only the mining times for each block at that depth matters. We formalize that here.

Lemma 4. *[Steady-state distribution] Let $\{E_d\}_{d\geq 1}$ be a sequence of random variables taking values in some measurable space, and suppose $E_d \xrightarrow{d} E$ in distribution as $d \to \infty$. Write $P_d^A(E)$ to be the probability of player A winning depth d given that E has happened. If $P_d^A = \mathbb{E}_{e\sim E_d}[P_d^A(e)]$, and $P^A(\cdot)$ is continuous almost everywhere with respect to the limiting distribution of E, then* $\lim_{d\to\infty} P_d^A = \mathbb{E}_{e\sim E}[P^A(e)]$

Proof. Because $0 \leq P^A(E) \leq 1$, the family $\{P^A(E_d)\}$ is uniformly bounded. And since $E_d \xrightarrow{d} E$ in distribution and P^A is sufficiently nice, we have that

$$P^A(E_d) \xrightarrow{d} P^A(E)$$

. Because the sequence is uniformly bounded, convergence in distribution along with boundedness gives us that

$$\lim_{d\to\infty} \mathbb{E}[P^A(E_d)] = \mathbb{E}[P^A(E)]$$

and since we said P_d^A satisfies $P_d^A = \mathbb{E}[P^A(E_d)]$, we have the claim. □ □

We want to get a closed-form for the payoff now. The overall idea is that we first find the limiting distribution of the difference in mining times at each depth, and then we find the probability of winning given that difference in mining times.

We assume that lower received blocks take precedence even if parents/ancestors are missing. In the real-world, this could be implemented either by accepting deeper blocks as true or by somehow including the contents of parents/ancestors in each block as well.

Consider the following simulation that lets us work with respect to the minimum depth of what the two players have observed, rather than the overall state of the blockchain with respect to time.

Note that $\Delta, \ell_{SH}, \ell_{HS}$ are independent - whenever we do correlate them, we immediately sample new latency variables for the next block.

We now aim to characterize the limiting distribution.

Algorithm 2 Simulate the dynamics of the difference in mining times for each depth.

```
Require: $\alpha \in (0, 1)$, latency distributions $L_{SH}, L_{HS}$, number of rounds $N$
Sample $h \sim \text{Exp}(\alpha)$ and $s \sim \text{Exp}(1 - \alpha)$
$\Delta \leftarrow h - s$
Sample $\ell_{SH} \sim L_{SH}$ and $\ell_{HS} \sim L_{HS}$
for $n \leftarrow 1$ to $N$ do
    if $\Delta > 0$ then
        if $\ell_{HS} < \Delta$ then                          ▷ H sees S's block before its own
            Sample $s' \sim \text{Exp}(1 - \alpha)$
            $\Delta \leftarrow \Delta - s'$
            Sample $\ell_{HS} \sim L_{HS}$
            continue
        end if
    else
        if $-\Delta > \ell_{SH}$ then                         ▷ S sees H's block before its own
            Sample $h' \sim \text{Exp}(\alpha)$
            $\Delta \leftarrow \Delta + h'$
            Sample $\ell_{SH} \sim L_{SH}$
            continue
        end if
    end if
                ▷ Neither party sees the other's block in time; both create a block at this depth
    Sample $s' \sim \text{Exp}(\alpha)$ and $h' \sim \text{Exp}(1 - \alpha)$
    $\Delta \leftarrow \Delta + (s' - h')$
    Sample $\ell_{SH} \sim L_{SH}$ and $\ell_{HS} \sim L_{HS}$
end for
return $\Delta$
```

Definition 3. The transition kernel $K(x, \cdot)$ for the one-step update of the simulation is as follows for the current state being $\Delta = x$. The next state Δ' has law given by the combination of three different possibilities:

