Optimal Selfish Mining Strategies in Bitcoin

Ayelet Sapirshtein[1], Yonatan Sompolinsky[1], and Aviv Zohar[1,2]

[1] School of Engineering and Computer Science,
The Hebrew University of Jerusalem, Israel
[2] Microsoft Research, Herzliya, Israel
{ayeletsa,yoni_sompo,avivz}@cs.huji.ac.il

Abstract. Bitcoin is a decentralized crypto-currency, and an accompanying protocol, created in 2008. Bitcoin nodes continuously generate and propagate blocks—collections of newly approved transactions that are added to Bitcoin's ledger. Block creation requires nodes to invest computational resources, but also carries a reward in the form of bitcoins that are paid to the creator. While the protocol requires nodes to quickly distribute newly created blocks, strong nodes can in fact gain higher payoffs by withholding blocks they create and selectively postponing their publication. The existence of such *selfish mining* attacks was first reported by Eyal and Sirer [9], who have demonstrated a specific deviation from the standard protocol (a strategy that we name SM1).
In this paper we extend the underlying model for selfish mining attacks, and provide an algorithm to find ϵ-optimal policies for attackers within the model, as well as tight upper bounds on the revenue of optimal policies. As a consequence, we are able to provide lower bounds on the computational power an attacker needs in order to benefit from selfish mining. We find that the *profit threshold* – the minimal fraction of resources required for a profitable attack – is strictly lower than the one induced by the SM1 scheme. Indeed, the policies given by our algorithm dominate SM1, by better regulating attack-withdrawals.
Our algorithm can also be used to evaluate protocol modifications that aim to reduce the profitability of selfish mining. We demonstrate this with regard to a suggested countermeasure by Eyal and Sirer, and show that it is slightly less effective than previously conjectured. Next, we gain insight into selfish mining in the presence of communication delays, and show that, under a model that accounts for delays, the profit threshold vanishes, and even small attackers have incentive to occasionally deviate from the protocol. We conclude with observations regarding the combined power of selfish mining and double spending attacks.

1 Introduction

In a recent paper, Eyal and Sirer [9] have highlighted a flaw in the incentive scheme in Bitcoin. Given that most of the network follows the "standard" Bitcoin protocol, a single node (or a pool) which possesses enough computational resources or is extremely well connected to the rest of the network can increase its expected rewards by deviating from the protocol. While the standard Bitcoin

protocol requires nodes to immediately publish any block that they find to the rest of the network, Eyal and Sirer have shown that participants can selfishly increase their revenue by selectively withholding blocks. Their strategy, which we denote SM1, thus shows that Bitcoin as currently formulated is not incentive compatible.

On the positive side, SM1 (under the model of Eyal and Sirer) becomes profitable only when employed by nodes that posses a large enough share of the computational resources, and are sufficiently well connected to the rest of the network.[3] It is important to note, however, that SM1 is not the optimal best-response to honest behaviour, and situations in which SM1 is not profitable may yet have other strategies that are better than strict adherence to the protocol. Our goal in this paper is to better understand the conditions under which Bitcoin is resilient to selfish mining attacks. To this end, we must consider other possible deviations from the protocol, and to establish bounds on their profitability.

The role of incentives in Bitcoin should not be underestimated: Bitcoin transactions are confirmed in batches, called *blocks* whose creation requires generating the solution to computationally expensive proof-of-work "puzzles". The security of Bitcoin against the reversal of payments (so-called double spending attacks) relies on having more computational power in the hands of honest nodes. Block creation (which is also known as *mining*), is rewarded in bitcoins that are given to the block's creator. These rewards incentivize more honest participants to invest additional computational resources in mining, and thus support the security of Bitcoin.

When all miners follow the Bitcoin protocol, a single miner's share of the payoffs is equal to the fraction of computational power that it controls (out of the computational resources of the entire network). However, Selfish mining schemes allow a strong attacker to increase its revenue at the expense of other nodes. This is done by exploiting the conflict-resolution rule of the protocol, according to which only one chain of blocks can be considered valid, and only blocks on the valid chain receive rewards; the attacker creates a deliberate fork, and (sometimes) manages to force the honest network to abandon and discard some of its blocks.

The consequences of selfish mining attacks are potentially destructive to the Bitcoin system. A successful attacker becomes more profitable than honest nodes, and is able to grow steadily.[4] It may thus eventually drive other nodes out of the system. Profits from selfish mining increase as more computational power is held by the attacker, making its attack increasingly effective, until it eventually holds over 50% of the computational resources in the network. At this point, the attacker is able to collect all block rewards, to mount successful double spending attacks at will, and to block any transaction from being processed (this is known as the 50% attack).

---

[3] This can partly explain why selfish mining attacks have not been observed in the Bitcoin network thus far.

[4] Growth is achieved either by buying more hardware, in the case of a single attacker, or by attracting more miners, in the case of a pool.

We summarize the contributions of this paper as follows:

1. We provide an efficient algorithm that computes an ϵ-optimal selfish mining policy for any $\epsilon > 0$, and for any parametrization of the model in [9] (i.e., one that maximizes the revenue of the attacker up to an error of ϵ, given that all other nodes are following the standard Bitcoin protocol). We prove the correctness of our algorithm and analyze its error bound. We further verify all strategies generated by the algorithm in a selfish mining simulator that we have designed to this end.

2. Using our algorithm we show that, indeed, there are selfish mining strategies that earn more money and are profitable for smaller miners compared to SM1. The gains are relatively small (see Fig. 1 below). This can be seen as a positive result, lower bounding the amount of resources needed for a profitable attacker.

3. Our technique allows us to evaluate different protocol modifications that were suggested as countermeasures for selfish mining. We do so for the solution suggested by Eyal and Sirer, in which miners that face two chains of equal weight choose the one to extend uniformly at random. We show that this modification unexpectedly enhances the power of medium-sized attackers, while limiting strong ones, and that unlike previously conjectured, attackers with less than 25% of the computational resources can still gain from selfish mining.

4. We show that in a model that accounts for the delay of block propagation in the network, the threshold vanishes: there is always a successful selfish mining strategy that earns more than honest mining, regardless of the size of the attacker.

5. We discuss the interaction between selfish mining attacks and double spending attacks. We demonstrate how any attacker for which selfish mining is profitable can execute double spending attacks bearing no costs. This sheds light on the security analysis of Satoshi Nakamoto [14], and specifically, on the reason that it cannot be used to show high attack costs, and must instead only bound the probability of a successful attack.

Below, we depict the results of our analysis, namely, the revenue achieved by optimal policies compared to that of SM1 as well as the profit threshold of the protocol. In the following, α stands for the attacker's relative hashrate, and γ is a parameter representing the communication capabilities of the attacker: the fraction of nodes to which it manages to send blocks first in case of a block race (see Section 2 for more details). Figure 1 depicts the revenue of an attacker under three strategies: Honest mining, which adheres to the Bitcoin protocol, SM1, and the optimal policies obtained by our algorithm. The three graphs correspond to $\gamma = 0, 0.5, 1$. We additionally illustrate the curve of $\alpha/(1-\alpha)$, which is an upper bound on the attacker's revenue, achievable only when $\gamma = 1$ (see Section 3). Figure 2 depicts the profit threshold for each γ: If the attacker's α is below the threshold then Honest mining is the most profitable strategy. For comparison, we depict the thresholds induced by SM1 as well.

(a) $\gamma = 0$

(b) $\gamma = 0.5$

(c) $\gamma = 1$
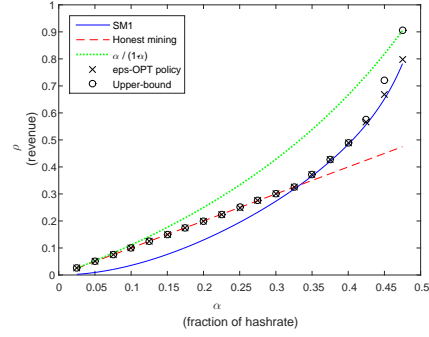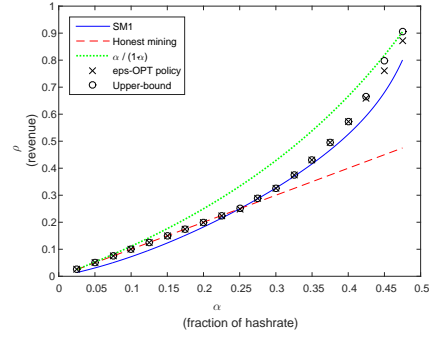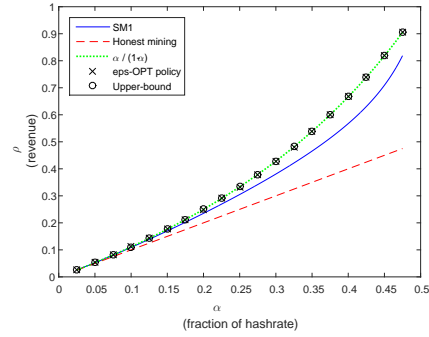
Fig. 1. The ϵ-optimal revenue and the computed upper bound, as a function of the attacker's hashrate α, compared to SM1, honest mining, and to the hypothetical bound provided in Section 3. The graphs differ in the attacker's communication capability, γ, valued 0, 0.5, and 1. The gains of the ϵ-optimal policies are very close to the computed upper bound, except when α is close to 0.5, in case which the truncation-imposed loss is apparent. See also Table 2.
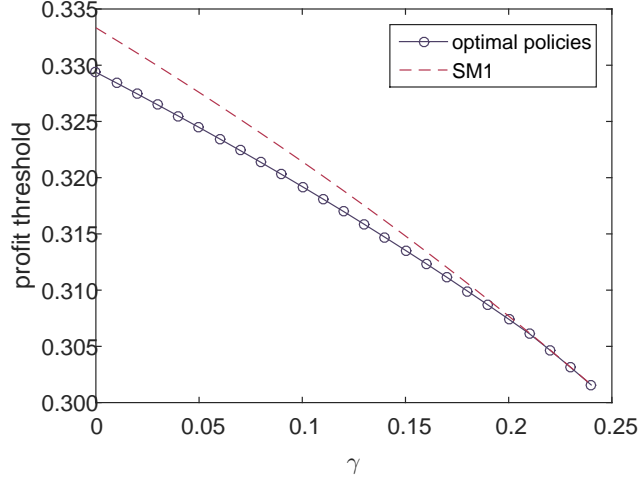
Fig. 2. The profit thresholds induced by optimal policies, and by SM1, as a function of γ. Thresholds at higher γ values match that of SM1 (but still, optimal strategies for these values earn more than SM1, once above the threshold).

The remainder of the paper is structured as follows: We begin by presenting our model, based principally on Eyal and Sirer's [9] (Section 2). Section 3 shows a theoretical bound on the attacker's revenue. In Section 4 we describe our algorithm to find optimal policies and values. In Section 5 we discuss more results, e.g., the optimal policies. Section 6 analyzes selfish mining in networks with delays. Section 7 discusses the interaction between selfish mining and double spending. We conclude with discussing related work (Section 8).

