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. The Bitcoin protocol requires nodes to quickly distribute newly created blocks. Strong nodes can, however, 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, who have demonstrated a specific deviation from the standard protocol (a strategy that we name SM1).
In this paper we investigate the *profit threshold* – the minimal fraction of resources required for a profitable attack. Our analysis provides a bound under which the system can be considered secure against such attacks. Our techniques can be adapted to protocol modifications to assess their susceptibility to selfish mining, by computing the optimal attack under different variants. We find that the profit threshold is strictly lower than the one induced by the SM1 scheme. The policies given by our algorithm dominate SM1 by better regulating attack-withdrawals. We further evaluate the impact of some previously suggested countermeasures, and show that they are less effective than previously conjectured.
We then 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 seminal 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 possess a large enough share of the computational resources, and are sufficiently well connected to the rest of

the network. 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. We begin with performing this analysis with respect to the standard Bitcoin protocol. Additionally, several protocol modifications have been put forth, some with the explicit goal of alleviating selfish mining, and we suggest our techniques as a tool to provably analyze their resilience to such attacks. To this end, we consider other possible deviations from the classic Bitcoin protocol, and establish bounds on their profitability. We later demonstrate how to apply the same techniques to some of its variants.

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 held by honest nodes than by misbehaving 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.[3] 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 more effective. Eventually, the attacker is able to collect all block rewards, to mount successful double spending attacks at will, and to prevent any transaction from being processed (this is known as the 50% attack, which requires 50% of the computational resources).

We summarize the contributions of this paper as follows:

1. We provide an efficient algorithm that computes an optimal selfish mining policy, for any parametrization of the model in [9] (i.e., one that maximizes the revenue of the attacker, given that all other nodes are following the standard Bitcoin protocol). We prove the correctness of our algorithm, and verify our results using simulations.

[3] 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.

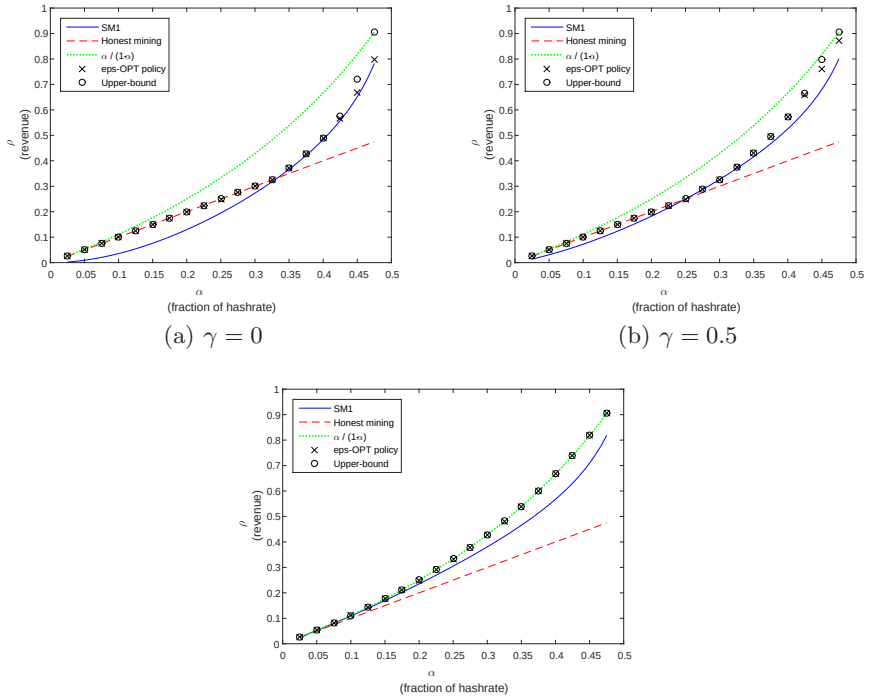
2. Using our algorithm we show that, indeed, there are selfish mining strategies that earn more money and are profitable for smaller miners compared to SM1. The gains are relatively small (see Fig. 1 below). This can be seen as a positive result, lower bounding the amount of resources needed for a profitable attacker.
3. We evaluate different protocol modifications that were suggested as countermeasures for selfish mining. For the solution suggested by Eyal and Sirer (in which miners randomly break ties between equal chains), we show that attackers with strictly less than 25% of the computational resources can still gain from selfish mining, unlike previously conjectured.
4. We show that in a model that accounts for the delay of block propagation in the network, attackers of *any* size can profit from selfish mining.
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 [15], 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 provide a preview of some of our final results, 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 hashrate α is below the threshold then honest mining is the most profitable strategy. For comparison, we depict the thresholds induced by SM1 as well.

