When is Slower Block Propagation More Profitable for Large Miners?

Zhichun Lu[1] and Ren Zhang[⋆1,2]

[1] Cryptape Co. Ltd., China
[2] Nervos
zhichunlu@cryptape.com, ren@nervos.org

Abstract. For years, Bitcoin miners put little effort into adopting several widely-acclaimed block acceleration techniques, which, as some argued, would secure their revenues. Their indifference inspires a theory that slower block propagation is beneficial for some miners. In this study, we analyze and confirm this counterintuitive theory. Specifically, by modeling inadvertent slower blocks, we show that a mining coalition that controls more than a third of the total mining power can earn unfair revenue by propagating blocks slower to outsiders. Afterward, we explore the strategies of an attacker that consciously exploits this phenomenon. The results indicate that an attacker with 45% of the total mining power can earn 58% of the total revenue. This attack is alarming as it is equally fundamental but more stealthy than the well-known selfish mining attack. At last, we discuss its detection and defense mechanisms.

Keywords: blockchain, slow block attack, selfish mining

1 Introduction

Nakamoto consensus (NC), implemented in Bitcoin [23] and hundreds of subsequent digital currencies [20], is the first and most influential protocol to maintain an inalterable ledger without relying on any prior knowledge of the participants' identities. The ledger, called the *blockchain*, is organized as a chain of *blocks*; each block contains a set of transactions. NC's participants, called *miners*, compete for the right to extend the ledger by solving a cryptographic puzzle, generated from the blockchain's latest block and a group of new transactions. The puzzle-solving process is called *mining*. A successful miner broadcasts the puzzle solution and the transactions as a new block, hoping that other miners would accept the block in their blockchains, so that the miner is entitled to a fixed *block reward*. When a block is mined during another one's propagation, these blocks may extend the same "latest block" and the blockchain thus *forks* into multiple chains. During a fork, NC prescribes miners to work on the *main chain*, which is the most computationally challenging one to produce—usually the longest one. When several chains are of equal "length", miners should work on the *first-received* one. We call

⋆ Corresponding author.

this situation a *tie*. Eventually, all miners would adopt the same longest chain, and blocks outside the chain are called *orphaned* and receive no reward.

Intuitively, the first-received policy incentivizes *all* miners to accelerate the propagation of their own blocks, because an accelerated block (1) is less likely to encounter a tie, and thus is less likely to be orphaned, and (2) could reach more miners before a slower competitor, and thus is more likely to win a tie. This intuition is stated or implicitly acknowledged in several early [5, 6] and recent [19, 27] studies. Bitcoin developers also released Fast Internet Bitcoin Relay Engine (FIBRE) [1] in 2016 to help the miners distribute blocks as fast as possible among each other.

However, contrary to the common expectation, a significant proportion of Bitcoin's mining power did not embrace FIBRE to avoid ties. This can be seen from the stable 0.2% *orphan rate*, i.e., the percentage of orphaned blocks, between 2015 and mid-2017 [8, 25]. The orphan rate dropped rapidly afterward thanks to the efforts of the network participants, i.e., *nodes*, rather than the miners. Specifically, in July 2017, the majority of nodes upgraded their clients to advocate their unwillingness for Bitcoin to split into multiple cryptocurrencies [4]. Such a massive-scale upgrade coincidentally deployed Compact Blocks [13], a network-level block propagation acceleration technique, which lowers the 3-second average latency for a new block to propagate to 50% of nodes to 500 milliseconds [14]. As a result, orphaned blocks were reduced from one every three to four days to several per year [8].

Maxwell [21] and several other researchers [11, 25] proposed a theory to explain the inconsistency between the miners' presumed rationale to accelerate their blocks' propagation and their indifference in reality. In this theory, slower block propagation might benefit larger miners, as they enjoy a headstart in finding the next block. This theory is not only *counterintuitive*—miners who accelerate their blocks may find themselves in a more disadvantageous situation—but also *subversive* to our understanding of NC's security—a system may be experiencing systemic unfairness even without any observable attacks. However, without a quantitative analysis of this phenomenon, whether or when can these seemingly inadvertent slow blocks profit their miners remains inconclusive.

In this paper, we address this situation by formally modeling the propagation of these slow blocks. We start by confirming this possibly inadvertent but systemic unfairness and computing its boundary conditions with a Markov process (MP), and then integrate Bitcoin's network parameters into the MP and evaluate against the boundary conditions. Further, we model a strategic adversary who consciously exploits this phenomenon for profit with a Markov decision process (MDP), which reveals more intricacies of NC's security.[3] In particular, our contributions include:

Confirming the Systemic Unfairness Caused by Slow Blocks. To confirm Maxwell's theory, we model the mining and block propagation process with some blocks slower than others with an MP. The MP is simple: the slow blocks' miner, termed *D-miner* for "delay", does not employ any strategic behavior based on

[3] Code available at `https://github.com/Mitsuhamizu/delayed-miner-reward`.

the status of the blockchain. Such a simple MP allows us to test the possibility of systemic unfairness even when all miners are benign.

The results show that slow blocks do raise D-miner's revenue share within a certain range of parameters. Specifically, a D-miner with a mining power share ρ[4] can gain an unfair profit when $\rho > (1-\gamma)/(2-\gamma)$, where γ is the proportion of other mining power that works on the slow block during a tie. The profitable threshold $(1-\gamma)/(2-\gamma)$ is less than 0.5 as long as $\gamma > 0$, contradicting the intuition that all miners should accelerate their blocks. Moreover, in line with the selfish mining attack [18], the D-miner's unfair profit grows superlinearly with ρ, which incentivizes rational miners to join their forces and form a coalition, damaging the network's decentralized structure, just like Maxwell predicted [21].

Evaluating the Boundary Conditions with Bitcoin's Network Parameters. As the key parameter γ in our MP is not directly measurable, our MP cannot be plugged right into reality and answer questions regarding the inadvertent D-miner with a given ρ, including (1) whether it should produce slow blocks (or propagate its blocks slower), and (2) what is the optimal delay. We thus extend our model so that γ can be "dissected" and computed from measurable data. This extension allows us to incorporate the measurement results from the Bitcoin network [2, 25], and thus answer the aforementioned questions.

The extended model indicates that in the pre-Compact-Block Bitcoin, a D-miner profits more by delaying its blocks with $\rho > 0.33$. The optimal additional delay grows roughly linearly from 0 when $\rho = 0.33$ to 6.8 seconds when $\rho = 0.49$.

Quantifying the Damage of Deliberate Attacks. Given that an inadvertent D-miner can make an unfair profit, likely, a *strategic* D-miner can further raise the profit by acting upon the blockchain's status. Rather than waiting for the slow blocks to propagate naturally like an inadvertent D-miner, here the adversary may push the blocks to the receivers when convenient. A fundamental difference between this *slow block attack* and selfish mining, where the adversary delays broadcasting the blocks as long as necessary to invalidate as many other miners' blocks as possible, is that our adversary never "delays" a block longer than a fixed maximum duration. In other words, the natural block propagation delay caps the delay time of all adversary blocks. Although such a constraint lowers the adversary's revenue, it also renders the attack undetectable via the traditional indicators of selfish mining [17], and thus does not risk causing a drop in the cryptocurrency's price. Therefore, we can regard the slow block attack as a stealthier, hence "safer" alternative to selfish mining.

We model the slow block attack with an MDP, where the adversary can choose what to do when one or more blocks are mined before a slow block finishes propagation. When $\rho < 0.42$, the optimal strategies output by our MDP are either honest or *naive*, i.e., to publish enough blocks to cause a tie with the honest public chain whenever possible. For $\rho \geq 0.42$, the strategy becomes more aggressive: the adversary may keep mining on its chain even when it is several blocks shorter than the honest chain. We also observe a rapid increase in the

[4] We list the notations in this paper in Table 3 of Appendix C.

unfair profit in this region. When $\rho = 0.45$, the adversary gains revenue 2% to 26% higher than mining honestly, depending on the maximum delay.