2   Model

We follow and extend the model of [9], to explicitly consider all actions available to the attacker at any given point in time.

We assume that the attacker controls a fraction α of the computational power in the network, and that the honest network thus has a $(1 - \alpha)$ fraction. Communication of newly created blocks is modeled to be much faster than block creation, so no blocks are generated while others are being transmitted.[5]

Blocks are created in the network according to a Poisson process with rate λ. Every new block is generated by the attacker with probability α, or by the honest network with probability $(1 - \alpha)$. The honest network follows the Bitcoin protocol, and always builds its newest block on top of the longest known chain.

---

[5] This is justified by Bitcoin's 10 minute block creation interval which is far greater than the propagation time of blocks in the network. This assumption is later removed when we consider networks with delay.

Once an honest node adopts a block, it will discard it only if a strictly longer competing chain exists. Ties are thus handled by each node according to the order of arrival of blocks. Honest nodes immediately broadcast blocks that they create.

Blocks generally form a tree structure, as each block references a single predecessor (with the exception of the first block that is called the genesis block). Since the honest nodes adopt the longest chain, blocks generate rewards for their creator only if they are eventually part of the longest chain in the block tree (all blocks can be considered revealed eventually).

To model the communication capabilities of the attacker, we assume that whenever it learns that a block has been released by the network, it is able to transmit an alternative block which will arrive *first* at nodes that possess a fraction γ of the computational power of the honest network (the attacker must have prepared this block in advance in order to be able to deliver it quickly enough). Thus, if the network is currently propagating a block of height h, and the attacker has a competing block of the same height, it is able to get $\gamma \cdot (1 - \alpha)$ of the computational power (owned by honest nodes) to adopt this block.

The attacker does not necessarily follow the Bitcoin protocol. Rather, at any given time t, it may choose to invest computational power in creating blocks that extend any existing block in history, and may withhold blocks it has created for any amount of time. A general selfish mining strategy dictates, therefore, two key behaviours: which block the attacker attempts to extend at any time t, and which blocks are released at any given time. However, given that all block creation events are driven by memoryless processes and that broadcast is modeled as instantaneous, any rational decision made by the attacker may only change upon the creation of a new block. The mere passage of time without block creation does not otherwise alter the expected gains from future outcomes.[6] Accordingly, we model the entire decision problem faced by an attacker using a discrete-time process in which each time step corresponds to the creation of a block. The attacker is thus asked to decide on a course of action immediately after the creation of each block, and this action is pursued until the next event occurs.

Instead of directly modeling the primitive actions of block extension and publication on general block trees, we can limit our focus to "reasonable" strategies where the attacker maintains a *single* secret branch of blocks that diverged from the network's chain at some point. (We show that this limitation is warranted and that this limited strategy space still generates optimal attacks in Appendix A). Blocks before that point are agreed upon by all participants. Accordingly, we must only keep track of blocks that are after the fork, and of the accumulated reward up to the fork. We denote by a the number of blocks that have been built by the attacker after the latest fork, and by h the number of those built by honest nodes.

Formally, if all other participants are following the standard protocol, the attacker faces a single-player decision problem of the form $M := \langle S, A, P, R \rangle$,

---

[6] See Section 6 for the implication of delayed broadcasting.

where S is the state space, A the action space, P a stochastic transition matrix that describes the probability of transitioning between states, and R the reward matrix. Though similar in structure, we do not regard M as an MDP, since the objective function is nonlinear: The player aims to maximize its share of the accepted blocks, rather than the absolute number of its own accepted ones; its goal is to have a greater return-on-investment than its counterparts.[7]

Actions. We begin with the description of the action space A, which will motivate the nontrivial construction of the state space.

- **Adopt**. The action *adopt* is always feasible, and represents the attacker's acceptance of the honest network's chain. The a blocks in the attacker's current chain are discarded.
- **Override**. The action *override* represents the publication of the attacker's blocks, and is feasible whenever $a > h$.
- **Match**. This action represents the case where the most recent block was built by the honest network, and the attacker now publishes a conflicting block of the same height. This action is not always feasible (the attacker must have a block prepared in advance to execute such a race). The state-space explicitly encodes the feasibility status of this action (see below).
- **Wait**. Lastly, the *wait* action, which is always feasible, implies that the attacker does not publish new blocks, but keeps working on its branch until a new block is built.

State Space. The state space, denoted S, is defined by 3-tuples of the form $(a, h, fork)$. The first two entries represent the lengths of the attacker's chain and the honest network's chain, built after the latest fork (that is, above the most recent block accepted by all). The field $fork$ obtains three values, dubbed *irrelevant*, *relevant* and *active*. State of the form $(a, h, relevant)$ means that the previous state was of the form $(a, h-1, \cdot)$; this implies that if $a \geq h$, *match* is feasible. Conversely, $(a, h, irrelevant)$ denotes the case where the previous state was $(a - 1, h, \cdot)$, rendering *match* now ineffective, as all honest nodes received already the h'th block. The third label, *active*, represents the case where the honest network is already split, due to a previous *match* action; this information affects the transition to the next state, as described below. *We will refer to states as (a, h) or $(a, h, \cdot)$, in contexts where the $fork$ label plays no effective role.*

Transition and Reward Matrices. In order to keep the time averaging of rewards in scale, every state transition corresponds to the creation of a new block. The initial state X_0 is $(1, 0, irrelevant)$ w.p. α or $(0, 1, irrelevant)$ w.p. $(1 - \alpha)$. Rewards are given as elements in $\mathbb{N}^2$, where the first entry represents blocks of the attacker that have been accepted by all parties, and the second one, similarly, for those of the honest network.

The transition matrix P and reward matrix R are succinctly described in Table 1. Largely, an *adopt* action "resets" the game, hence the state following

---

[7] Another possible motivation for this is the re-targeting mechanism in Bitcoin. When the block creation rate in the network is constant, the adaptive re-targeting implies that the attacker will also increase its absolute payoff, in the long run.

it has the same distribution as X_0; its immediate reward is h in the coordinate corresponding to the honest network. An *override* reduces the attacker's secret chain by $h+1$ blocks, which it publishes, and which the honest network accepts. This bestows a reward of $h + 1$ blocks to the attacker. The state following a *match* action depends on whether the next block is created by the attacker (α), by honest nodes working on their branch of the chain ($(1 - \gamma) \cdot (1 - \alpha)$), or by an honest node which accepted the sub-chain that the attacker published ($\gamma \cdot (1 - \alpha)$). In the latter case, the attacker has effectively overridden the honest network's previous chain, and is awarded h accordingly.

Table 1. A description of the transition and reward matrices P and R in the decision problem M. The third column contains the probability of transiting from the state specified in the left-most column, under the action specified therein, to the state on the second one. The corresponding two-dimensional reward (the reward of the attacker and that of the honest nodes) is specified on the right-most column.

State $\times$ Action	State	Probability	Reward
$(a, h, \cdot), adopt$	$(1, 0, irrelevant)$	α	$(0, h)$
	$(0, 1, irrelevant)$	$1 - \alpha$	
$(a, h, \cdot), override^{\dagger}$	$(a - h, 0, irrelevant)$	α	$(h + 1, 0)$
	$(a - h - 1, 1, relevant)$	$1 - \alpha$	
$(a, h, irrelevant), wait$	$(a + 1, h, irrelevant)$	α	$(0,0)$
$(a, h, relevant), wait$	$(a, h + 1, relevant)$	$1 - \alpha$	$(0,0)$
$(a, h, active), wait$	$(a + 1, h, active)$	α	$(0,0)$
$(a, h, relevant), match^{\ddagger}$	$(a - h, 1, relevant)$	$\gamma \cdot (1 - \alpha)$	$(h, 0)$
	$(a, h + 1, relevant)$	$(1 - \gamma) \cdot (1 - \alpha)$	$(0,0)$

†feasible only when $a > h$
‡feasible only when $a \geq h$

Objective Function. As explained in the introduction, the attacker aims to maximize its relative revenue, rather than its absolute one as usual in MDPs. Let π be a policy of the player; we will write $\pi(a, h, fork)$ for the action that π dictates be taken at state $(a, h, fork)$. Denote by X_t^{π} the state visited by time t under π, and let $r(x, y, \pi) = (r^1(x, y, \pi), r^2(x, y, \pi))$ be the immediate reward from transiting from state x to state y, under the action dictated by π. X_t^{π} will denote the t'th state that was visited. We will abbreviate $r_t(X_t^{\pi}, X_{t+1}^{\pi}, \pi)$ and write simply $r_t(\pi)$ or even r_t, when context is clear. The objective function of the player is its *relative* payoff, defined by

$$REV := \mathbb{E} \left[\liminf_{T \to \infty} \frac{\sum_{t=1}^{T} r_t^1(\pi)}{\sum_{t=1}^{T} \left(r_t^1(\pi) + r_t^2(\pi) \right)} \right]. \tag{1}$$

We will specify the parameters of REV depending on the context (e.g., $REV(\pi, \alpha, \gamma)$, $REV(\pi)$, $REV(\alpha)$), and will occasionally denote the value of REV by ρ. In addition, for full definiteness of REV, we rule out pathological

behaviours in which the attacker waits forever—formally, the expected time for the next non-null action of the attacker must be finite.

Honset Mining and SM1. We now define two policies of prime interest to this paper. Honest mining is the unique policy which adheres to the protocol at every state. It is defined by

$$\text{honest mining}\,(a, h, \cdot) = \left\{ \begin{array}{ll} adopt & h > a \\ override & a > h \end{array} \right\}, \tag{2}$$

and *wait* otherwise. Notice that under our model, $REV(\text{honest mining}, \alpha, \gamma) = \alpha$ for all γ.[8] Eyal and Sirer's selfish mining strategy, SM1, can be defined as

$$SM1\,(a, h, \cdot) := \left\{ \begin{array}{ll} adopt & h > a \\ match & h = a = 1 \\ override & h = a - 1 \geq 1 \\ wait & \text{otherwise} \end{array} \right\}. \tag{3}$$

Profit threshold. Keeping the attacker's connectivity capabilities (γ) fixed, we are interested in the minimal α for which employing dishonest mining strategies becomes profitable. We define the profit threshold by:

$$\hat{\alpha}(\gamma) := \inf_{\alpha} \left\{ \exists \pi \in A : \; REV(\pi, \alpha, \gamma) > REV(\text{honest mining}, \alpha, \gamma) \right\}. \tag{4}$$

3   A Simple Upper Bound

The mechanism implied by the longest-chain rule leads to an immediate bound on the attacker's relative revenue. Intuitively, we observe that the attacker cannot do better than utilizing every block it creates to override one block of the honest network. The implied bound is provided here merely for general insight—it is usually far from the actual maximal revenue.

Proposition 1. *For any π, $REV(\pi, \alpha, \gamma) \leq \frac{\alpha}{1-\alpha}$. Moreover, this bound is tight, and achieved when $\gamma = 1$.*

See Appendix B for the proof.