1. With probability $L_{HS}(x)$, we subtract an independent $h \sim \text{Exp}(1-\alpha)$:

$$\Delta' = x - h$$

2. With probability $L_{SH}(-x)$, we add an independent $s \sim \text{Exp}(\alpha)$:

$$\Delta' = x + s$$

3. With remaining probability, we sample both s and h as above and set

$$\Delta' = x + s - h$$

This defines a Markov kernel $K(x, dy)$ on $\mathbb{R}$. For any measurable function $g : \mathbb{R} \to \mathbb{R}$, we have

$$\int g(y)K(x, dy) = \mathbb{E}[g(\Delta')|\Delta = x]$$

Moreover, we can write $K(x, dy) = k(x, y)dy$ with

$$\begin{aligned} k(x,y) =& L_{HS}(x)(1-\alpha)e^{-(1-\alpha)(x-y)}\mathbf{1}_{x-y>0}+ \\ & L_{SH}(-x)\alpha e^{-\alpha(y-x)}\mathbf{1}_{y-x>0}+ \\ & r(x)\delta(y-x) \end{aligned}$$

where $r(x) = 1 - L_{HS}(x) - L_{SH}(-x)$ and

$$\delta(x) = \alpha(1-\alpha)\begin{cases} e^{(1-\alpha)x} & x < 0 \\ e^{-\alpha x} & x \geq 0 \end{cases}$$

Definition 4. The Markov operator induced by $K(x, dy)$ is given by

$$F(\mu)(dy) = \int_{\mathbb{R}} K(x, dy)\mu(dx)$$

This tells us how a given distribution will update. We know that the limiting distribution must be a fixed point, so we have the following.

Lemma 5. *Suppose Δ is a limiting distribution of Δ_n for the simulation - i.e. $\Delta_n \xrightarrow{d} \Delta$. Then*

(a) Δ satisfies $F(\Delta) = \Delta$.

(b) Δ admits a continuous Lebesgue density $f(y)$. That is, $\Delta(dy) = f(y)dy$, and moreover, $f(y)$ is continuous.

Proof. (a) simply follows from the properties of limits.
(b) follows from the density of the Markov kernel being continuous. □

We are now ready to actually express the limiting distribution in some form.

Theorem 6. *The PDF of the limiting distribution satisfies*

$$f''(x) = f'(x)Q(x) - Q'(x)f(x)$$

$$Q(x) = \begin{cases} 1 - 2\alpha + \alpha L_{SH}(-x) & x < 0 \\ 1 - 2\alpha - (1-\alpha)L_{HS}(x) & x > 0 \end{cases}$$

wherever $L_{SH}(-x)$ and $L_{HS}(x)$ have continuous first derivatives. $L_{AB}(x)$ represents the probability that a block sent by player A at time t is visible to player B by time $t + x$.

Proof. We have shown that f must exist and be continuous. It must satisfy

$$f(x) = \int_{\mathbb{R}} f(x')k(x', x)dx'$$

. A more clear representation of $k(x, y)$ is given by

$$k(x,y) = \begin{cases} \left(1 - L_{SH}\left(-x\right)\right) \alpha \left(1-\alpha\right) e^{(1-\alpha)(y-x)} & 0 > x > y \\ \left(1 - L_{HS}\left(x\right)\right) \alpha \left(1-\alpha\right) e^{-\alpha(y-x)} & y > x > 0 \\ \left(\left(1-\alpha\right) L_{HS}\left(x\right) + \alpha\right) \left(1-\alpha\right) e^{-(1-\alpha)(x-y)} & x > \max(0,y) \\ \left(\alpha L_{SH}\left(-x\right) + 1 - \alpha\right) \alpha e^{-\alpha(y-x)} & x < \min(0,y) \end{cases}$$

Note that for all x, $k(x, y)$ is smooth in y except for at $y = x$. The derivative with respect to y is given by $k_y = -\alpha k$ for $x < y$ and by $k_y = (1 - \alpha)k$ for $x > y$.

We have

$$f(y) = \int_{x<y} f(x)k(x, y)dx + \int_{x \geq y} f(x)k(x, y).$$

By the Liebniz Integration Rule, the derivative is then equal to

$$f'(y) = f(y)\left(k(y_-, y) - k(y_+, y)\right) - \alpha \int_{x<y} f(x)k(x, y)dx + (1-\alpha)\int_{x>y} f(x)k(x, y)dx$$

where $k(y_{\text{dir}}, y)$ is shorthand for $\lim_{x \to y^{\text{dir}}} k(x, y)$.