Discussing Countermeasures. At last, we discuss how to detect and/or prevent the slow block attack. All the technical solutions require the collective effort of the nodes and the miners. Therefore the core issue at hand is to raise awareness of the stealthiness and fundamentality of the attack. Consequently, we call for the community to rethink the implicit assumption that an NC system, or any other system, is fair and safe when there is no observable attack, and to replace the widely-believed "universal" 50% security threshold with a value matching the system's actual network condition.

2 Block Propagation: the Faster, the Better?

2.1 Nakamoto Consensus

NC is among the most influential and actively studied consensus protocols since the inception of Bitcoin. It is implemented in hundreds of subsequent cyptocurrencies [20], including Ethereum, the cryptocurrency with the second largest market capitalization, before it switches to another protocol in September 2022. Henceforth we use Ethereum to denote the cryptocurrency before the switch.

Each block in NC contains (1) its *height*—distance from the hard-coded *genesis block*, (2) the hash value of the *parent block*, (3) a set of transactions, (4) a timestamp when the block is mined, and (5) a nonce. Embedding the parent hash ensures that a miner chooses which chain to mine on before starting to mine. To construct a valid block, miners work on finding the right nonce so that the block hash is smaller than the *difficulty target.* In Bitcoin, this target is adjusted every 2016 blockchain blocks so that on average one block is appended to the blockchain in ten minutes. In Ethereum, the target is slightly adjusted per block, leading to an average block interval of 13 seconds [16].

NC prescribes miners to publish blocks the moment they are found. Blocks of the same height are *competing blocks.* Eventually, all but one of the competing blocks are *orphaned*, i.e., discarded by all miners, and receive no reward.

2.2 Selfish Mining

The most influential attack against NC is *selfish mining*, first analyzed by Eyal and Sirer [18] and later generalized to a family of strategies [24, 26, 28]. In these attacks, a *selfish miner* keeps discovered blocks secret and mines on top of them, hoping to gain a larger lead on the public chain of *honest blocks* mined by other miners. The selfish miner publishes the secret chain when the public chain catches up, or right before that, to invalidate as many honest blocks as possible.

Selfish mining is one of the most fundamental attacks against NC as it allows the attacker to gain a higher percentage of block rewards than its mining power share. As the attacker's revenue rises superlinearly with the mining power, rational miners are incentivized to attack collectively for higher profits. This situation not only damages the system's decentralized structure but also raises the success rates of various other attacks.

Luckily, the community generally believes that selfish mining has never happened in Bitcoin as it is easily detectable [17]. Essentially, as the attacker cannot predict when the next honest block will be mined—hence when the secret block should be released, a secret block's timestamp is usually inconsistent with its releasing time, which would expose the attacker. As we shall see, the slow block attack undermines NC's security in the same way but is more difficult to detect.

2.3 Two Conflicting Opinions

The absence of selfish mining is often attributed to its detectability, as visible attacks on a cryptocurrency often cause sharp declines in its price, resulting in a financial loss larger than the attacker's gain. This argument is termed *exchange rate rationality* by Bonneau et al. [9]. Given that rational miners would not risk being detected to mine selfishly, whether due to exchange rate rationality or some other reasons, people disagree on whether all miners are incentivized to accelerate the propagation of their own blocks.

On the one hand, it is a general belief that although it might be irrational to propagate *other* miners' blocks, all miners would accelerate their *own* blocks' propagation, to avoid ties or to raise the probability of winning potential ties by being the first-received ones. This argument is first proposed by Babaioff as early as 2012 [5] and has echoed for a decade [6, 19, 27].

On the other hand, observing the slow adoption of FIBRE, Maxwell mentioned in a talk in 2017 that slower block propagation might benefit larger miners [21]. He further suspected that this phenomenon, as in selfish mining, would drive miners to form a coalition that propagates blocks immediately to insiders but slower to outsiders. Likewise, Neudecker and Hartenstein [25] speculated that "the block propagation delay gives the miner of the last block an advantage in finding the subsequent block, until other miners have received the block." Cao et al. also mentioned in [11] that a slower block may cause some miners to "waste hashing power on an already solved cryptographic puzzle".

The conflict between these two opinions has profound implications for NC's security. If the former is true, as long as no mining coalition controls more than half of the mining power, NC systems are seemingly incentive compatible in the absence of detectable selfish mining. Otherwise, we need to reevaluate the commonly-believed 50% security threshold and the effectiveness of exchange rate rationality in securing the network. We aim to resolve these conflicting views by quantitatively analyzing the slower block propagation behavior.

3 Modeling the Inadvertent D-Miner

Our analysis starts with the simplest case, where the slow blocks' miner, despite the (possibly inadvertent) delay, strictly follows NC. First, we discuss the potential causes of such a delay. We then introduce our threat model and how we model such a D-miner with an MP, whose results confirm the systemic unfairness.

3.1 The Potential Causes of the Longer Block Propagation Delay

The delay may reside in multiple phases in a block's lifecycle.

Pre-propagation. Before broadcasting a block, the miner processes it internally. Such processing includes (1) combining the puzzle solution with the transactions and (2) sending the block to the "guard nodes" [22], who are in charge of the broadcast. Both steps may be time-consuming when the guard nodes crash or when the mining pool communication software does not behave as expected.

In-propagation. Two reasons may lead to in-propagation delay. First, the miner may broadcast from some poorly-connected nodes, with few connections and low bandwidth. Second, the block itself may take longer to synchronize, perhaps because it is larger than the other blocks. Note that Compact Blocks (CBs), which hope to reduce the propagation latency by optimistically not transferring the transactions by default, do not eliminate the second case. This is because CBs accelerate a block's propagation only when all its transactions are already synchronized when the block is mined. If some transactions are new, or only known to the block miner, for each hop of the block's propagation, an extra round trip is required to query these transactions [13].

Post-propagation. After receiving a new block, miners should verify the validity of its transactions before starting to work on it, to avoid wasting time on an invalid block. This can be time-consuming when some transactions refer to a large number of previous transactions stored physically distantly from each other on the hard disk [10].

An inadvertent delay may happen at any phase. However, for a malicious D-miner, pre-propagation is the most convenient phase, because it does not require a large or slow-to-verify block, allowing the attacker to, when convenient, stop the delay and push the block to the receivers before the competing blocks.

3.2 The Threat Model for Our MP

We choose a weak yet realistic threat model to showcase that systemic unfairness exists even without any sophisticated attacks. Time is continuous and mining is modeled as a Poisson process with an average block interval T. Accordingly, the probability that all miners find exactly n blocks in t seconds is $(t/T)^n/n! \cdot e^{-t/T}$. There is only one D-miner with mining power share $\rho < 1$, whose blocks are delayed up to D seconds. We do not prescribe $\rho < 0.5$ to cover the case that several large mining pools propagate blocks quickly among each other but slowly to the outside world. All other miners, who control mining power share μ (where $\mu + \rho = 1$) broadcast their blocks immediately, which cannot be delayed by the D-miner. Since there is no need to distinguish these other miners, we use the singular form "the undelayed miner" for simplicity. Both ρ and μ remain unchanged throughout the process. All miners follow the longest chain rule, and each fork lasts at most one block. This assumption is reasonable because all Bitcoin forks measured by Neudecker and Hartenstein [25] are one-block long. We neglect transaction fees and only consider block reward in this paper, as the former only makes up 1% of the miners' rewards in Bitcoin [7].

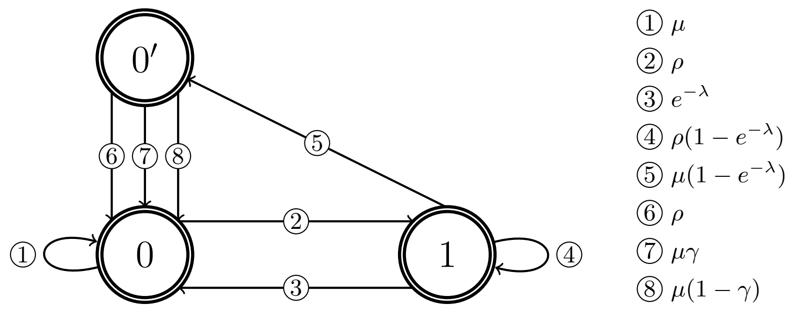

Fig. 1: Markov model with an inadvertent D-miner. The double-circled nodes denote the blockchain's states. The directed edges denote the transitions, whose probabilities are written on the right.