4   Solving for the Optimal Policy

Finding an optimal policy is not a trivial task, as the objective function (1) is nonlinear, and depends on the entire history of the game. To overcome this we introduce the following method. We assume first that the optimal value of the objective function is ρ, then construct an infinite un-discounted average reward MDP (with "standard" linear rewards), compute its optimal policy (using standard MDP solution techniques), and if the reward of this policy is zero then it is optimal also in the original decision problem M. We elaborate on this approach below.

---

[8] Indeed, in networks without delay, honest mining is equivalent to the policy $\{\,adopt$ if $(a, h) = (0, 1)\,;\; override$ if $(a, h) = (1, 0)\,\}$, as these are the only reachable states. Delays allow other states to be reached, and will be covered in Section 6.

4.1 Method

For any $\rho \in [0, 1]$, define the transformation $w_\rho : \mathbb{N}^2 \to \mathbb{Z}$ by $w_\rho(x, y) := (1 - \rho) \cdot x - \rho \cdot y$. Define the MDP $M_\rho := \langle S, A, P, w_\rho(R) \rangle$; it shares the same state space, actions, and transition matrix as M, while M's immediate rewards matrix is transformed according to w_ρ. For any admissible policy π denote by v_ρ^π the expected mean revenue under π, namely,

$$v_\rho^\pi = \mathbb{E} \left[\liminf_{T \to \infty} \frac{1}{T} \sum_{t=1}^T w_\rho(r_t(\pi)) \right], \tag{5}$$

and by

$$v_\rho^* = \max_{\pi \in A} \left\{ v_\rho^\pi \right\} \tag{6}$$

the value of M_ρ.[9] Our solution method is based on the following proposition:

Proposition 2. *1. If for some $\rho \in [0, 1]$, $v_\rho^* = 0$, then any policy π^* obtaining this value (thus maximizing v_ρ^π) also maximizes REV, and $\rho = REV(\pi_\rho^*)$.*
2. v_ρ^* is monotonically decreasing in ρ.

Following these observations we can utilize the family M_ρ to obtain an optimal policy: We perform a simple search for a ρ such that the optimal solution of M_ρ has a value of 0. Since v_ρ^* is monotonically decreasing, this search can be done efficiently, using binary search. In practice, our algorithm relies on a variation of Proposition 2, which will be proven formally in Appendix C.

Due to the fact that the search domain is continuous, practically, one would need to halt the search at a point that is sufficiently close to the actual value, but never exact. Moreover, in practice, MDP solvers can solve only finite state space MDPs, and even then only to a limited degree of accuracy. Our algorithm copes with these computational limitations by using finite MDPs as bounds to the original problem, and by analyzing the potential error that is due to inexact solutions.

4.2 Translation to Finite MDPs

We now introduce two families of MDPs, closely related to the family M_ρ: Fix some $T \in \mathbb{N}$. We define an *under-paying* MDP, M_ρ^T, which differs from M_ρ only in states where $\max \{a, h\} = T$, in which it only allows only for the *adopt* action. We denote this modified action space by A^T. Clearly, the player's value in M_ρ^T lower bounds that in M_ρ, since in the latter the attacker might be able to do better by not adopting in the truncating states. Consequently, this MDP can only be used to upper bound the threshold (in a way described below).

To complete the picture we need to bound the optimal value from above, and we do so by constructing an *over-paying* MDP, N_ρ^T. This MDP shares the

---

[9] The equivalence of this formalization of the value function and alternatives in which the order of expectation and limit is reversed is discussed in [4].

same constraint as M_ρ^T, yet it compensates the attacker in the states where $\max\{a, h\} = T$, by granting it a reward greater than what it could have gotten in the un-truncated process: When $T = a \geq h$, the attacker is awarded

$$(1 - \rho) \cdot \frac{\alpha \cdot (1 - \alpha)}{(1 - 2 \cdot \alpha)^2} + \frac{1}{2} \cdot \left(\frac{a - h}{1 - 2 \cdot \alpha} + a + h \right). \tag{7}$$

On the other hand, when $T = h \geq a$, it is awarded

$$\left(1 - \left(\frac{\alpha}{1 - \alpha} \right)^{h-a} \right) \cdot (-\rho \cdot h) + \left(\frac{\alpha}{1 - \alpha} \right)^{h-a} \cdot (1 - \rho) \cdot \left(\frac{\alpha \cdot (1 - \alpha)}{(1 - 2 \cdot \alpha)^2} + \frac{h - a}{1 - 2 \cdot \alpha} \right).$$

Denote by v_ρ^{T*} and u_ρ^{T*} the average-sum optimal values of the under-paying M_ρ^T and the over-paying N_ρ^T, respectively (i.e., the expected liminf of the average value, for the best policy in A^T, similar to (5)-(6)). The following proposition formalizes the bounds provided by the over-paying and under-paying MDPs:

Proposition 3. *For any $T \in \mathbb{N}$, if $v_\rho{}^* \geq 0$ then $u_\rho^{T*} \geq v_\rho{}^* \geq v_\rho^{T*}$. Moreover, these bounds are tight: $\lim\limits_{T \to \infty} u_\rho^{T*} - v_\rho^{T*} = 0$.*

The proof is differed to the appendix. Having introduced these MDP families, we are now ready to present an algorithm which utilizes them to obtain upper and lower bounds on the attacker's profit.

4.3 Algorithm

Algorithm 1
Input: α and γ, a truncation parameter $T_0 \in \mathbb{N}$, and error parameters $0 < \epsilon < 8 \cdot \alpha$, $0 < \epsilon' < 1$

1. low $\leftarrow 0$, high $\leftarrow 1$
2. do
3. $\rho \leftarrow (low + high)/2$
4. $(\pi, v) \leftarrow mdp\_solver(M_\rho^{T_0}, \epsilon/8)$
5.   if $(v > 0)$
6.   low $\leftarrow \rho$
7.   else
8.   high $\leftarrow \rho$
9. while($high - low \geq \epsilon/8$)
10. lower-bound $\leftarrow (\rho - \epsilon)$
11. lower-bound-policy $\leftarrow \pi$
12. $\rho' \leftarrow \max\{low - \epsilon/4, 0\}$
13. $(\pi, u) \leftarrow mdp\_solver(N_{\rho'}^{T_0}, \epsilon')$
14. upper-bound $\leftarrow (\rho' + 2 \cdot (u + \epsilon'))$

The algorithm initializes the search segment to be $[0, 1]$ (line 1) and begins a binary search: ρ is assigned the middle point of the search segment (line 3), and the algorithm outputs an $\epsilon/8$-optimal policy of $M_\rho^{T_0}$ and its value (line 4). The loop halts if the size of the search segment is smaller then $\epsilon/8$. Otherwise, it restricts the search to the larger half of the segment, if the value is positive (line 6), or to the lower half, in case it is negative (line 8). This essentially represents a binary search for an approximate-root of $v_\rho^{T_0}$, which is a monotonically decreasing function of ρ. The algorithm outputs $(\rho - \epsilon)$ as a lower bound on the player's relative revenue, and π as an ϵ-optimal policy. These assertions are formalized in the proposition below:

Proposition 4. *For any $T_0 \in \mathbb{N}$ and $\epsilon > 0$, Algorithm 1 halts, and its output (ρ, π) satisfies: $\left| \rho - REV(\pi) \right| < \epsilon$ and $\left| \rho - \max_{\pi' \in A^{T_0}} \{REV(\pi')\} \right| < \epsilon$.*

The second part of the algorithm (lines 12-14) computes an ϵ'-optimal policy for the over-paying MDP $N_{\rho'}^{T_0}$, for $\rho' = (low - \epsilon/4)^+$ (using the value assigned last to low). If u is the outputted value, the algorithm returns $\rho + 2 \cdot (u + \epsilon')$ as an upper bound to the player's revenue (line 14).

Proposition 5. *If u and ρ' are the outcome of the computation in Algorithm 1, lines 12-13, then $\rho' + 2 \cdot (u + \epsilon') > \max_{\pi' \in A} \{REV(\pi')\}$.*

Both propositions are proved in Appendix C.

4.4   Profit threshold Calculation

The threshold $\hat{\alpha}(\gamma)$ marks the minimal computational power an attacker needs in order to gain more than its fair share (see Section 2). It is crucial in assessing the system's resilience: An attacker above the threshold is able to receive increased returns on its investment, to grow steadily in resources,[4] and eventually to push other nodes out of the game. The system is safe against such a destructive dynamic if all miners hold less than $\hat{\alpha}(\gamma)$ of the computational power.

Fix γ. A simple method allows us to lower bound the threshold: We first modify the action space of the overpaying N_α^T so as to disable the option of honest mining; technically, this is done by removing *override* from the feasible actions in $(1, 0)$ and then, separately, removing *adopt* in $(0, 1)$. Denote this modified MDP by $\widehat{N_\alpha^T}$. Then we solve $\widehat{N_\alpha^T}$, for some α, error parameter ϵ, and truncation T. If the *mdp_solver* returns a value smaller than $(-\epsilon)$ (both for when *override* is disabled in $(1, 0)$ and when *adopt* is disabled in $(0, 1)$), we are assured that honest mining is optimal in the original setup. We perform a search for the maximal α satisfying this requirement, i.e., $\hat{\alpha}(\gamma)$, in a fashion similar to the search in Algorithm 1.

Corollary 6. *Fix γ and α. If u is the value returned by $mdp\_solver(\widehat{N_\alpha^T}, \epsilon)$, and $u \le -\epsilon$, then honest mining is optimal for α. In other words, $\hat{\alpha}(\gamma) \ge \alpha$.*

5 Results

5.1 Optimal Values

We ran Algorithm 1 for γ from $\{0, 0.5, 1\}$, with various values of α, using an MDP solver for MATLAB (an implementation of the relative value iteration algorithm developed by Chadés et al. [6]). The error parameter ϵ was set to be 10^{-5} and the truncation was set to $T = 75$. The values of ρ returned by the algorithm, for $\gamma = 0, 0.5, 1$, are depicted in Figure 1 above. Additionally, some values for $\gamma = 0$ appear in Table 2, computed for parameters $T = 95$ and $\epsilon = 10^{-5}$. The results demonstrate a rather mild gap between the attacker's optimal revenue and the revenue of SM1. In addition, the graphs depict the upper bound on the revenue provided in Section 3; as we stated there, the bound is obtained when $\gamma = 1$, which is observed clearly in the corresponding graph.

Table 2. The revenue of the attacker under SM1 and under the ϵ-OPT policies, compared to the computed upper bound, for various α and with $\gamma = 0$.