We split this into two cases: $y > 0$ and $y < 0$. With $y > 0$, we have

$$\begin{aligned} k(y_-, y) &= \left(1 - L_{HS}(y)\right) \alpha \left(1-\alpha\right) \\ k(y_+, y) &= \left(\left(1-\alpha\right) L_{HS}(y) + \alpha\right) \left(1-\alpha\right) \\ k(y_-, y) - k(y_+, y) &= -(1-\alpha)L_{HS}(y) \end{aligned}$$

So for $y > 0$, we have $f'(y) = -(1-\alpha)L_{HS}(y)f(y) - \alpha \int_{x<y} f(x)k(x, y)dx + (1-\alpha)\int_{x>y} f(x)k(x, y)dx$. If we differentiate again, we get

$$\begin{aligned} f''(y) =& -(1-\alpha)(L'_{HS}(y)f(y) + L_{HS}(y)f'(y)) \\ & - (\alpha k(y_-, y) + (1-\alpha)k(y_+, y))f(y) \\ & + \alpha^2 \int_{x<y} f(x)k(x, y)dx \\ & + (1-\alpha)^2 \int_{x>y} f(x)k(x, y)dx \\ =& -(1-\alpha)(L'_{HS}(y)f(y) + L_{HS}(y)f'(y)) \\ & - (1-\alpha)\left(\alpha + \left(1 - 2\alpha\right) L_{HS}\left(y\right)\right) f(y) \\ & + \alpha^2 \int_{x<y} f(x)k(x, y)dx \\ & + (1-\alpha)^2 \int_{x>y} f(x)k(x, y)dx \end{aligned}$$

Each of the integrals can be found as linear combinations of f, f' with coefficients according to L_{HS} and L'_{HS}. Specifically, we get the form in the given equation.

Similarly, we can write an equation for $y < 0$ as well. □

5.2.1 Winning Probabilities

For each depth, we define

$$\Delta = T^S - T^H$$

as what would be the difference in mining times for that depth. Instead of working with the winning probability of the selfish miner, we actually work with the winning probability of the honest miner since it is more convenient. We define the winning probability

$$W(x) := \Pr(\text{honest miner wins this depth} \mid \Delta = x).$$

This function will later be integrated against the steady state to give us the overall payoff.

There are multiple situations for resolving $W(x)$. If a player immediately sees the block made by the other player before they themselves can win, that resolves W. Otherwise, it could also resolve without the selfish player making a block at the next depth by the honest miner making their block at the next depth after seeing the selfish player's block and tie-breaking in favor of it. In the final case (i.e. it didn't get resolved yet), we update to the next x by considering the gaps in mining times and recurse.

Algorithm 3 Simulate $W(x)$

```
Draw $\ell_{SH} \sim L_{SH}$, $\ell_{HS} \sim L_{HS}$
if $\ell_{HS} < x$ then
    return H                                        ▷ Honest wins
end if
if $\ell_{SH} < -x$ then
    return S                                        ▷ Selfish wins
end if
Draw $s \sim \text{Exp}(\alpha)$,  $h \sim \text{Exp}(1-\alpha)$
Tie-breaking check:
if $\ell_{SH} < h - x$ and Bernoulli($\gamma$) then
    return S
end if
Recurse with updated difference:
return $W(x + s - h)$
```

More formally, draw ℓ_{SH}, ℓ_{HS} from the respective latency distributions, s from $\text{Exp}(\alpha)$, and h from $\text{Exp}(1-\alpha)$. There are the following cases:

1. If $\ell_{SH} < -x$, the selfish player wins immediately.
2. If $\ell_{HS} < x$, the honest player wins immediately.
3. If neither of the previous happen and $\ell_{SH} < h - x$, then the honest miner sees the selfish block. With probability γ, the selfish player wins immediately here.
4. If none of the previous happen, add $s - h$ to x and recurse.