3.3 Our Markov Process

MP is commonly used to model the mining process without any strategic behaviors. In line with previous work [18], the blockchain's statuses are encoded as *states*, i.e., the double-circled nodes in Fig. 1, whose transitions are triggered by two kinds of events: (1) a new block is mined, or (2) D seconds has passed.

There are three states, named after the D-miner's lead to the undelayed miner. In state 0, both miners work on the same block. In state 1, the D-miner has just found a block and is delaying it. Here the D-miner is the only one working on this latest block. In state 0', the blockchain is forked, and the undelayed mining power may be split: some, with a proportion γ, works on the D-miner's block, while the other $1-\gamma$ works on the latest block mined by the undelayed miner.

We now describe the transitions, starting from state 0. If the undelayed miner finds a block (with probability μ, hereafter referred to as "w.p."), the system stays at state 0, and the undelayed miner gets a block reward (①). Otherwise, the next block is mined by the D-miner (w.p. ρ), who starts the delay and the system enters state 1 (②).

Three transitions may happen at state 1. If no block is mined in D seconds, the D-miner releases the block, gets a block reward, and then the state returns to 0 (③). As mining is modeled as a Poisson process, ③ happens with probability $e^{-\lambda}$, where $\lambda = D/T$ is the expected number of blocks mined in these D seconds. If there are new blocks and the first one is mined by the D-miner (w.p. $\rho(1-e^{-\lambda})$), the previous slow block is broadcast immediately, issuing one reward to the D-miner, which starts delaying the new block, transiting the state to 1 again (④). If there are new blocks and the first one is mined by the undelayed miner (w.p. $\mu(1-e^{-\lambda})$), the blockchain is forked and the state transits to 0' (⑤).

Three transitions may happen at state 0'; all end with state 0. If the D-miner finds the next block (w.p. ρ), it wins the tie, claiming two rewards (⑥). If the undelayed miner finds a block on the D-miner's block (w.p. $\mu\gamma$), each miner gets one reward (⑦). Otherwise, if the undelayed miner finds a block on the undelayed block (w.p. $\mu(1-\gamma)$), the undelayed miner gets two rewards (⑧).

3.4 State Probabilities and Relative Revenues

Stationary Distribution. We derive the following equations from Fig. 1, where p_0, p_1, and $p_{0'}$ are the stationary probability of states 0, 1, and 0', respectively:

$$\begin{cases} p_0 = p_0\mu + p_1 e^{-\lambda} + p_{0'} \\ p_1 = p_0\rho + p_1(1 - e^{-\lambda}\rho) \\ p_{0'} = p_1(1 - e^{-\lambda})\mu \\ 1 = p_0 + p_{0'} + p_1 \end{cases} . \tag{1}$$

Solving Eqn. (1) gives us these probabilities:

$$\begin{aligned} p_0 &= (\mu + \rho e^{-\lambda})/(1 + \mu\rho + \rho^2 e^{-\lambda}) \ , \\ p_{0'} &= \rho\mu(1 - e^{-\lambda})/(1 + \mu\rho + \rho^2 e^{-\lambda}) \ , \\ p_1 &= \rho/(1 + \mu\rho + \rho^2 e^{-\lambda}) \ . \end{aligned} \tag{2}$$

Relative Revenue. The revenues for the D-miner and the undelayed miner, denoted r_d and r_u, can be computed from the transition probabilities among the states and their corresponding rewards:

$$\begin{aligned} r_d &= 2p_{0'}\rho + p_{0'}\mu\gamma + p_1(1 - e^{-\lambda})\rho + p_1 e^{-\lambda} \ , \\ r_u &= p_0\mu + 2p_{0'}\mu(1 - \gamma) + p_{0'}\mu\gamma \ . \end{aligned} \tag{3}$$

Combining Eqn. (2) and (3) allows us to compute the *relative revenue*, i.e., the proportion of the D-miner's revenue among all the rewards:

$$R_d = \frac{r_d}{r_d + r_u} = \rho(e^{-\lambda} + (1 - e^{-\lambda})(\rho^2\gamma - 2\rho^2 - 2\rho\gamma + 3\rho + \gamma)) \ . \tag{4}$$

Equation (4) shows that R_d is a function of three inputs ρ, γ, and λ. We define the *unfair revenue* as $R_d - \rho$, i.e., the difference between R_d and the D-miner's fair reward share, and plot how it varies with these inputs in Fig. 2.

Analysis. Here are three patterns from Fig. 2 and their underlying reasons.

Observation 1. *The D-miner earns unfair revenue with a large enough ρ.*

This confirms the systemic unfairness, that propagating blocks slower can be more profitable than mining honestly, despite that the forks are no more than one block long. Even when $\gamma = 0$, i.e., no undelayed mining power works on the D-miner's block in a tie, the D-miner can still earn an unfair profit with $\rho > 0.5$.

Essentially, the unfair revenue comes from orphaning the undelayed miner's blocks in ties. The D-miner earns a profit if more than μ percentage of orphaned blocks are mined by the undelayed miner, but suffers a loss otherwise.

Observation 2. *Whether the D-miner earns unfair revenue depends only on ρ and γ, not on λ.*

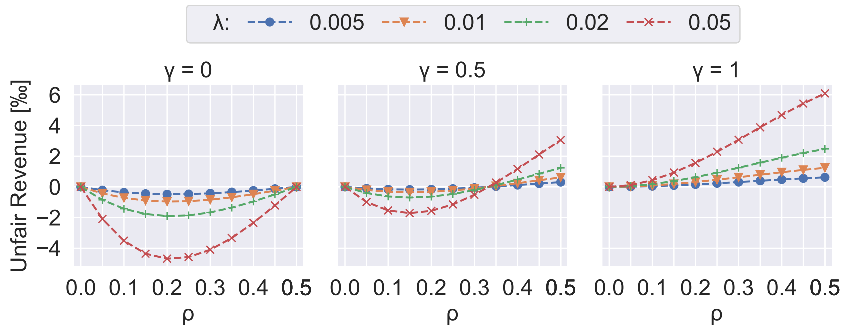

Fig. 2: Overview of unfair revenue for the D-miner.

To quantify when the D-miner earns an unfair profit, we solve the inequality $R_d > \rho$, leading to the condition $\rho > (1-\gamma)/(2-\gamma)$. Interestingly, this condition does not involve the delay duration λ, because here the D-miner's profitability depends only on the *probability of winning ties*, which further relies on ρ and γ, but not on the *frequency of ties*, which relies on ρ and λ.

The multifunctionality of ρ also explains why when γ and λ are fixed, the D-miner's unfair revenue first decreases, and then increases with a growing ρ. Specifically, when ρ is small, the D-miner loses almost all ties, thus increasing ρ results in a higher loss as it raises the frequency of ties; when ρ grows larger, the D-miner wins more ties, thus profiting more from the ties.

Observation 3. *When γ and ρ are fixed, the D-miner's profit/loss amplifies with a larger λ.*

A larger λ means a higher frequency of ties, amplifying the D-miner's profit/loss.

4 Are Inadvertent Slow Blocks Profitable in Reality?

Results from the previous section cannot be applied to reality yet, as, unlike ρ and λ, which are either known or controllable/measurable, the key parameter γ is not only unknown but also not directly measurable. Therefore, in this section, we extend our model by dissecting γ with real-world data, which enables us to quantify the profitability of an inadvertent D-miner in the Bitcoin network.

4.1 Extracting the D-γ Relationship in the Bitcoin Network

Intuitively, when both competing blocks propagate naturally, the distribution of mining power working on each block—other than their own miners—is mainly decided by the interval between their announcements. This intuition guides us to express γ as a function of D, which consists of two tasks: (1) express γ as a function of the *headstart*, i.e., the (equivalent) announcement interval, (2) express the headstart as a function of D.