α	$SM1$	ϵ-OPT	Upper-Bound
1/3	1/3	0.33705	0.33707
0.35	0.36650	0.37077	0.37079
0.375	0.42118	0.42600	0.42604
0.4	0.48372	0.48866	0.48904
0.425	0.55801	0.56808	0.57226
0.45	0.65177	0.66891	0.70109
0.475	0.78254	0.80172	0.90476

5.2 Optimal Policies

Below we illustrate two examples of the behaviour of the ϵ-optimal policies returned by the algorithm. The policies are described by tables, with the row index corresponding to a and the columns to h. The table-entry (a, h) contains three characters, specifying the actions to be taken in states $(a, h, irrelevant)$, $(a, h, relevant)$, and $(a, h, active)$ correspondingly. Table 3 contains a description of an optimal policy, for an attacker with $\alpha = 0.45, \gamma = 0.5$. Table 4 describes optimal actions for the setup $\alpha = 1/3, \gamma = 0$. Notice that in the latter the *match* action is irrelevant, which allows us to regard in the second table only states with $fork = irrelevant$. In both tables only a subset of the states is depicted, the whole space being infinite.

To illustrate how Table 3 should be read, consider entry $(a, h) = (3, 3)$, for instance. The string "$wm*$" in this entry reads: "in case a fork is *irrelevant* (that is, the previous state was $(2, 3)$), *wait*; in case it is *relevant* (the previous state was $(3, 2)$), *match*; the case where a fork is already *active* is not reachable".

Table 3. The optimal policy for an attacker with $\alpha = 0.45$ and $\gamma = 0.5$, for states $(a, h, \cdot)$ with $a, h \leq 8$. The rows index the attacker's chain length (a), and the columns the honest network's (h). The three characters in each entry represent the action to be taken if $fork = irrelevant$, $relevant$, or $active$. 'a', 'o', 'm', and 'w' stand for *adopt*, *override*, *match*, and *wait*, respectively, while '*' represents an unreachable state.

a \ h	0	1	2	3	4	5	6	7	8
0	***	*a*	***	***	***	***	***	***	***
1	w**	*m*	a**	***	***	***	***	***	***
2	w**	*mw	*m*	w**	a**	***	***	***	***
3	w**	*mw	*mw	wm*	w**	a**	***	***	***
4	w**	*mw	*mw	omw	wm*	w**	w**	a**	***
5	w**	*mw	*mw	*mw	omw	wm*	w**	w**	a**
6	w**	*mw	*mw	*mw	*mw	omw	wm*	w**	w**
7	w**	*mw	*mw	*mw	*mw	*mw	ooo	w**	w**
8	w**	*ww	*mw	*mw	*mw	*mw	*m*	oo*	w**

Table 4. The optimal policy for an attacker with $\alpha = 0.35$ and $\gamma = 0$. The table describes the actions only for states of the form $(a, h, irrelevant)$ with $a, h \leq 8$. (See previous caption)

a \ h	0	1	2	3	4	5	6	7	8
0	*	a	*	*	*	*	*	*	*
1	w	w	w	a	*	*	*	*	*
2	w	o	w	w	a	*	*	*	*
3	w	w	o	w	w	a	*	*	*
4	w	w	w	o	w	w	w	a	*
5	w	w	w	w	o	w	w	w	a
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

Looking into these optimal policies we see they differ from SM1 in two ways: First, they defer using *adopt* in the upper triangle of the table, if the gap between h and a is not too large, allowing the attacker to "catch up from behind". Thus, apart from block withholding, an optimal attack may also contain another feature: attempting to catch up with the longer public chain from a disadvantage. This implies that the attacker violates the longest-chain rule, a result which counters the claim that the longest-chain rule forms a Nash equilibrium (see [11], and discussion in Section 8)

Secondly, they utilize *match* more extensively, effectively overriding the honest network's chain (w.p. γ) using one block less.

5.3 Thresholds

Following the method described in Section 4.4, we are able to introduce lower bounds for the profit thresholds. Figure 2 depicts the thresholds induced by optimal policies, compared to that induced by SM1. The results demonstrate some cutback of the thresholds, when considering policies other than SM1.

5.4 Evaluation of Protocol Modifications

Several protocol modifications have been suggested to counter selfish mining attacks. It is important to provably verify the merit of such suggestions. This can be done by adapting our algorithm to the MDPs induced by these modifications. Below we demonstrate this with respect to the rule suggested by Eyal and Sirer. According to the Bitcoin protocol, a node which receives a chain of length equal to that of the chain it currently adopts, ought to reject the new chain. Eyal and Sirer suggest to instruct nodes to accept the new chain with probability 1/2. We refer to it below as "uniform tie breaking".

The immediate effect of this modification is that it restricts the efficiency of the *match* action to 1/2, even when the attacker's communication capabilities correspond to $\gamma > 1/2$. Admittedly, this limits the power of strongly communicating attackers, and thus guarantees a positive lower bound on the threshold for profitability of SM1 (which was 0, when $\gamma = 1$). On the other hand, it has the apparent downside of enhancing the power of poorly communicating attackers, that is, it allows an attacker to *match* with a success-probability 1/2 even if its "real" γ is smaller than 1/2.

Unfortunately, our results show that this protocol enhances the profit of some attackers from deviations. For example, by applying Algorithm 1 to the setup induced by uniform tie breaking, we found that attackers in the range $\{\gamma = 0.5 \, , \, 0.2321 < \alpha < 0.5\}$ benefit from this modification. In particular, the profit threshold deteriorates from 0.25 to 0.2321. Figure 3 demonstrates this by comparing the attacker's optimal revenue under the uniform tie breaking protocol with the optimal revenue under the original protocol. The dominating policy is described in Table 5.

The intuition behind this result is as follows: Under uniform tie breaking, two chains of equal length will be mined equally regardless of the passage of time

between the transmission of their last blocks. This allows an attacker to perform *match* even if it did not have a block prepared in advance, thereby granting it additional chances to catch up from behind. Deviation from the longest-chain rule thus becomes even more tempting.
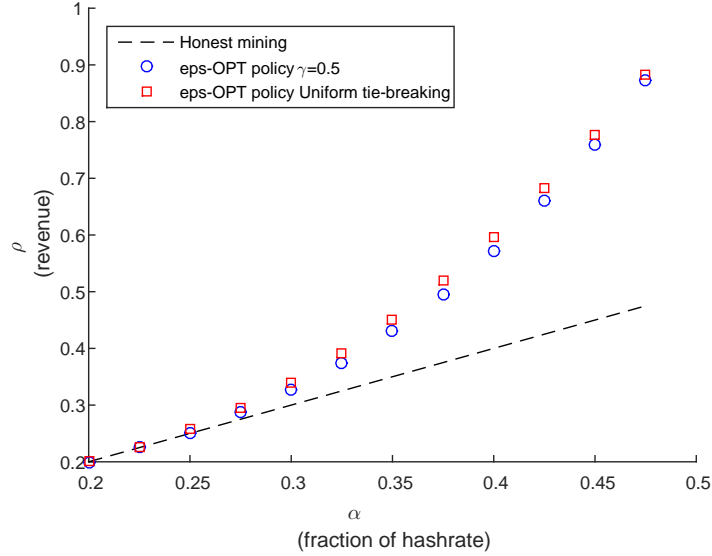

Fig. 3. The attacker's optimal revenue under uniform tie breaking, compared to that under the original protocol (with $\gamma = 1/2$) and to honest mining.

5.5 Simulations

In order to verify the results above we built a selfish mining simulator which we implemented in Java. We ran the simulator for various values of α and γ (as in the figures above), where the attacker follows the policies generated by the algorithm. Each run was performed for 10^7 rounds (block creation events). The relative revenue of the attacker matched the revenues returned by the algorithm, up to an error of at most $\pm 10^{-6}$.

6 A Model that Considers Delays

So far, our model assumed that no new block is created until all preceding published blocks arrived at all nodes. In reality, there are communication delays between nodes in the network, including between the attacker and others. Thus, instead of modeling the attacker's communication capabilities via the parameter

Table 5. The optimal policy for an attacker with $\alpha = 0.25$, under the "50-50" protocol modification suggested in [9]. Only states $(a, h, \cdot)$ with $a, h \leq 8$ are depicted. This policy outperforms honest mining.

	0	1	2	3	4	5	6	7	8
0	*w*	aa*	***	***	***	***	***	***	***
1	w**	*m*	w**	a**	***	***	***	***	***
2	w**	*oo	m**	w**	a**	***	***	***	***
3	w**	*w*	*oo	m**	w**	a**	***	***	***
4	w**	ww*	*w*	*oo	m**	w**	w**	a**	***
5	w**	ww*	ww*	*w*	*oo	m**	w**	w**	a**
6	w**	ww*	ww*	ww*	*w*	*oo	m**	w**	w**
7	w**	ww*	ww*	ww*	ww*	*w*	*oo	m**	w**
8	w**	ww*	ww*	ww*	ww*	ww*	*w*	*oo	m**

γ, it may be better to directly consider the non-negligible effect of network latency directly. Delays are especially noticeable when the system's throughput is increased by allowing larger blocks to form or by increasing block creation rates (see [17]). While this makes the encoding of the game rather complicated, *a priori* we can make the following observations:

1. The attacker has only a partial knowledge of the world state. Furthermore, blocks which it publishes may arrive at the honest network too late, which potentially reduces the benefit of block withholding.
2. Natural forks occur within the honest network, and consequently its chain grows in a rate lower than one block per round; this potentially makes attacks more successful.
3. Natural forks involving the attacker imply that the game arrives at non-trivial states, even under honest mining. The attacker may thus mine honestly until some particular deviation becomes feasible.
4. In the presence of delays, the attacker's share under honest mining might be greater than α, which raises the bar for dishonest strategies to prevail.[10]

The overall effect of the above cannot be determined without knowing the topology of the network and the attacker's location in it (as well as its knowledge about the topology). Still, some insight is possible. Following the third observation above, we notice that a dishonest policy π is one in which for some $h > a : \pi(a, h) \neq adopt$ and/or for some $a > h : \pi(a, h) \neq override$ (whereas under no delays honesty in $(1, 0)$ and $(0, 1)$ suffices). We claim that, consequently, the profit threshold equals 0. In other words, *every* attacker benefits from some form of dishonest mining.

Claim 7 *When the network suffers some delays, the attacker has a strict better-response strategy to honest mining, for any $\alpha > 0$.*

---

[10] See [13], for one result quantifying this effect.

Below we provide a proof sketch, which contains the jist of the claim while avoiding the involved formalization of the process under delays. We do mention that the rewards in M_ρ are now given by the expected outcome of future events, specifically the resolution of future conflicts. For instance, following an *override* action in (a, h), the attacker is awarded $(1 - \rho) \cdot (h + 1)$ times the probability that its block will be accepted by all nodes, eventually.[11]

Proof (sketch). Fix $k \in \mathbb{N}$, and let π_k be the policy in which the attacker mines honestly, until some state $(k-1, k)$ is reached (and observable to it). Upon which, instead of adopting, the attacker tries to catch up from behind, until it either succeeds (and then it overrides) or it learns of another block of the honest network (and then it adopts). Formally, $\pi_k(a, h) := \left\{ \begin{array}{ll} override & a > h \\ adopt & h > a \wedge h \neq k \end{array} \right\}$.
Since π_k is stationary, we can analyze its long-term earnings in M_ρ following the result from Lemma 8.