Combining all these cases, $W(x)$ satisfies

$$W(x) = \mathbb{E}_{\ell_{SH},\ell_{HS},s,h}\left[\mathbf{1}_{\ell_{HS}<x} + \mathbf{1}_{\ell_{SH}>-x,\ell_{HS}>x}\left(1 - \gamma \mathbf{1}_{\ell_{SH}<h-x}\right) W(x+s-h))\right]$$

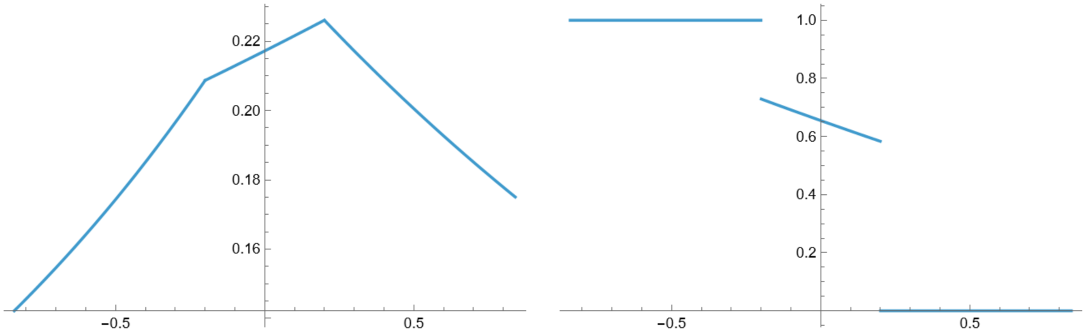
We can write this using a differential equation as well.

5.2.2 Analysis with deterministic and symmetric latency

A reasonable latency distribution is just to have all players delay blocks by the same constant amount of time. If $\ell_{SH} = \ell_{HS} = L$ almost surely, then:

- The density of the steady state is exponential on each interval $(-\infty, -L)(-L, L)(L, \infty)$.

- $W(x)$ is exponential on $(-L, L)$ and constant for $x < -L$ and $x > L$.



Steady-state density $f(x)$

Win probability $W(x)$

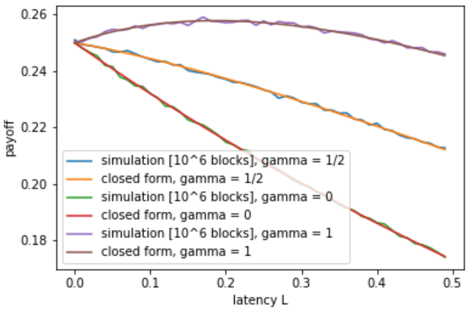
Parameters: $\alpha = 0.4,\ \gamma = 1/2,\ L = 0.2$

The closed form for the payoff in this setting with deterministic and symmetric latency can be written in closed form as a very complicated expression.

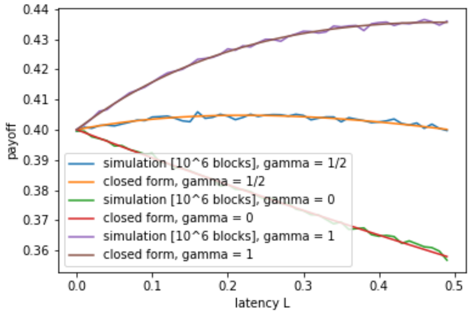
$$\mathcal{P}^S = \frac{e^{8L\alpha}(\gamma(-1+\alpha)-\alpha)\alpha^3 + \cdots + \alpha(3+(-1+\gamma)\alpha)}{(e^{2L}(1-\alpha)^2 - e^{4L\alpha}\alpha^2)(e^{2L}\cdots + e^{4L\alpha}\cdots)}$$

6 Experimental Validation and Numerical Analysis

We focus on the case with symmetric and deterministic latency here.

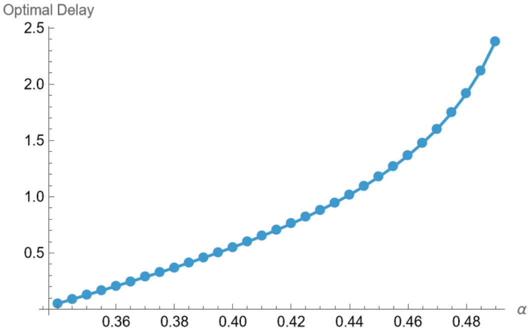


$\alpha = 0.25$

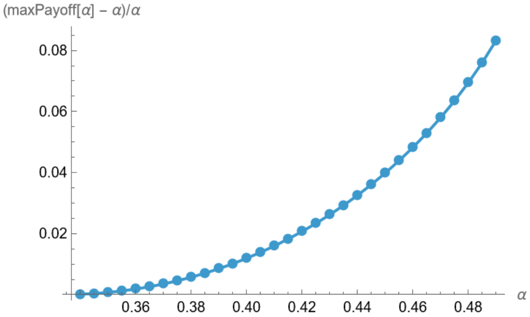


$\alpha = 0.40$

A natural question is to consider what the optimal delay is for the selfish miner, and what is the best payoff they can get.