Headstart to γ. We can only learn the relation between the headstart t_{hs} and γ from a series of (t_{hs}, γ) data points, where t_{hs} measures how long the block is announced *before* its competitor, in seconds. We fetch the t_{hs} values directly from Neudecker and Hartenstein's measurement study [25], which covers all Bitcoin forks between 2015 and 2017. Although γ is not directly measurable, we can estimate it by extending our model. According to [25], a miner wins a tie with probability $\text{P}_{\text{win}} = 3.07 \times 10^{-5} t_{\text{hs}} + 0.63$. On the other hand, $\text{P}'_{\text{win}} = \rho + \gamma(1-\rho)$ in our model, which is the combination of transitions ⑥ and ⑦ in Fig. 1. Assuming $\text{P}'_{\text{win}} = \text{P}_{\text{win}}$, we have $\gamma = (3.07 \times 10^{-5} t_{\text{hs}} + 0.63 - \rho)/(1-\rho)$. By inputting the miners' then-mining-power share—fetched from IntoTheBlock [2]—as ρ into this equation, we now have the estimated γ for each data point.

To fit these (t_{hs}, γ) data into a curve, we introduce an additional heuristic that $\gamma = 0.5$ when $t_{\text{hs}} = 0$. This is reasonable as two simultaneously-announced competing blocks should have an equal chance to be selected by a third party. Not surprisingly, a linear equation $\gamma(t_{\text{hs}}) = 6.37 \times 10^{-2} t_{\text{hs}} + 0.5$ for $t_{\text{hs}} \in [-7.8, 7.8]$, learned via the least squares method, already gives us a good estimation: the root-mean-square deviation (RMSD) is as low as 0.12.

This (t_{hs}, γ) relation implies that $\gamma = 1$ when $t_{\text{hs}} = 7.8$, meaning that it takes 7.8 seconds for a block to be propagated to all the miners. This result justifies the reasonableness of our model as it is consistent with the measured data: blocks propagated to 90% of nodes in 5 to 20 seconds [25].

We further verify this relation with Ethereum's data. Specifically, we fetch 236 553 fork instances, group those with similar ρ and t_{hs}, and compare each (ρ, t_{hs}) group's estimated $\text{P}'_{\text{win}} = \rho + \gamma(t_{\text{hs}}) \cdot (1-\rho)$ and the actual P_{win}. The results confirm the accuracy of our model, whose details are in Appendix A.

D to Headstart. A stable delay D does not imply a stable t_{hs}, as the competing block may be announced anytime during D. If the undelayed competing block is mined in the first $d = D - 7.8$ seconds, the undelayed block may enjoy a headstart, i.e., $t_{\text{hs}} \leq 0$. Otherwise, the slow block enjoys a headstart and $t_{\text{hs}} > 0$.

We now solve how d, i.e., the slow block's delay in addition to the natural propagation latency, affects the probability distribution of t_{hs}. As mining is a Poisson process, the interval between the slow block's and the undelayed block's mining, denoted t_{in}, follows an exponential distribution, whose density function is $f(t_{\text{in}}) = \mu/600 \times e^{-(\mu/600)t_{\text{in}}}$ for $t_{\text{in}} \in (0, \infty)$, where $\mu/600$ is the expected number of undelayed blocks mined in a second. We compute the probability density function of t_{hs} from $f(t_{\text{in}})$ via two post-processing steps. First, $t_{\text{hs}} = t_{\text{in}} - d$ as the slow block's announcement is delayed for d seconds. Second, the density function is normalized by dividing $1 - e^{-(\mu/600)(d+7.8)}$ to exclude the situation that no undelayed blocks are mined during the slow block's propagation.

Finally, by linking these two relations, we can estimate γ with a given d: $\gamma = 0.748 - 0.0318d$ when $d \ll 600$. We omit the detailed process as it is relatively straightforward compared to the previous two steps.

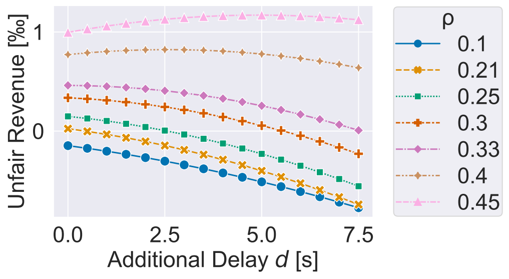

Fig. 3: Extra revenue varies with d and ρ.

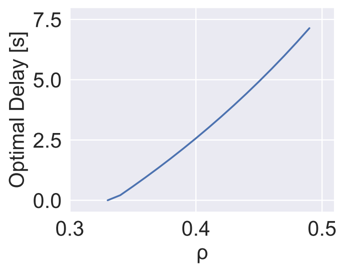

Fig. 4: The optimal delay d.

4.2 Applying the Extended Model to the Bitcoin Network

We instantiate our MP in Sect. 3.3 with $\lambda = D/T = (d + 7.8)/600$ and $\gamma = 0.748 - 0.0318d$. We plot how d and ρ affect the unfair revenue $R_d - \rho$ in Fig. 3 and the most profitable additional delay d in Fig. 4.

Two thresholds—0.21 and 0.33—are identifiable from Fig. 3. Miners with $\rho > 0.21$ gain unfair revenue with no additional delay beyond the universal 7.8 seconds. Miners with $\rho > 0.33$ can increase their earnings by intentionally delaying block propagation, as their unfair revenue is monotonically increasing at $d = 0$. This 0.33 threshold roughly corresponds with $\gamma = 0.5$ in Fig. 2. The optimal—most profitable—additional delay grows roughly linearly, from 0 when $\rho = 0.33$ to 6.8 seconds when $\rho = 0.49$. As a side note, the unfair revenue for ρ between 0.2 and 0.33 mainly comes from receiving undelayed blocks earlier than other miners, which decreases with a growing d.

5 Modeling the Strategic D-Miner

We now analyze how and how much a strategic D-miner can profit from the systemic unfairness by modeling its decisions with an MDP. We name the output strategy the *slow block attack*.

5.1 The Threat Model for Our MDP

We highlight some key settings here; other settings are identical to that of our MP in Sect. 3.2. We limit $\rho < 0.5$ to avoid pathological actions. In line with previous MDP-based analyses [26,28], the strategic D-miner can (1) choose which block to mine on, (2) withhold multiple blocks, and (3) decide when and how many blocks to publish. Such freedom does not render the problem unsolvable, because, in the longest chain rule, a rational for-profit attacker maintains at most one secret chain and only mines on the tips of chains, as proved by Sapirshtein et al. [26]. Their proof applies to our model. Henceforth we use *undelayed blocks* to denote "blocks mined by the undelayed miner" for brevity. Our model differs from previous analyses in that the D-miner must broadcast a secret block within every D seconds. This constraint adds a type of transition “delay” to our MDP, which models the passage of time and thus complicates the modeling due to the continuity of time. Consequently, we limit the attacker’s actions when accurate modeling is infeasible, so that our MDP outputs achievable strategies and lower bounds on the D-miner’s profitability, demonstrating the severity of the attack.

5.2 Our Markov Decision Process

Modeling Mining Processes as MDPs. An MDP models decision-making in situations where outcomes are partly random and partly under the control of a strategic player. Formally, an MDP is a four-element tuple (S, A, P, R). S is the *state space*, encoding all status and history information that might influence the player’s decision. A is the *action space*, which includes all possible rational choices in an arbitrary state. P is the *transition matrix*, which encodes all possible outcome states for each $(state, action)$ pair and their probability distribution. R is the *reward matrix*, which records a *reward* for every $(state, action, new\_state)$ transition; the reward is used to compute the final utility.