Denote $\rho_h := REV(\text{honest mining})$ and $\rho_k := REV(\pi_k)$. Upon reaching $(k-1, k)$, the attacker's immediate reward under honest mining is $(-\rho_h \cdot k)$. On the other hand, if it follows π_k, its expected immediate rewards are at least

$$q \cdot (1 - \rho_k) \cdot (k + 1) - (1 - q) \cdot \rho_k \cdot (k + 1), \tag{8}$$

where q is a lower bound on the probability that it will succeed to bypass the honest network's chain *and* override it in time. The positive term in (8) corresponds to the case where the scheme ends successfully (with an *override*), and the negative one to the complementary scenario. To avoid dependencies on k, q can be taken to equal

$$\int_0^\infty \int_0^\infty (\alpha \cdot \lambda)^2 \cdot e^{-\alpha \lambda \cdot (t+s)} \cdot e^{-(1-\alpha) \cdot \lambda \cdot (t+s+d_{a,h}+d_{h,a})} ds\, dt, \tag{9}$$

where $d_{h,a}$ is the communiation delay on the link from the honest cluster to the attacker, and $d_{a,h}$ the delay on the reversed link (for simplicity, we assume that in both directions there are single links connecting these parties to one another). Indeed, the integrand above represents the probability that the next two blocks of the attacker will take a time of $t + s$ to be generated $((\alpha \cdot \lambda)^2 \cdot e^{-\alpha \lambda \cdot (t+s)})$, and that the honest network hasn't been able to create a block since the beginning of the propagation of its k'th block, and until the attacker's $(k+1)$-block propagated throughout the network $(e^{-(1-\alpha) \cdot \lambda \cdot (t+s+d_{a,h}+d_{h,a})})$.

Assume by way of negation that $\rho_h \geq \rho_k$. If k is large enough, the following relation holds:

$$q \cdot (1 - \rho_k) \cdot (k + 1) - (1 - q) \cdot \rho_k \cdot (k + 1) - (-\rho_h \cdot k) \geq \tag{10}$$
$$q \cdot (1 - \rho_k) \cdot (k + 1) - (1 - q) \cdot \rho_k \cdot (k + 1) + \rho_k \cdot k =$$
$$(k + 1) \cdot q - \rho_k > 0. \tag{11}$$

This implies that the expected rewards of π_k, resulting form state $(k-1, k)$ being reached, exceed those of honest mining upon reaching this state. Since this is

---

[11] It can be shown that this is actually decided in finite time, in expectation [17].

the only state in which these strategies differ, the inequality above implies that π_k strictly dominates honest mining, thus $\rho_k = REV(\pi_k) > REV(\text{honest mining}) = \rho_h$. We conclude that any attacker can benefit from deviating in some states from honest mining, hence that the profit threshold vanishes.

$\square$

The intuition behind this result is clear: The attacker suffers a significant loss if it adopts in $(k-1, k)$, when k is large, and it thus prefers to continue the fork that formed naturally, and attempt to catch up.

This illustrates the importance of the policies found by Algorithm 1. As we've seen (Section 5), those dominate SM1 in that they delay adoption, i.e., they allow the continuation of the attack even when the honest network's chain is longer than the attacker's. While the additional benefit was rather mild, this added feature becomes more important in networks with delays, where splits in the chain occur naturally with some probability, even when honest mining is practiced by all.

To gain further understanding of selfish mining under delays it would be important to quantify the optimal gains from such deviations. We leave this as an open question for future research. Still, it is clear that Bitcoin will be more vulnerable to selfish mining if delays become more prominent, e.g., in the case of larger blocks (block size increases are currently being discussed within the Bitcoin developers community).

7   Effect on Double Spending Attacks

In this section we discuss the qualitative effect selfish mining has on the security of payments. The regular operation of bitcoin transactions is as follows: A payment maker signs a transaction and pushes it to the Bitcoin network, then nodes add it to the blocks they are attempting to create. Once a node succeeds it publishes the block with its content. Although the payee can now see this update to the public chain of blocks, it still waits for it to be further extended before releasing the good or service paid for. This deferment of acceptance guarantees that a conflicting secret chain of blocks (if exists) will not be able to bypass and override the public one observed by the payee, thereby discard the transaction. Building a secret chain in an attempt to reverse payments is called a *double spending* attack.

Success-probability. Satoshi Nakamoto, in his original white paper, provides an analysis regarding double spending in probabilistic terms: *Given that the block containing the transaction is followed by n subsequent blocks, what is the probability that an attacker with computational power α will be able to override this chain, now or in the future?* Nakamoto showed that the success-probability of double spending attacks decays exponentially with n. Alternative and perhaps more accurate analyses exist, see [15],[17].

Cost. While a single double spending attack succeeds with negligible probability (as long as the payee waits long enough), regrettably, an attacker which *continuously* executes double spending attempts will eventually succeed (*a.s.*). We

should therefore be more interested in the cost of an attack than in its success-probability. Indeed, every failed double spending attack costs the attacker the potential award it could have gotten had it avoided the fork and published its blocks right away.

Observe, however, that a smart strategy for an attacker would be to continuously employ selfish mining attacks, and upon success combine them with a double spending attack. Technically, this can be done by regularly engaging in public transactions, while always hiding a conflicting one in the attacker's secret blocks.[12] There is always some probability that by the time a successful selfish mining attack has ended, the payment receiver has already accepted the payment, which additionally results in a successful double spending.

To summarize, the existence of a miner for which selfish mining is at least as profitable as honest mining fundamentally undermines the security of payments, as this attacker bears no cost for continuously attempting to double spending, and it eventually must succeed. Similarly, an attacker that cannot profit from selfish mining alone, might be profitable in the long run if it combines it with double spending, which potentially has grave implications on the profit threshold.

8   Related Work

The Bitcoin protocol was introduced in a white paper published in 2008 by Satoshi Nakamoto [14]. In the paper, Nakamoto shows that the blockchain is secure as long as a majority of the nodes in the Bitcoin network follow the protocol. Kroll et al. [11] show that, indeed, always extending the latest block in the blockchain forms a (weak, non-unique) Nash equilibrium, albeit under a simpler model that does not account for block withholding.

On the other hand, it has been suggested by various people in the Bitcoin forum that strong nodes might be incentivized to violate the protocol by withholding their blocks [1]. Eyal and Sirer proved this by formalizing a block withholding strategy SM1 and analyzing its performance [9]. Their strategy thus violates the protocol's instruction to immediately publish one's blocks, but still sticks to the longest-chain rule (save a selective tie breaking). SM1 1 still abandons its chain if the honest nodes create a longer chain. One result of our paper is that even adhering to the longest-chain rule is not a best response. We also prove what the optimal policies are, and compute the threshold under which honest mining is a (strict, unique) Nash equilibrium. Additional work on selfish mining via block withholding appears in [3]. Transaction propagation in Bitcoin has also been analyzed from the perspective of incentives. Results in [2] show that nodes have an incentive not to propagate transactions, and suggests a mechanism to correct this. Additional analysis from a game theoretic perspective has also been conducted with regards to interactions pools, either from a cooperative game theory perspective [12], or when considering attacks between pools [8].

---

[12] In the worst case, the attacker is frequently engaged in "real" transactions anyways, hence suffers no loss from them being occasionally confirmed, when attacks fail.

A recent paper by Göbel et al. has evaluated SM1 in the presence of delays [10]. They show that SM1 is not profitable under a model of delays that greatly differs from our own (in particular, they assume that block transmission occurs as a memoryless process). While SM1 may indeed be unprofitable when delay is modeled, we show that other profitable selfish mining attacks exist. Additional analysis of block creation in the presence of delays and its effects on throughput and double spending appears in [17,13,7].

Further discussion on Bitcoin's stability can be found in a recent survey by Bonneau et al. [5].

References

1. `https://bitcointalk.org/index.php?topic=2227`, 2008. [Online; accessed 07-July-2015].
2. Moshe Babaioff, Shahar Dobzinski, Sigal Oren, and Aviv Zohar. On bitcoin and red balloons. In *Proceedings of the 13th ACM conference on electronic commerce*, pages 56–73. ACM, 2012.
3. Lear Bahack. Theoretical bitcoin attacks with less than half of the computational power (draft). *arXiv preprint arXiv:1312.7013*, 2013.
4. K-J Bierth. An expected average reward criterion. *Stochastic processes and their applications*, 26:123–140, 1987.
5. Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A Kroll, and Edward W Felten. Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. 2015.
6. Iadine Chadès, Guillaume Chapron, Marie-Josée Cros, Frédérick Garcia, and Régis Sabbadin. Mdptoolbox: a multi-platform toolbox to solve stochastic dynamic programming problems. *Ecography*, 37(9):916–920, 2014.
7. Christian Decker and Roger Wattenhofer. Information propagation in the bitcoin network. In *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on*, pages 1–10. IEEE, 2013.
8. Ittay Eyal. The miner's dilemma. *arXiv preprint arXiv:1411.7099*, 2014.
9. Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *Financial Cryptography and Data Security*, pages 436–454. Springer, 2014.
10. Johannes Göbel, Paul Keeler, Anthony E Krzesinski, and Peter G Taylor. Bitcoin blockchain dynamics: the selfish-mine strategy in the presence of propagation delay. *arXiv preprint arXiv:1505.05343*, 2015.
11. Joshua A Kroll, Ian C Davey, and Edward W Felten. The economics of bitcoin mining, or bitcoin in the presence of adversaries. In *Proceedings of WEIS*, volume 2013, 2013.
12. Yoad Lewenberg, Yoram Bachrach, Yonatan Sompolinsky, Aviv Zohar, and Jeffrey S Rosenschein. Bitcoin mining pools: A cooperative game theoretic analysis. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 919–927. International Foundation for Autonomous Agents and Multiagent Systems, 2015.
13. Yoad Lewenberg, Yonatan Sompolinsky, and Aviv Zohar. Inclusive block chain protocols. *Financial Cryptography and Data Security*, 2015.
14. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Consulted*, 1(2012):28, 2008.

15. Meni Rosenfeld. Analysis of hashrate-based double spending. *arXiv preprint arXiv:1402.2009*, 2014.
16. Richard Serfozo. *Basics of applied stochastic processes*. Springer Science & Business Media, 2009.
17. Yonatan Sompolinsky and Aviv Zohar. Secure high-rate transaction processing in bitcoin. *Financial Cryptography and Data Security*, 2015.
18. Frederick Stern. Conditional expectation of the duration in the classical ruin problem. *Mathematics magazine*, pages 200–203, 1975.

A Generality of the Model

As mentioned in Section 2, the most general setup would be for an attack-strategy to consider also building its blocks in different places in the block-tree (say, extending a previously abandoned chain, or adopting a subchain of the public chain) and/or to publish more than $h + 1$ blocks upon overriding the public honest chain. It is clear, intuitively, why such actions are suboptimal. Below we make this formal.

Let π be an optimal strategy, when the above actions are available to the attacker as well.