Optimal delay for $\gamma = 1/2$



Optimal payoff for $\gamma = 1/2$

6.1 Experimental Conclusions

The main conclusion is that the simulated payoffs match very closely with the closed-form from the steady-state approach. This validates the correctness of the derived $W(x)$ and steady states.

If we look at the optimal delays/payoffs, we can see that the payoff increase above α is very small for α near the critical threshold. Manipulating the latency in the way we have discussing only provides a small advantage unless α is noticeably larger than the critical threshold.

However, this is still problematic. Honest miners would be incentivized to join the selfish mining pool. This defeats the purpose of a decentralized network

7 Conclusion

In this work, we introduced a new model for selfish mining in continuous timem focusing on fully undetectable attacks - those which are indistinguishable from honest behavior even when the distinguisher has real-time monitoring on the blockchain. In order to remain undetectable, the selfish miner simply acts like an honest miner with higher latency. Once we constrain the selfish miner in this way, the problem decays into analyzing how latency affects the miners' payoffs.

We analyzed what happens in the regime of small simulated latencies, and when a miner can be profitable. We show that if an attacker controls at least

$$\alpha > \frac{1-\gamma}{2-\gamma}$$

of the total hash power, where γ denotes the probability of the honest player choosing the selfish player's fork, the selfish player is profitable with small latency. We also give a necessary condition for selfish mining to be profitable in the regime of small simulated latencies.

We also looked at the general payoff and considered a steady-state model where we consider the limiting behavior of each miner with respect to depth. Perhaps surprisingly, we get representations of steady states and winning probabilities in terms of differential equations.

7.1 Future Work

The following problems are interesting to consider:

1. Optimizing payoff. What is the absolute best that the selfish miner can choose for the latency distributions?

2. Multiple selfish miners and Nash equilibria. What if two dishonest players each have more than 1/3 of the total hash power?

3. Relaxing undetectability. Real-world attackers may allow small detectable deviations.

4. Latency between honest miners or real latency from/to the selfish miner.

5. Impossibility of undetectable attack with $\gamma = 0$ and $\alpha < 1/2$. Although our results suggest this is the case, it is possible that other types of undetectable attacks exist where such an attack is possible.

Overall, we show that selfish mining that is both fully undetectable even with real-time monitoring and profitable is possible.

References

[1] Maryam Bahrani and S. Matthew Weinberg. "Undetectable Selfish Mining". In: *Proceedings of the 25th ACM Conference on Economics and Computation*. EC '24. New Haven, CT, USA: Association for Computing Machinery, 2024, pp. 1017–1044. ISBN: 9798400707049. DOI: `10.1145/3670865.3673485`. URL: `https://doi.org/10.1145/3670865.3673485`.

[2] Ittay Eyal and Emin Gün Sirer. "Majority is not Enough: Bitcoin Mining is Vulnerable". In: *CoRR* abs/1311.0243 (2013). arXiv: `1311.0243`. URL: `http://arxiv.org/abs/1311.0243`.

[3] Satoshi Nakamoto. "Bitcoin: A Peer-to-Peer Electronic Cash System". In: (May 2009). URL: `http://www.bitcoin.org/bitcoin.pdf`.

[4] Akira Sakurai and Kazuyuki Shudo. *The Rich Get Richer in Bitcoin Mining Induced by Blockchain Forks*. 2025. arXiv: `2506.13360 [cs.CR]`. URL: `https://arxiv.org/abs/2506.13360`.

[5] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. "Optimal Selfish Mining Strategies in Bitcoin". In: *CoRR* abs/1507.06183 (2015). arXiv: `1507.06183`. URL: `http://arxiv.org/abs/1507.06183`.