MDP is commonly employed in modeling mining processes [26,28]. We summarize Sapirshtein [26]’s selfish mining MDP here as the baseline of our design. In their MDP, a state transition is triggered by a mining event, and the attacker makes decisions at the beginning of a state. Blocks accepted or abandoned by both miners are *settled*, whose corresponding rewards are allocated to the miners. Settled blocks are removed from the state encoding, as they do not affect the attacker’s decisions. Specifically, a state is a 3-tuple $(l_d, l_u, fork)$ where l_d and l_u represent the lengths of the unsettled attacker chain and the public chain, respectively, and *fork* indicates the latest block’s miner and whether the attacker has the option to Match, which is defined next. There are at most four available actions at any moment: Adopt to throw away the attacker chain and mine on the public chain, Override to publish until the $(l_u + 1)$-th attacker block to invalidate the public chain, Match to publish until the l_u-th attacker block to cause a tie, which is available only when the honest miner has just mined a block and the attacker has a competing secret block, and Wait to keep mining on the attacker chain. We omit the reward distribution and the state transition matrices here.

Overview. Our MDP differs from previous works as we introduce a new type of transition called *delay*: the passage of D seconds. When the D-miner chooses not to publish all withheld blocks, the next transition must be a delay.

Formalizing these transitions is highly nontrivial as both the D-miner and the undelayed miner may mine blocks during the delay, and it is infeasible to encode all information that might influence the D-miner’s decision. For example, the D-miner may make decisions based on the time of the first undelayed block: to Override if the block is mined at the beginning of D, and to Match if it is mined at the end. However, we cannot encode time into the state, as time is continuous, and the number of states must be finite. Dividing D into several slots and recording the mining sequence in each slot is also impractical, as in that case, the number of states is too large to be solvable.

To address this challenge, we prescribe the D-miner's strategy during the delay so that the system's state after the delay only depends on the pre-delay state and the number of blocks mined by each miner during the delay. Through careful engineering of the state space S and the action space A, the number of states becomes solvable after this simplification, yet our MDP still reveals a series of insights into the slow block attack. Next, we describe our MDP design.

State Space. A state is a four-element tuple $(l_d, l_w, l_u, \mathit{fork})$. The lengths of the D-miner's and the undelayed miner's chains are encoded as l_d and l_u, respectively. Note that the common ancestors are not counted in l_d or l_u as they are settled. The variable l_w is the number of withheld blocks, which satisfies $l_w \leq l_d$ as these blocks are a suffix of the D-miner's chain. The variable *fork* indicates whether some undelayed mining power is working on the D-miner's chain, which is meaningful only in a tie, i.e., $l_d - l_w = l_u$. It has two possible values:

- Active. The l_u-th D-miner's block is published *along with* the last undelayed block, so some undelayed mining power may work on the D-miner's chain.
- Inactive. The l_u-th D-miner's block is published *after* the last undelayed block, so all undelayed mining power works on the undelayed chain.

There are two differences between our state space and that of [26]. First, we encode l_w explicitly so that the D-miner can learn/decide whether the next transition is a delay, and how many such delays to look forward to. Second, our $fork$ has only two options, because the D-miner never needs to explicitly choose the Match action, whose reason is explained next.

Action Space. There are only three actions Adopt, Override, and Wait in our MDP. The definitions of Adopt and Override are identical to their counterparts in [26]. The Wait and Match actions in [26] are merged into our Wait action:

- Wait. If there are no withheld blocks, the D-miner keeps mining on its chain until the next block generation event. Otherwise, i.e., during a delay, the D-miner mines on its chain until the delay ends, and publishes enough blocks to cause a tie, i.e., "Match", when the undelayed chain "catches up from behind" and reaches the D-miner's *pre-delay* chain length l_d.

To understand this change, we first introduce *how* we prescribe the D-miner's strategy during the delay. Given that a fixed strategy is necessary to avoid an overwhelming number of states, we want a strategy that is simple enough to be computationally feasible, yet still reasonable for the D-miner. A naive strategy is to force the D-miner to keep mining on its chain without publishing anything throughout the delay. This strategy causes a significant loss when the undelayed chain *overtakes*—catches up from behind and surpasses—the D-miner's chain during the delay, as the D-miner loses its entire chain with a high probability. A better strategy is to prescribe the D-miner to publish the entire chain at the exact moment when the undelayed chain catches up. However, as both miners may find blocks during the delay, it is difficult to predict *when* the catch-up happens unless we encode the full sequence of mining events during the delay into the MDP, which is computationally infeasible. Therefore, we choose a middle

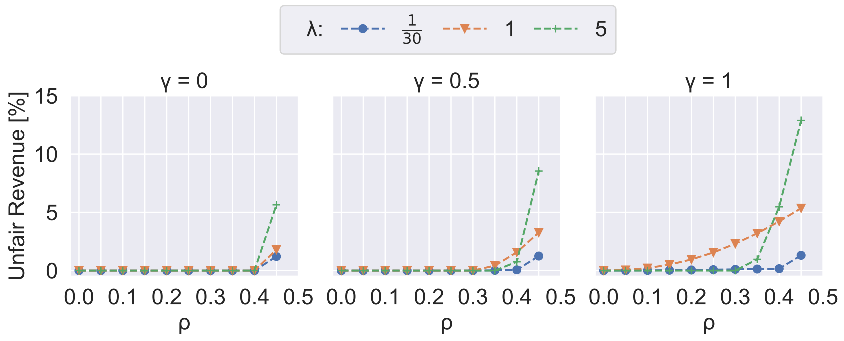

Fig. 5: The ϵ-optimal unfair revenue.

ground between these two strategies and prescribe the D-miner to cause a tie when the undelayed chain catches up to the D-miner's pre-delay chain length, which is reasonable as it lowers the attacker's risk of losing the whole chain, yet still manageable as there is no need to enumerate all possible mining sequences.

As an unexpected benefit of this "middle-ground" strategy, the D-miner never needs to explicitly choose Match. If the last undelayed block is mined during a delay, Match is automatic; otherwise, i.e., outside delays, the D-miner has no secret block by our threat model, thus cannot choose Match.

Transition and Reward Matrices. We leave the full matrices to Appendix B and briefly overview how we compute the post-delay transitions here. We denote the number of blocks mined by the D-miner and the undelayed miner during the delay as n_d and n_u, respectively. They are independent and both follow the Poisson distribution, allowing us to compute their joint probability distribution of the resulting states $(l_d + n_d, l_w + n_d, l_u + n_u, \text{inactive})$. We then enumerate the possible outcomes for the implicit Match action for all $l_u < l_d \leq l_u + n_u$ and their corresponding probability, and update the corresponding transitions.

Solving the MDP. We define the utility as the D-miner's relative revenue and solve the MDP with the *RelativeValueIteration* method of pymdptoolbox [12]. The stopping criterion is set to $\epsilon < 10^{-4}$. The upper bound for l_d and l_u is set to 60. We solve the MDP for all combinations of $\rho = \{0, 0.05, \cdots, 0.45\}$, $\gamma = \{0, 0.5, 1\}$, and $\lambda = \{1/30, 1, 5\}$.

5.3 Unfair Revenues and Profitable Thresholds

Relative Revenues. We visualize the D-miner's unfair revenue under various λ, γ, and ρ in Fig. 5, and notice two patterns:

Observation 4. *The strategic D-miner's unfair revenue increases with* λ.

This is consistent with our intuition: a longer delay upper bound gives the D-miner larger room for malicious manipulation, thus increasing the unfair revenue.

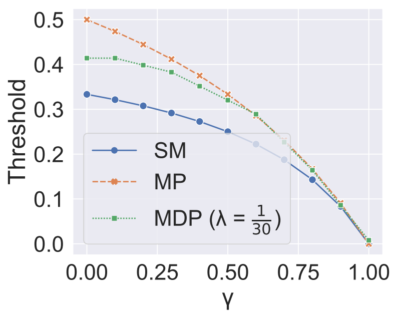

Fig. 6: Comparison of thresholds.

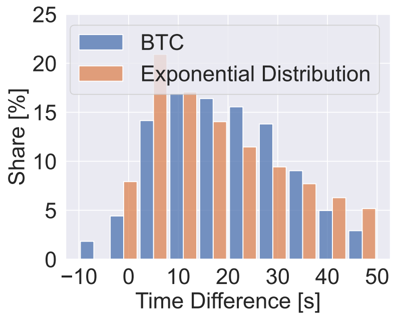

Fig. 7: The time difference.