*Part I:*Assume there exists a state (a, h) where the attacker publishes $h + j$ blocks with $j > 1$; we denote this by $\pi(a, h) =$"*override by j*". We now construct a policy π', which follows π everywhere except that $\pi'(a, h) = wait$. By Corollary 10, it suffices to show that $v_\rho^{\pi'} \geq v_\rho^{\pi} = 0$, where ρ is the relative revenue induced by π; this will imply that no reduction in REV occurs when switching from "*override by j*" to *wait*.

For every state X, denote by $v_\rho^{\pi}(X)$ the expected value of $(1-\rho) \cdot \mathbb{E}\left[R^{1,1}(\pi)\right] - \rho \cdot \mathbb{E}\left[R^{2,1}(\pi)\right]$ conditioned on arriving at state X (recall that τ_1 is the terminating state of the first run). We need to show that $v_\rho^{\pi'}(a, h) \geq v_\rho^{\pi}(a, h)$, as this is the only states where these policies differ. Observe that $v_\rho^{\pi}(a, h) = (1-\rho) \cdot j + v_\rho^{\pi}(a - j, 0)$. This is because either $j < a$, and this action cannot lead to a termination (hence the addition of $v_\rho^{\pi}(a-j, 0)$), or $j = a$, and then $v_\rho^{\pi}(a-j, 0) = v_\rho^{\pi}(0, 0) = 0$, which fits the fact that a termination occurred. We thus need to show that $v_\rho^{\pi'}(a, h) \geq (1 - \rho) \cdot j + v_\rho^{\pi}(a - j, 0)$.

Indeed, consider the case where π' performs "*override by j*" if $X = (a, h+1)$ or "*override by (j+1)*" if $X = (a+1, h)$. Note that the action in the first case is feasible, since $a \geq h+j > h+1$, and similarly $a+1 \geq h+j+1 > h$, for the second case. In the former case we obtain $v_\rho^{\pi'}(a, h+1) = (1-\rho) \cdot j + v_\rho^{\pi}(a-j, 0)$, and in the latter, $v_\rho^{\pi'}(a, h+1) = (1-\rho) \cdot (j+1) + v_\rho^{\pi}(a+1-(j+1), 0)$. Therefore, we have presented an action-scheme which guarantees π' the value of π. As π (hence π') optimize the value $v_\rho^{\pi}(X)$, for any X, we have that the value of π (hence of π') in the states $(a+1, h)$ and $(a, h+1)$ is at least as high as $(1-\rho) \cdot j + v_\rho^{\pi}(a-j, 0)$, which completes this part of the proof.

Part II: We claimed, additionally, that the attacker will never adopt branches in the block-tree other than its current secret one and the honest faction's current longest one. We now aim to justify this assertion, albeit with some informalities; a formal proof is not possible under our model, because it implicitly

assumes that actions as *override* and *adopt* grant immediate reward, whereas if the attacker adopts older abandoned chains it can hypothetically reverse such decisions. Nonetheless it is very clear why this would be suboptimal:

For any (a, h), let $A_1, ..., A_a$ denote the attacker's chain, and $H_1, ..., H_h$ the honest network's chain, and let H_0 be the block that A_1 and H_1 extend (it is now public, but may have belonged to the attacker).[13] Let now (a, h) be the first state at which the attacker decides to deviate and extend a block B other than A_a or H_h. If B was not created after H_0 (and $B \neq H_0$), then it was available to the attacker at the time it began extending H_0. By the choice of (a, h), extending H_0 was then at least as profitable as extending B, and this dominance is invariant under future events (e.g., by the public chain that formed above H_0). Thus the attacker can just as well repeat its initial choice of H_0 over B.

A similar argument holds for the case where B was created after H_0 (or $B = H_0$). Denote by l the length of the attacker's chain upon the creation of B. Extending A_l was then at least as profitable as extending B, by the choice of (a, h), and this again is not altered by future events. All the same, the attacker can just as well repeat its choice and choose A_l over B. in conclusion, we can restrict our attention to strategies restricted to our three-action model (four, with *wait*), without loss of generality. This also enables a Markovian model, fortunately, as described in Section 2.

B Proof of Proposition 1

Proposition 1:
For any π, $REV(\pi, \alpha, \gamma) \leq \frac{\alpha}{1-\alpha}$. Moreover, this bound is tight, and achieved when $\gamma = 1$.

Proof. We can map every block of the honest network which was overridden, to a block of the attacker; this is because *override* requires the attacker to publish a chain longer than that of the honest network's.

Let k_T be the number of blocks that the attacker has built up to time T. The honest network thus built $l_T := T - k_T$ by this time. The argument above shows that $l_T - \sum_{t=1}^{T} r_t^2 \leq k_T$. Also, $\Pr(l_T > k_T) \to 1$, when $T \to \infty$. Therefore, the relative revenue satisfies:

$$REV(\pi) = \lim_{T \to \infty} \frac{\sum_{t=1}^{T} r_t^1}{\sum_{t=1}^{T} r_t^1 + \sum_{t=1}^{T} r_t^2} \leq \lim_{T \to \infty} \frac{\sum_{t=1}^{T} r_t^1}{\sum_{t=1}^{T} r_t^1 + l_T - k_T} = \tag{12}$$

$$\lim_{T \to \infty} \frac{1}{1 + (l_T - k_T) / \left(\sum_{t=1}^{T} r_t^1 \right)} \leq \lim_{T \to \infty} \frac{1}{1 + (l_T - k_T) / k_T} = \lim_{T \to \infty} \frac{k_T}{l_T}. \tag{13}$$

The SLLN applies naturally to k_T and l_T, implying that the above equals $\frac{\alpha \cdot T}{(1-\alpha) \cdot T} = \frac{\alpha}{1-\alpha}$ (*a.s.*).

---

[13] In case the honest network is forked, pick one of them arbitrarily; blocks are anonymous, and they are only accepted or rejected according to the lengths of their chains, which are in this case equal.

To see that the bound is achieved in $\gamma = 1$, observe that the policy SM1 satisfies the property that every block of the attacker overrides one block of the honest network, and that none of the attacker's blocks are overridden (as the policy never reaches a state where it needs to *adopt*, except when $a = 0$). This turns both inequalities in (12)-(13) into equalities. $\qquad\square$

C Correctness of Algorithm 1

In this section we prove that Algorithm 1 halts and that its output meets the conditions specified therein. We begin with applying here a Strong Law of Large Numbers, which will prove useful along our path. Under a fixed stationary policy π, we denote by τ_1 the renewal time of the game. Formally, τ_1 is the time, or number of visited states, until the game reaches a state s from which the transition probabilities are α to state $(1, 0)$ and $1 - \alpha$ to $(0, 1)$.

Lemma 8. *Let π be some fixed policy of $M_\rho^{T_0}$. Denote $R^{k,1}(\pi) = \sum_{t=1}^{\tau_1} r_t^k(\pi)$ (for $k = 1, 2$).*

$$\lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} r_t^k(\pi) = \mathbb{E}\left[\lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} r_t^k(\pi)\right] = \frac{\mathbb{E}\left[R^{k,1}(\pi)\right]}{\mathbb{E}[\tau_1]} \quad (a.s.), \qquad (14)$$

for $k = 1, 2$. Similarly,

$$\lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} w_\rho(r_t(\pi)) = \mathbb{E}\left[\lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} w_\rho(r_t(\pi))\right] \qquad (15)$$

$$= \frac{(1 - \rho) \cdot \mathbb{E}\left[R^{1,1}(\pi)\right] - \rho \cdot \mathbb{E}\left[R^{2,1}(\pi)\right]}{\mathbb{E}[\tau_1]} \quad (a.s.) \qquad (16)$$

Proof. Define by $\mathcal{C}^\pi$ the states reachable from state $s_0 := (1, 0, irrelevant)$, when π is employed. We will show that $\mathcal{C}^\pi$ is an irreducible positive recurrent Markov chain. For any state X in $\mathcal{C}^\pi$ it must be that the waiting time for the next visit of s_0 has finite expectation: Assume that after T' steps the honest network created $M(T')$ blocks and the attacker $m(T')$. If $M(T') > m(T')$ then as long as the player does not adopt $h - a = M(T') - m(T')$; this is regardless of other actions which the attacker possibly made in the past. As block creations are *i.i.d*, the process $Y(T') = M(T') - m(T')$ is equivalent to a random walk on $\mathbb{Z}$ with a positive drift, hence the expected time of the *last* time it returns to the origin is finite. After which the only action the attacker can make is *adopt* and *wait*. As our model does not allow for pathological strategies in which the attacker waits for periods of infinite expected length, the next adoption occurs in finite expected time. Finally, every adoption leads to X_0 with probability α, thus the next return to X_0 is of finite expectation. This state is thus positive recurrent. We conclude that $\mathcal{C}^\pi$ consists of a single communicating class (the finite expectation of the return implies the existence of a t for which there's a positive probability to return to s_0 within t steps), hence that π induces a single

irreducible Markov chain $\mathcal{C}^\pi$, which is also positive recurrent, as s_0 is. We can thus use The Strong Law of Large Numbers for Markov chains (see, e.g., [16] pg. 50, Corollary 79) to arrive at (14) and (16). The right-hand side equality in (14) follows from the SLLN applied to renewal reward processes. $\qquad\square$

The following are immediate corollaries of the strong law above:

Corollary 9. *For any admissible policy π of $M_\rho^{T_0}$,*

$$REV(\pi) = \frac{\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} r_t^1}{\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} (r_t^1 + r_t^2)} = \frac{\mathbb{E}\left[R^{1,1}\right]}{\mathbb{E}\left[R^{1,1}\right] + \mathbb{E}\left[R^{2,1}\right]} \quad (a.s.) \qquad (17)$$

Corollary 10. *Let π and π' be two policies.*

1. *If $(1-\alpha)\cdot\mathbb{E}\left[R^{1,1}\right] - \alpha\cdot\mathbb{E}\left[R^{2,1}\right] \geq 0$, then π dominates honest-mining.*
2. *If $(1-REV(\pi))\cdot\mathbb{E}\left[R^{1,1}(\pi')\right] - REV(\pi)\cdot\mathbb{E}\left[R^{2,1}(\pi')\right] > 0$, then π' dominates π.*

Both assertions become strict together with the inequalities.

The following lemma states that an optimal policy in $M_\rho^{T_0}$, whose value is small enough, is approximately optimal in M, if only truncated policies are considered:

Lemma 11. *Let $\rho \in [0,1]$, $\epsilon > 0$, and $T_0 \in \mathbb{N}$. If $\pi \in A^{T_0}$ is optimal in $M_\rho^{T_0}$ and $|v_\rho^\pi| < \epsilon/2$, then*

1. *$\left|\rho - REV(\pi)\right| < \epsilon$*
2. *$\left|\rho - \max_{\pi'\in A^{T_0}} \{REV(\pi')\}\right| < \epsilon$*

Proof. Observe that $\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T}\left(r_t^1(\pi') + r_t^2(\pi')\right)$ represents the average number of blocks added to the agreed pubic chain (aka main chain), per round, when π' is deployed. Under the honest strategy, this rate equals 1, as every round accounts for the addition of a new block (see Section 2). On the other hand, no positive recurrent strategy can more than halve the growth rate of the main chain: For every block that is overridden and excluded from the main chain there's a corresponding overriding block is included in it (see also the proof of Proposition 1).[14] Thus, $\mathbb{E}\left[\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T}\left(r_t^1(\pi') + r_t^2(\pi')\right)\right] \geq 1/2$.

---

[14] This assumption is without loss of generality, as at some point the player would need to adopt, and the waiting time for it is finite in expectation. See the proof of Lemma 8.

Part I: Relying on Lemma 8 we can manipulate the limits to obtain

$$\epsilon/2 > v_\rho^\pi = \mathbb{E}\left[\liminf_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} w_\rho(r_t^1(\pi), r_t^2(\pi))\right] = \tag{18}$$

$$\mathbb{E}\left[\liminf_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T}(1-\rho)\cdot r_t^1(\pi) - \rho\cdot r_t^2(\pi)\right] =$$

$$\mathbb{E}\left[\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} r_t^1(\pi)\right] - \rho\cdot\mathbb{E}\left[\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T}\left(r_t^1(\pi) + r_t^2(\pi)\right)\right].$$

Using Corollary 9 we obtain

$$REV(\pi) = \frac{\mathbb{E}\left[\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} r_t^1(\pi)\right]}{\mathbb{E}\left[\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T}\left(r_t^1(\pi) + r_t^2(\pi)\right)\right]} <$$

$$\rho + \frac{\epsilon/2}{\mathbb{E}\left[\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T}\left(r_t^1(\pi) + r_t^2(\pi)\right)\right]} \le \rho + \epsilon.$$

Similarly, $v_\rho^\pi > -\epsilon/2$ implies

$$REV(\pi) > \rho - \frac{\epsilon/2}{\mathbb{E}\left[\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T}\left(r_t^1(\pi) + r_t^2(\pi)\right)\right]} \ge \rho - \epsilon,$$

which concludes the first part.

Part II: We use here the same technique as previously. Assume by negation that for some policy $\pi' \in A^{T_0}$, $REV(\pi') \ge \rho + \epsilon$. Then, similar to the previous article, we have

$$\frac{\mathbb{E}\left[\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} r_t^1(\pi')\right]}{\mathbb{E}\left[\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T}\left(r_t^1(\pi') + r_t^2(\pi')\right)\right]} = REV(\pi') \ge \rho + \epsilon \implies \tag{19}$$

$$v_\rho^{\pi'} = \mathbb{E}\left[\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} r_t^1(\pi')\right] - \rho\cdot\mathbb{E}\left[\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T}\left(r_t^1(\pi') + r_t^2(\pi')\right)\right] \ge$$

$$\left(\mathbb{E}\left[\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T}\left(r_t^1(\pi') + r_t^2(\pi')\right)\right]\right)\cdot\epsilon \ge 1/2\cdot\epsilon > v_\rho^\pi, \tag{20}$$

which contradicts the optimality of v_ρ^π. This proves that $\rho > \max_{\pi' \in A^{T_0}} \{REV(\pi)\}$ $-\epsilon$. On the other hand, assume in negation that $REV(\pi) \le \rho - \epsilon$. We then have,

$$\frac{\mathbb{E}\left[\lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} r_t^1(\pi)\right]}{\mathbb{E}\left[\lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \left(r_t^1(\pi) + r_t^2(\pi)\right)\right]} \le \rho - \epsilon \implies$$

$$v_\rho^\pi = \mathbb{E}\left[\lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} r_t^1(\pi)\right] - \rho \cdot \mathbb{E}\left[\lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \left(r_t^1(\pi) + r_t^2(\pi)\right)\right] \le$$

$$\left(\mathbb{E}\left[\lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \left(r_t^1(\pi) + r_t^2(\pi)\right)\right]\right) \cdot -\epsilon \le 1/2 \cdot (-\epsilon) < v_\rho^\pi,$$

and we arrive again at a contradiction. Therefore, $REV(\pi) > \rho - \epsilon$, hence $\max_{\pi' \in A^{T_0}} \{REV(\pi')\} > \rho - \epsilon$. □

Corollary 12. *If π is $\epsilon/4$-optimal in $M_\rho^{T_0}$ and $|v_\rho^\pi| < \epsilon/4$, then the inequalities guaranteed by Lemma 11 hold.*

Proof. The first inequality holds for π, as in its proof we didn't use the assumption on π's optimality. The second inequality is a property of ρ (and not of the policy); it holds because $|v_\rho^\pi| < \epsilon/4$ together with π being $\epsilon/4$-optimal imply $|v_\rho^{\hat\pi}| < \epsilon/2$, for an optimal policy $\hat\pi$. □

Finally, we are ready to prove the correctness of Algorithm 1:
Proposition 4:
For any $T_0 \in \mathbb{N}$ and $\epsilon > 0$, Algorithm 1 halts, and its output (ρ, π) satisfies: $|\rho - REV(\pi)| < \epsilon$ and $\left|\rho - \max_{\pi' \in A^T} \{REV(\pi')\}\right| < \epsilon$.

Proof. Observe that v_ρ^{T*}, the optimal value of $M_\rho^{T_0}$, is monotonically decreasing in ρ: If $\rho_1 > \rho_2$ and π_1 is optimal in $M_{\rho_1}^{T_0}$, then $v_{\rho_2}^{T_0*} \ge v_{\rho_2}^{T_0,\pi_1} > v_{\rho_1}^{T_0,\pi_1} = v_{\rho_1}^{T_0*}$, where the strict inequality holds because w_ρ is strictly decreasing. Furthermore, $v_\rho^{T_0*}$ is continuous in ρ, as w_ρ is.

Now, the quantity $(high - low)$ is halved at every iteration of the loop (lines (6),(8)), hence the number of iterations must be finite. To understand what we can say about v when the algorithm halts and $high - low < \epsilon/8$, we make use of loop invariants: First, we claim that for every value assigned to low throughout the algorithm's run, the value returned by $mdp\_solver(M_{low}^{T_0}, \epsilon/8)$ is positive. Indeed, low begins with a value of 0. Honest mining gains the attacker a value of α, in $M_0^{T_0}$; $mdp\_solver(M_{low}^{T_0}, \epsilon/8)$ thus returns a positive value, assuming $\epsilon < 8 \cdot \alpha$. Any further alteration of low's value, in line 6, is conditioned to satisfy this assertion.

Similarly, the value returned by $mdp\_solver(M_1^{T_0}, \epsilon/8)$ must be negative, since the attacker's profits for its blocks vanishes, and its revenue for blocks it adopts is negative (and such events occur in finite time, in epxectation; see Lemma 8). In addition, any new assignment to $high$ is conditioned to be non-positive, by line 8.

From the monotonocity and continuity of $v_\rho^{T_0\,*}$ we deduce that the root of $v_\rho^{T_0\,*}$ lies between *low* and *high*. However, $high - low < \epsilon/8$ implies that $|v| < \epsilon/8$: Indeed, assume in negation that $v_\rho^\pi \geq \epsilon/8$. Then

$$\epsilon/8 \leq v_\rho^\pi = \mathbb{E}\left[\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} r_t^1(\pi)\right] - \rho \cdot \mathbb{E}\left[\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T}\left(r_t^1(\pi) + r_t^2(\pi)\right)\right] \leq$$

$$\mathbb{E}\left[\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} r_t^1(\pi)\right] - (\rho + \epsilon/8) \cdot \mathbb{E}\left[\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T}\left(r_t^1(\pi) + r_t^2(\pi)\right)\right] + \epsilon/8$$

(21)

$$\leq v_{\rho+\epsilon/4}^{T_0\,*} + \epsilon/8 < v_{high}^{T_0\,*} + \epsilon/8.$$

We used here the inequality $\mathbb{E}\left[\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T}\left(r_t^1(\pi) + r_t^2(\pi)\right)\right] \leq 1$ (see the proof of Lemma 11), and the strict monotonicity of $v_\rho^{T_0\,*}$. This contradicts $v_{high}^{T_0\,*} \leq 0$. A similar derivation rules out the case $v_\rho^\pi \leq -\epsilon/8$, which holds as a loop invariant. We conclude that $|v| < \epsilon/8$. □

Proposition 3. *For any* $T \in \mathbb{N}$, *if* $v_\rho^* \geq 0$ *then* $u_\rho^{T\,*} \geq v_\rho^* \geq v_\rho^{T\,*}$. *Moreover, these bounds are tight:* $\lim_{T\to\infty} u_\rho^{T\,*} - v_\rho^{T\,*} = 0$.

We precede the proof of the proposition with some (fun!) probability analysis. Denote by *Low-Triangle* the set of states $\{(a, h) : a \geq h\}$. Fix some policy π and a state (a_0, h_0). Denote by Y_t^π the random process defined by our game, where the initial state is (a_0, h_0). Let ψ be a stopping time defined by $\max\{t : Y_t^\pi \in Low\text{-}Triangle\}$. If $Y_\psi^\pi = (a_1, a_1)$ (observe that Y_ψ^π must lie in the main diagonal), we denote $last(a_0, h_0) := a_1$. $last(a_0, h_0)$ represents the number of blocks the attacker (or the honest network, for the matter) has, before leaving *Low-Triangle* for the rest of the epoch.

Lemma 13. *For any state* $(a, h_0) \in Low\text{-}Triangle$,

$$\mathbb{E}[last(a_0, h_0)] = \frac{\alpha \cdot (1 - \alpha)}{(1 - 2 \cdot \alpha)^2} + \frac{1}{2} \cdot \left(\frac{a_0 - h_0}{1 - 2 \cdot \alpha} + a_0 + h_0\right)$$

(22)

Proof. Note first that $last(a_0, h_0) = \frac{1}{2} \cdot (\psi - (a_0 - h_0)) + a_0$, because if the attacker created k blocks after reaching (a_0, h_0), the honest network needs to create precisely $k + a_0 - h_0$ blocks in order to leave *Low-Triangle*. We are thus left with the task of calculating $\mathbb{E}[\psi]$. Consider a random walk on $\mathbb{Z}$, starting at $a_0 - h_0$, with probability α of moving one step towards positive infinity and $(1 - \alpha)$ of moving towards negative infinity. Let ψ' be the time until the last visit of the origin. Observe that ψ' has the same distribution as ψ (!), we thus identify them with each other, henceforth.

We further break ψ into stopping times: Let N be the number of visits to the origin (we have $N > 0$ *almost surely*, since the drift is towards negative infinity). Let ψ_1 be the first time up to the first visit of the origin, and for $1 < k \leq N$, let

ψ_k be the time that elapsed between ψ_{k-1} and the next visit to the origin. Any two travels that begin and end at the origin are i.i.d, and, moreover, the number of such travels is independent of their lengths. Therefore, by Wald's equation, $\mathbb{E}\left[\psi - \psi_1\right] = \mathbb{E}\left[N\right] \cdot \mathbb{E}\left[\psi_2\right]$.