Also, the unfair revenue never goes below zero, as the D-miner has the honest strategy as a safe choice.

Observation 5. *The unfair revenue rockets when $\rho \geq 0.42$ for all γ and λ.*

We locate the reason by examining the optimal strategies. For $\rho < 0.42$, the unfair revenue mainly comes from winning ties; the D-miner chooses Abandon if its chain is shorter than the undelayed chain. For $\rho \geq 0.42$, the optimal strategy becomes more aggressive: it keeps mining on its own chain even when it is two to four blocks behind. Admittedly, this also makes the attack detectable. We list the full strategy when $\rho = 0.45$, $\gamma = 1$ in Table 1. This strategy is counterintuitive given that $\rho < \mu$. We attribute this to the D-miner's higher risk tolerance than the undelayed miner: the D-miner gives up its chain at its chosen time, but the undelayed miner gives up as soon as it is one block behind. Indeed in the gambler's ruin problem, if one gambler has two to four coins and a 45% one-time winning rate, and the other one has only one coin but a 55% one-time winning rate, the latter is 1.48 to 2.47 times more likely to bankrupt than the former.

Profitable Thresholds. We compare the profitable thresholds, i.e., the minimum ρ to be more profitable than the honest strategy of our MP ($(1-\gamma)/(2-\gamma)$), MDP, and selfish mining ($(1-\gamma)/(3-2\gamma)$, from [18]) in Fig. 6. The results show that the threshold of the strategic D-miner resides between that of the inadvertent D-miner and the selfish miner.

6 Detection and Defense

Detecting via the Timestamp-Announcement Difference. As mentioned in Sect. 2.2, it is long known that selfish mining can be detected by measuring the difference between the blocks' timestamp and their announcement time. Unfortunately, it is difficult to apply the same trick to detect the slow block attack, at least in Bitcoin, due to the miners' long timestamp updating cycle. We plot Bitcoin's timestamp-announcement difference distribution in Fig. 7 (blue bars), whose data are provided by Grundmann, the maintainer of a Bitcoin monitoring site [14]. There are 33 453 blocks from January 1 to August 20, 2020, and 93% of their differences are within $[-10, 50]$ seconds. The distribution is far from ideal, where all data concentrate at 0. Instead, it is close to the exponential distribution with an expectation of 30 seconds (orange bars). We speculate that this is because mining pools use specialized software, e.g., P2Pool [3], to assign tasks and collect shares among individual miners, which updates the timestamp roughly every 30 seconds. Consequently, as long as the D-miner keeps the delay within 30 seconds, e.g., $\lambda < 1/20$, it is difficult, if not impossible, to detect the slow block attack from the timestamp-announcement difference.

Detecting via the Orphan Rate. Most mining pools nowadays publish the blocks they mined for transparency, enabling us to compute a pool's percentage of blocks that have competing blocks. A mining pool encountering block races more often than the others is a strong indicator of systemic unfairness. Also, a rise in the overall orphan rate may indicate network issues or malicious behaviors.

Eliminating the Inadvertent Delays from the Protocol Level. Our analysis shows that we cannot expect the miners, especially large ones, to accelerate their blocks' propagation, as their incentive is not aligned. Yet we can prevent these inadvertent delays from happening via protocol-level efforts, which do not need the miners' individual consent or proactive operations. For example, to avoid in-propagation delay due to transaction synchronization, NC-Max [29] prescribes that transactions must be synchronized before their confirmation, so that blocks are always propagated at the maximum possible speed. NC-Max thus reduces the latency to just 18.7% of that in NC, given a context of 40-second average block interval and 100 transactions per second workload. This also reduces the unfair revenue of miners with $\rho = 0.4$ to 19.2% of that in NC.

Modifying the Tie-Breaking Mechanism. When a delayed block is forced to be broadcast due to the announcement of a competing block, its timestamp is usually inconsistent with its announcement time. This is because the D-miner cannot predict when the competing block will be mined. This phenomenon inspires us to propose a new tie-breaking mechanism, which favors the block with a more accurate timestamp. This new mechanism only works if all undelayed miners synchronize their clocks and keep updating their blocks' timestamps. We leave the detailed threshold and rule of this mechanism to future work.

7 Conclusion

Despite numerous efforts from the Bitcoin community, many miners refused to accelerate their blocks' propagation, as revealed by the slow adoption of FIBRE and Compact Blocks. In this paper, we confirmed Maxwell's theory that slower propagation could be more profitable. These seemingly-benign slow blocks lead to systemic unfairness, which could be deliberately exploited for higher revenue. The slow block attack fundamentally undermines NC's security just like the selfish mining attack, yet is more difficult to detect. To mitigate this unfairness and deter such inadvertent or malicious behaviors, we call on the community to (1) keep accelerating block propagation on the protocol and the network layer,

(2) synchronize the clocks and update the block timestamp more frequently, and (3) modify the protocol to defend against this attack. Most importantly, we must explicitly address it via proactive actions, rather than hoping that the miners' incentive will be spontaneously aligned.

This attack is another example that our attack detection metrics are limited by existing security analyses—most formal analyses of NC assume a fixed block propagation latency. Therefore, we—researchers—should keep looking for attacks folded into the assumptions of these analyses.

References

1. FIBRE. https://bitcoinfibre.org, accessed: 2022-09-01
2. intotheblock. https://www.intotheblock.com, accessed: 2022-09-01
3. P2Pool. http://p2pool.in, accessed: 2022-09-01
4. User Activated Soft Fork. https://en.bitcoinwiki.org/wiki/User_Activated_Soft_Fork, accessed: 2022-09-01
5. Babaioff, M., Dobzinski, S., Oren, S., Zohar, A.: On Bitcoin and red balloons. In: 13th ACM Conference on Electronic Commerce. pp. 56–73. ACM (2012)
6. Bahack, L.: Theoretical Bitcoin attacks with less than half of the computational power (draft). arXiv preprint arXiv:1312.7013 (2013), https://arxiv.org/pdf/1312.7013.pdf
7. Blockchain: Bitcoin block explorer (2017), https://blockchain.info/
8. Blockchain Luxembourg S.A.: Orphaned blocks - blockchain.info (Aug 2019), https://www.blockchain.com/btc/orphaned-blocks
9. Bonneau, J., Miller, A., Clark, J., Narayanan, A., Kroll, J.A., Felten, E.W.: SoK: Research perspectives and challenges for Bitcoin and cryptocurrencies. In: IEEE Symposium on Security and Privacy (S&P). pp. 104–121. IEEE, IEEE (2015)
10. Buterin, V.: The limits to blockchain scalability (2021), https://vitalik.ca/general/2021/05/23/scaling.html
11. Cao, T., Decouchant, J., Yu, J., Esteves-Verissimo, P.: Characterizing the impact of network delay on bitcoin mining. In: 2021 40th International Symposium on Reliable Distributed Systems (SRDS). pp. 109–119. IEEE (2021)
12. Chadès, I., Chapron, G., Cros, M.J., Garcia, F., Sabbadin, R.: Mdptoolbox: a multi-platform toolbox to solve stochastic dynamic programming problems. Ecography **37**(9), 916–920 (2014)
13. Corallo, M.: Compact block relay (2016), https://github.com/bitcoin/bips/blob/master/bip-0152.mediawiki
14. DNS Research Group, KASTEL @ KIT: Bitcoin network monitor (2019), https://dsn.tm.kit.edu/bitcoin/
15. Etherscan: Ethereum ETH blockchain explorer (Aug 2019), https://etherscan.io/
16. Ethstats: Ethereum network status (Jun 2020), https://ethstats.net/
17. Eyal, I., Sirer, E.G.: How to detect selfish miners (Jan 2014), https://hackingdistributed.com/2014/01/15/detecting-selfish-mining/
18. Eyal, I., Sirer, E.G.: Majority is not enough: Bitcoin mining is vulnerable. In: Financial Cryptography and Data Security. pp. 436–454. Springer (2014)
19. Mao, Y., Venkatakrishnan, S.B.: Less is more: Fairness in wide-area proof-of-work blockchain networks (2022). https://doi.org/10.48550/ARXIV.2204.02461, https://arxiv.org/abs/2204.02461

20. mapofcoins: Map of coins: BTC map (2018), `http://mapofcoins.com/bitcoin`
21. Maxwell, G.: Advances in block propagation (2017), `https://www.youtube.com/watch?v=EHIuuKCm53o`
22. Miller, A., Litton, J., Pachulski, A., Gupta, N., Levin, D., Spring, N., Bhattacharjee, B.: Discovering bitcoin's public topology and influential nodes (2015), `https://www.cs.umd.edu/projects/coinscope/coinscope.pdf`
23. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008), `http://www.bitcoin.org/bitcoin.pdf`
24. Nayak, K., Kumar, S., Miller, A., Shi, E.: Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In: IEEE European Symposium on Security and Privacy (EuroS&P). pp. 305–320. IEEE, IEEE (2016)
25. Neudecker, T., Hartenstein, H.: Short paper: An empirical analysis of blockchain forks in Bitcoin. In: Financial Cryptography and Data Security. pp. 84–92. Springer (2019)
26. Sapirshtein, A., Sompolinsky, Y., Zohar, A.: Optimal selfish mining strategies in Bitcoin. In: Financial Cryptography and Data Security. LNCS, vol. 9603, pp. 515–532. Springer (2016)
27. Xiao, Y., Zhang, N., Lou, W., Hou, Y.T.: Modeling the impact of network connectivity on consensus security of proof-of-work blockchain. In: IEEE INFOCOM 2020-IEEE Conference on Computer Communications. pp. 1648–1657. IEEE (2020)
28. Zhang, R., Preneel, B.: Lay down the common metrics: Evaluating proof-of-work consensus protocols' security. In: 40th IEEE Symposium on Security and Privacy (S&P). pp. 1190–1207. IEEE (May 2019)
29. Zhang, R., Zhang, D., Wang, Q., Wu, S., Xie, J., Preneel, B.: NC-Max: Breaking the security-performance tradeoff in Nakamoto consensus. In: The Network and Distributed System Security (NDSS) Symposium (2022)

A Verifying the headstart-γ Relation in Ethereum

We apply the (ρ, t_{hs}) relation we learned in Sect. 4.1 to Ethereum network's data to test its accuracy. We obtain 236 553 fork instances from Etherscan [15], ranging from 2015 to 2022, which includes the then mining power of the competing blocks' miners. Since t_{hs} is not available, we approximate it as the difference between the competing blocks' timestamps. We then group these fork instances by ρ in steps of 0.05, and t_{hs} in steps of one second. We exclude groups with less than 1000 instances to reduce stochastic errors. At last, we plot each group's estimated win rate $\mathrm{P}'_{\text{win}} = \rho + \gamma(t_{\text{hs}}) \cdot (1 - \rho)$ and the actual P_{win}, which is the number of winning cases divided by the total number of cases, in Fig. 8.

The results show that P'_{win} and P_{win} not only follow the same pattern but are also numerically close, confirming the (ρ, t_{hs}) relation we learned, except for two differences. First, P_{win} escalates faster with an increasing t_{hs}. We think this is because the network condition is improved in Ethereum's data, measured until 2022, compared with Bitcoin's pre-compact-block data [25]. When blocks propagate faster, the same positive t_{hs} yields a stronger advantage than before.

Second, for groups with $t_{\text{hs}} = 0$, P'_{win} overestimates P_{win} by roughly 9%. We provide two possible explanations here. First, the P'_{win} formula overestimate the win rate as it ignores the producer of the competing block. In reality, the competing block's producer always works on its own block, rather than with γ probability. Secondly, 5.3% of ties in Ethereum involve three or more blocks, causing P_{win} to be lower than P'_{win} as the latter only covers the two-block case. These phenomena are not significant when $t_{\text{hs}} > 0$ as their effects are mitigated by the advantage of the early announcement.

B The State Transition and Reward Matrices of MDP

The transition and reward matrices are defined in Table 2. The transition matrix describes the candidate states and corresponding probabilities for a given *state* $\times$ *action* combination, and the reward is a two-element tuple (r_u, r_d). Beside, we list an optimal strategy in Table 1, where $\rho = 0.45$, $\gamma = 0$, $l_d, l_u \leq 8$, fork $=$ inactive and $l_w = l_d$. A, W, and O stand for Adopt, Wait, Override respectively.

B.1 Pruning Our MDP

Our model differs from selfish mining MDP [26] in that it introduces a new type of transition called "delay". The "delay" transition depends on the number of blocks mined by both parties. Given the number of mined blocks can be infinite, the MDP's transitions and states become unlimited, making it unsolvable. To overcome this while ensuring accuracy, we prune low-probability transitions.

Based on the probability calculation given in Sect. 3.2, we use ρ_n to denote the probability of a miner with mining power ρ to mine n blocks during the delay. Then, we set a cutoff n^* and approximate mining n blocks as n^* when $n > n^*$, with n^* being the maximum n where $\rho_n \geq 1 \times 10^{-9}$. We will reset n^* to $60 - n$ if the fork length limitation is reached first. Besides, we use n_d^+, n_u^+ to denote positive n_d, n_u to identify cases where a party has mined blocks.

B.2 Detailing Our MDP Transitions

When the action is adopt, the adversary takes the honest fork, so the honest miners get l_u blocks reward. Then, the state transition depends on the producer of the next block, either the adversary w.p. ρ or an honest miner w.p. μ.

When the action is override, the adversary first publishes its blocks to $l_u + 1$ to orphan the honest fork. The subsequent transition and reward are determined by the number of withheld blocks at this point, which is $l_d - l_u - 1$. If there are no withheld blocks, the transitions follow the same pattern as those in adopt, but the adversary wins $l_u + 1$ blocks instead. If there are withheld blocks, the following transition is the passage of withholding time, and the adversary will release one block at the end of it. In addition, the transition is determined by the number of blocks mined by the honest miners during the withholding. If no honest block is mined, the released block will reset the block race again and the adversary wins $l_u + 2$ blocks. Otherwise, the adversary still wins $l_u + 1$ blocks.

For the wait action, we only detail transitions where $l_w \neq 0$ and fork $=$ inactive. Firstly, we ignore transitions where $l_w = 0$ since they follow the same

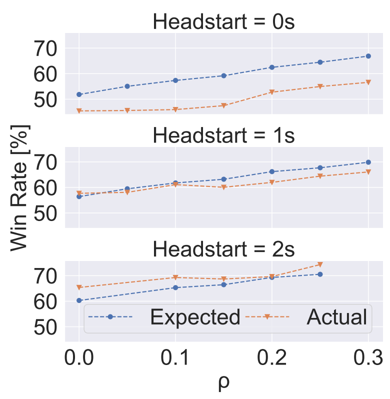

Fig. 8: The win rate about forks in Ethereum.

Table 1: The optimal actions.

l_d \ l_u	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
0	A	W	A	A	A	A	A	A	A
1	W	W	W	W	A	A	A	A	A
2	W	O	W	W	W	W	A	A	A
3	W	W	O	W	W	W	W	A	A
4	W	W	W	O	W	W	W	W	W
5	W	W	W	W	O	W	W	W	W
6	W	W	W	W	W	O	W	W	W
7	W	W	W	W	W	W	O	W	W
8	W	W	W	W	W	W	W	O	W

pattern of adopt. Secondly, transitions where fork = inactive can be inferred by further dividing those under fork = *inactive*, based on the first honest block's location. For instance, the state is $(1, 0, 1, \text{fork})$ and honest miners find the next block. If fork = inactive, all honest miners work on the honest fork so the new block must be on it, resulting in the transited state $(1, 0, 2, \text{inactive})$. Otherwise, the honest mining power is split and the first honest block can be found on either the adversary fork w.p. γ or on the honest fork w.p. $1 - \gamma$, resulting in states $(2, 1, 1, \text{inactive})$ or $(1, 0, 2, \text{inactive})$ respectively.