We can interpret N as counting the number of failures before one success, where a success represents a visit of the origin which never returns to it (this is equivalent, *almost surely*, to never returning to the nonnegative side of $\mathbb{Z}$). The probability of a success is $\left(1 - \frac{\alpha}{1-\alpha}\right)$, implying that $\mathbb{E}\left[N\right] = \frac{\frac{\alpha}{1-\alpha}}{1 - \frac{\alpha}{1-\alpha}} = \frac{\alpha}{1-2\cdot\alpha}$.

Whenever the walk starts at $+1$ the expected return time to the origin is $\frac{1}{1-2\cdot\alpha}$. The same expression holds for the expected return time when starting at -1, conditioned on a return occurring (see [18]). Counting the first step to ± 1 as well, the expected next return to the origin, conditioned on its occurrence, is $\left(1 + \frac{1}{1-2\cdot\alpha}\right)$.[15] We conclude that $\mathbb{E}\left[\psi - \psi_1\right] = \frac{\alpha}{1-2\cdot\alpha} \cdot \left(1 + \frac{1}{1-2\cdot\alpha}\right)$.

Another result in [18] implies that $\mathbb{E}\left[\psi_1\right] = \frac{a_0 - h_0}{1-2\cdot\alpha}$. We obtain:

$$\mathbb{E}\left[\psi\right] = \frac{\alpha}{1 - 2\cdot\alpha} \cdot \left(1 + \frac{1}{1 - 2\cdot\alpha}\right) + \frac{a_0 - h_0}{1 - 2\cdot\alpha} \implies \tag{23}$$

$$\mathbb{E}\left[last(a_0, h_0)\right] = \frac{1}{2} \cdot \left(\mathbb{E}\left[\psi\right] - (a_0 - h_0)\right) + a_0 = \frac{1}{2} \cdot \left(\mathbb{E}\left[\psi\right] + a_0 + h_0\right) = \tag{24}$$

$$\frac{1}{2} \cdot \left(\frac{\alpha}{1 - 2\cdot\alpha} \cdot \left(1 + \frac{1}{1 - 2\cdot\alpha}\right) + \frac{a_0 - h_0}{1 - 2\cdot\alpha} + a_0 + h_0\right) = \tag{25}$$

$$\frac{\alpha \cdot (1 - \alpha)}{(1 - 2\cdot\alpha)^2} + \frac{1}{2} \cdot \left(\frac{a_0 - h_0}{1 - 2\cdot\alpha} + a_0 + h_0\right). \tag{26}$$

$\square$

Proof (of Proposition 3). Part I: Let π be an optimal policy in M_ρ. Assume the game has reached state $(a, h) \in Low\text{-}Triangle$, and an oracle lets the attacker know that this is the last state in $Low\text{-}Triangle$ which the game will reach before a future *adopt*. Assume further that the oracle lets the attacker "cheat" and perform the action *match* with success probability 1 (granting him, effectively, $\gamma = 1$) and even if the previous state was not in $Low\text{-}Triangle$ (ignoring thus restrictions on the feasibility of *match*). Obviously, the attacker can only benefit from this oracle, by waiting for the last state in $Low\text{-}Triangle$, and then performing *match* (it has nothing to lose by taking only the null action up to that point).

Upon which, performing *match* on the main diagonal marks the end of the first epoch, since the respective chains of the attacker and the honest network collapse, hence the next state is distributed as X_0 is. As a result, we may bound the accumulated immediate rewards from state $(a, h) \in Low\text{-}Triangle$ onwards, up to τ_1, in M_ρ, by

$$(1 - \rho) \cdot \mathbb{E}\left[last(a, h)\right] = (1 - \rho) \cdot \frac{\alpha \cdot (1 - \alpha)}{(1 - 2\cdot\alpha)^2} + \frac{1}{2} \cdot \left(\frac{a - h}{1 - 2\cdot\alpha} + a + h\right). \tag{27}$$

---

[15] Note that, starting at $+1$, the expected return time is unaffected by conditioning on an eventual return, since this occurs w.p.1.

This is precisely the reward given in state $(a, h) \in$ *Low-Triangle* with $a = T$, in the over-paying MDP N_ρ^T.

We follow the same approach to bound the accumulated rewards from states $(a, h) \notin$ *Low-Triangle*. Assume that the oracle tells the attacker whether it will ever return to the main diagonal (without adopting first) or not. Clearly, if the oracle carries the negative message, the attacker is better off adopting right away, minimizing its negative reward.[16] This will imply a reward of $-\rho \cdot h$. On the other hand, if the oracle says the process will eventually return to the main diagonal, the attacker is better off waiting for that event. If we denote by (a_0, a_a) the next arrival at *Low-Triangle* (which is necessarily on the main diagonal), then $\mathbb{E}[a_0|\text{return occurs}] = \frac{h-a}{1-2\cdot\alpha}$ ([18]).

Upon which the attacker's future rewards up to τ_1 are bounded from above by $(1-\rho) \cdot \mathbb{E}[last(a_0, a_0)] = (1-\rho) \cdot \left(\frac{\alpha\cdot(1-\alpha)}{(1-2\cdot\alpha)^2} + a_0\right)$, by (22). Since this is linear in a_0, we conclude that the expected reward from state $(a, h) \notin$ *Low-Triangle*, conditioned on returning to the *Low-Triangle*, is upper bounded by $(1-\rho) \cdot \left(\frac{\alpha\cdot(1-\alpha)}{(1-2\cdot\alpha)^2} + \frac{h-a}{1-2\cdot\alpha}\right)$. The probability of this event is $(\alpha/(1-\alpha))^{h-a}$. All in all, the attacker's rewards from state $(a, h) \notin$ *Low-Triangle* onward are upper bounded by

$$\left(1 - \left(\frac{\alpha}{1-\alpha}\right)^{h-a}\right) \cdot (-\rho \cdot h) + \tag{28}$$

$$\left(\frac{\alpha}{1-\alpha}\right)^{h-a} \cdot (1-\rho) \cdot \left(\frac{\alpha\cdot(1-\alpha)}{(1-2\cdot\alpha)^2} + \frac{h-a}{1-2\cdot\alpha}\right).$$

This, again, is exactly the reward given in the over-paying N_ρ^T when state $(a, h) \notin$ *Low-Triangle* with $h = T$ is reached.

Part II: Recall the result of Lemma 8:

$$v_\rho^\pi = \frac{(1-\rho) \cdot \mathbb{E}\left[R^{1,1}(\pi)\right] - \rho \cdot \mathbb{E}\left[R^{2,1}(\pi)\right]}{\mathbb{E}[\tau_1]}. \tag{29}$$

When the optimal (in M_ρ) π is applied in N_ρ^T, with an *adopt* in the truncating states, the expected epoch time cannot be greater. Therefore, if $v_\rho^\pi = v_\rho^* \geq 0$, this transformation can only increase v_ρ^π. We conclude that if π is optimal policy in M_ρ, then the expected average value of (the truncated version of) π in N_ρ^T upper bounds $v_\rho^\pi = v_\rho^*$. An optimal policy of N_ρ^T can only do better, hence $u_\rho^{T*} \geq v_\rho^*$, which concludes the involved part of the proof.

Part III: That $v_\rho^* \geq v_\rho^{T*}$ is trivial, since any policy that is feasible in M_ρ^T is feasible in M_ρ, and the rewards are identical. Finally, we show that $u_\rho^{T*} \searrow v_\rho^T$. First, observe that the reward from visiting a state $(a, h) \notin$ *Low-Triangle*, given in (28), converges to $\rho \cdot h$. Thus, as T goes to infinity, the reward from these states in N_ρ^T converges to that of M_ρ^T. On the other hand, the probability to

---

[16] Recall it is forced to *adopt* at *some* stage, as we've seen before.

reach a state in *Low-Triangle* vanishes exponentially with T (e.g., by applying Chernoff's bound). The reward given in N_ρ^T in the truncating states of the form $(T = a \geq h)$ grows only linearly in T (see (27); a and h are linear in T). Therefore, the expected reward from these states (without conditioning on reaching them) vanishes. We thus obtain $\left(u_\rho^{T*} - v_\rho^T\right) \to 0$, as T goes to infinity. $\quad\square$

Corollary 14. *If $v_\rho^{T*} \geq 0$ then $\rho + 2 \cdot u_\rho^{T*} \geq \max_{\pi' \in A} \{REV(\pi')\}$.*

Proof. The proofs in Appendix C did not use the truncation of the process. We can therefore follow the same steps as in the proof of Lemma 11, *Part II*: Put $\epsilon = 2 \cdot u_\rho^{T_0*}$. Then $v_\rho^* \geq v_\rho^{T_0*} \geq 0$, hence $u_\rho^{T*} \geq v_\rho^*$, by Proposition 3. Similarly to the implication following (19)-(20), we can deduce that $\rho + 2 \cdot u_\rho^{T_0*} \geq \max_{\pi' \in A} \{REV(\pi')\}$. $\quad\square$

Proposition 5. *If u and ρ' are the outcome of the computation in Algorithm 1, lines 12-13, then $\rho' + 2 \cdot (u + \epsilon') > \max_{\pi' \in A} \{REV(\pi')\}$.*

Proof. If $low \leq \epsilon/4$ then ρ' is assigned the value 0. In this case, as shown above, $v_1^{T_0*} > 0$. Assume that $low > \epsilon/4$. In the proof of Proposition 4 it was shown that the value returned by $mdp\_solver(M_{low}^{T_0}, \epsilon/8)$ is positive. Therefore, $v_{low}^{T_0*} > -\epsilon/8$. Applying the proof of Lemma 11 we deduce that $v_{low-2\cdot\epsilon/8}^{T_0*} + \epsilon/8 > \epsilon/8$, hence $v_{\rho'}^{T_0*} > 0$. Corollary 14 thus applies to ρ', and we obtain $\rho' + 2 \cdot u_\rho^{T*} \geq \max_{\pi' \in A} \{REV(\pi')\}$. Observing that $u + \epsilon' > u_\rho^{T_0*}$ completes the proof. $\quad\square$

We complete the appendix with the proof of Corollary 6:

Corollary 6: *Fix γ and α. If u is the value returned by $mdp\_solver(\widehat{N_\alpha^T}, \epsilon)$, and $u \leq -\epsilon$, then honest mining is optimal for α. In other words, $\hat{\alpha}(\gamma) \geq \alpha$.*

Proof. If $u \leq -\epsilon$ then the value of $\widehat{N_\alpha^T}$ is smaller than 0. If we denote by $\widehat{M_\alpha}$ the same modification (of disabling honest mining) applied now to M_α, then the value of $\widehat{M_\alpha}$ cannot be poisitive (similarly to Proposition 3). However, honest mining guarntees a value of 0 in M_α, and we conclude that honest mining (weakly) dominates other strategies. $\quad\square$