When describing the wait under inactive, we discuss the transition of the wait with and without the match separately due to their significant difference. We first detail the transition of wait without match. Assuming match is untriggered, the adversary only releases a block at the end of the withholding. The following transition depends on whether this block resets the block race, i.e., whether the lengths of the public part of both sides' fork are identical ($l_u + n_u = l_d - l_w$) before the block release. The expression can be further simplified by removing n_u since it must be zero. A positive n_u implies $l_u < l_d - l_w$, suggesting that the public part of the adversary's fork is longer than the honest fork before the withholding. In this case, all honest miners will accept the adversary's fork, which contradicts our preliminary assumption that a fork existed before the withholding. If this condition is met, the state becomes $(l_w - 1 + n_d, l_w - 1 + n_d, 0, i)$ and the adversary gets a block reward of $l_u + 1$. If not, the state remains $(l_d + n_d, l_w + n_d - 1, l_u + n_u^+, i)$.

Without match, transitions under wait depend on whether an honest block is mined after match. If not ($l_d = l_u + n_u^+$), fork remains active, and the transited state is $(l_d + n_d, n_d, l_u + n_u^+, a)$. Otherwise, transitions split according to the first honest block's location post-match. If it's on the adversary's fork, miners adopt l_d adversary's blocks, leading to $(n_d, n_d, l_u + n_u^+ - l_d, i)$. If the honest block is on the honest fork, the race persists and the state becomes $(l_d + n_d, n_d, l_u + n_u^+, i)$.

Table 2: State transition and reward matrices of MDP.

State × Action	**Condition**	**State**	**Probability**	**Reward**
$(l_d, l_w, l_u, \cdot)$, adopt	$\cdot$	$(1, 1, 0, i)$	ρ	$(0, l_u)$
		$(0, 0, 1, i)$	μ	
$(l_d, l_w, l_u, \cdot)$, override[a]	$l_d = l_u + 1$	$(1, 1, 0, i)$	ρ	$(l_u + 1, 0)$
		$(0, 0, 1, i)$	μ	
	$l_d > l_u + 1$	$(l_d - l_u - 2 + n_d, l_d - l_u - 2 + n_d, 0, i)$	$\rho_{n_d}\mu_0$	$(l_u + 2, 0)$
		$(l_d - l_u - 1 + n_d, l_d - l_u - 2 + n_d, n_u^+, i)$	$\rho_{n_d}\mu_{n_u^+}$	$(l_u + 1, 0)$
(l_d, l_w, l_u, i), wait	$l_w = 0$	$(l_d + 1, l_w + 1, l_u, i)$	ρ	$(0, 0)$
		$(l_d, l_w, l_u + 1, i)$	μ	
	$l_w > 0$ \&\& $(l_d \leq l_u \Vert l_d > l_u + n_u^+)$ \&\& $l_u = l_d - l_w$	$(l_w - 1 + n_d, l_w - 1 + n_d, 0, i)$	$\rho_{n_d}\mu_0$	$(l_u + 1, 0)$
		$(l_d + n_d, l_w + n_d - 1, l_u + n_u^+, i)$	$\rho_{n_d}\mu_{n_u^+}$	$(0, 0)$
	$l_w > 0$ \&\& $(l_d \leq l_u \Vert l_d > l_u + n_u^+)$ \&\& $l_u > l_d - l_w$	$(l_d + n_d, l_w + n_d - 1, l_u + n_u, i)$	$\rho_{n_d}\mu_{n_u}$	$(0, 0)$
	$l_w > 0$ \&\& $l_u < l_d \leq l_u + n_u^+$ \&\& $l_d = l_u + n_u^+$	$(l_d + n_d, n_d, l_u + n_u^+, a)$	$\rho_{n_d}\mu_{n_u^+}$	$(0, 0)$
	$l_w > 0$ \&\& $l_u < l_d \leq l_u + n_u^+$ \&\& $l_d < l_u + n_u^+$	$(n_d, n_d, l_u + n_u^+ - l_d, i)$	$\rho_{n_d}\mu_{n_u^+}\gamma$	$(l_d, 0)$
		$(l_d + n_d, n_d, l_u + n_u^+, i)$	$\rho_{n_d}\mu_{n_u^+}(1-\gamma)$	$(0, 0)$
(l_d, l_w, l_u, a), wait[b]	$l_w = 0$	$(l_d + 1, 1, l_u, a)$	ρ	$(0, 0)$
		$(0, 0, 1, i)$	$\mu\gamma$	$(l_u, 0)$
		$(l_d, 0, l_u + 1, i)$	$\mu(1-\gamma)$	$(0, 0)$
	$l_w > 0$ \&\& $(l_d \leq l_u \Vert l_d > l_u + n_u^+)$	$(l_w - 1 + n_d, l_w - 1 + n_d, 0, i)$	$\rho_{n_d}\mu_0$	$(l_u + 1, 0)$
		$(l_w + n_d, l_w - 1 + n_d, n_u^+, i)$	$\rho_{n_d}\mu_{n_u^+}\gamma$	$(l_u, 0)$
		$(l_d + n_d, l_w - 1 + n_d, l_u + n_u^+, i)$	$\rho_{n_d}\mu_{n_u^+}(1-\gamma)$	$(0, 0)$
	$l_w > 0$ \&\& $l_u < l_d \leq l_u + n_u^+$ \&\& $l_d = l_u + n_u^+$	$(l_w + n_d, n_d, l_w, a)$	$\rho_{n_d}\mu_{n_u^+}\gamma$	$(l_u, 0)$
		$(l_d + n_d, n_d, l_u + n_u^+, a)$	$\rho_{n_d}\mu_{n_u^+}(1-\gamma)$	$(0, 0)$
	$l_w > 0$ \&\& $l_u < l_d \leq l_u + n_u^+$ \&\& $l_d < l_u + n_u^+$	$(n_d, n_d, n_u^+ - l_w, i)$	$\rho_{n_d}\mu_{n_u^+}\gamma^2$	$(l_d, 0)$
		$(l_w + n_d, n_d, n_u^+, i)$	$\rho_{n_d}\mu_{n_u^+}(\gamma - \gamma^2)$	$(l_u, 0)$
		$(n_d, n_d, n_u^+ - l_w, i)$	$\rho_{n_d}\mu_{n_u^+}(\gamma - \gamma^2)$	$(l_d, 0)$
		$(l_d + n_d, n_d, l_u + n_u^+, i)$	$\rho_{n_d}\mu_{n_u^+}(1-\gamma)^2$	$(0, 0)$

[a] feasible only when $l_d > l_u$
[b] feasible only when $l_d \geq l_u$

C Notations

Table 3: Notations in this paper.

Notation	**Description**
ρ, μ	The mining power shares of the D-miner and the undelayed miner, respectively.
γ	The proportion of other mining power that works on the D-miner's fork during a tie.
n_d, n_u	The number of blocks mined by the D-miner and the undelayed miner during the delay, respectively.
n_d^+, n_u^+	The number of blocks mined by the D-miner and the undelayed miner during the delay respectively, under the condition that this number is positive.
D	The maximum delay length in seconds for the D-miner.
T	The average block interval.
λ	The expected number of blocks mined by the entire network during the delay, i.e., D/T.
r_d, r_u	The absolute revenues for the D-miner and the undelayed miner, respectively.
R_d	The relative revenue for the D-miner.
t_{hs}	The headstart of the block producer in seconds, that is, the time span between when a block is mined and when it is announced.
P_{win}	The D-miner's win rate in ties, which is $3.07 \times 10^{-5} t_{\text{hs}} + 0.63$ derived from real-world Bitcoin data.
P'_{win}	The theoretical win rate of the D-miner in ties, which is $\rho + \gamma(1 - \rho)$.
d	The additional delay in seconds that the slow blocks experience beyond the natural propagation latency.
l_d, l_u	The lengths of the forks that belong to the D-miner and the undelayed miner, respectively.
l_w	The number of blocks mined and intentionally withheld by the D-miner.