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.[4]

[4] 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.



(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.

Blocks are created in the network according to a Poisson process with rate λ. A new block belongs to the attacker w.p. α or to the honest network w.p. $(1 - \alpha)$. The honest network follows the Bitcoin protocol, and always builds its newest block on top of the longest known chain. 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 (save the genesis block). Since 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).

Following the communication model of Eyal and Sirer, we assume that whenever the attacker learns that a block has been released by the network it is able

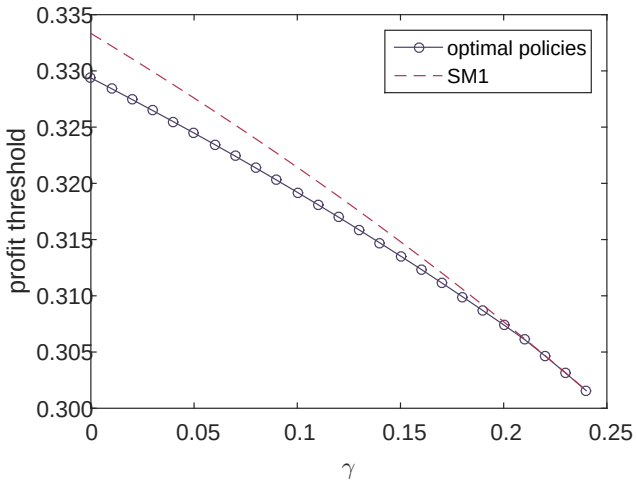


Fig. 2: The profit thresholds induced by optimal policies, and by SM1, as a function of γ. Thresholds at higher γ values match those of SM1 (but still, optimal strategies for these values earn more than SM1, once above the threshold).

to transmit an alternative block (which it created beforehand) that will arrive *first* at nodes that possess a fraction γ of the computational power of the honest network. 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.[5] 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 right 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

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

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 the full version). Blocks before that point are agreed upon by all participants. Accordingly, we must only keep track of blocks involved in the fork, and of the accumulated reward up to the fork. We denote by l_a and l_h the number of blocks built after the latest fork, by the attacker and by honest nodes, respectively.

Formally, if all other participants are following the standard protocol, the attacker faces a single-player decision problem of the form $M := \langle S, A, P, R \rangle$, where S is the state space, A the action space, P the stochastic transition matrix, 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.[6]

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

- **Adopt**. This action represents the attacker's acceptance of the honest network's chain. This action is always feasible, and following it the l_a blocks in the attacker's current chain are discarded.
- **Override**. This action represents the publication of $l_h + 1$ of the attacker's blocks. It is feasible whenever $l_a > l_h$.
- **Match**. Here the attacker responds to a freshly mined block of the honest network with the publication of its block of the same height. 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**. This is the null action, under which 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 $(l_a, l_h, fork)$. 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 possible values, dubbed *irrelevant*, *relevant* and *active*. State of the form $(l_a, l_h, relevant)$ means that the previous state was of the form $(l_a, l_h - 1, \cdot)$; this implies that if $l_a \geq l_h$, the *match* action is feasible. Conversely, $(l_a, l_h, irrelevant)$ denotes the case where the previous state was $(l_a - 1, l_h, \cdot)$, and the attacker has built the most recent block; in this situation, the attacker is unable to match the hth block of the honest network, which already propagated through the network. 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* (l_a, l_h) *or* $(l_a, l_h, \cdot)$*, in contexts where the* $fork$ *label plays no effective role.*

[6] 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.

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 it has the same distribution as X_0; its immediate reward is l_h in the coordinate corresponding to the honest network. An *override* reduces the attacker's secret chain by l_h+1 blocks, which it publishes, and which the honest network accepts. This bestows a reward of $l_h + 1$ 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 l_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 (that of the attacker and that of the honest nodes) is specified on the right-most column.

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

†feasible only when $l_a \geq l_h$
‡feasible only when $l_a \geq l_h$

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 $\pi(l_a, l_h, fork)$ for the action that π dictates be taken at state $(l_a, l_h, fork)$. The immediate reward from transiting from state x to state y, under the action dictated by π, is denoted $r(x, y, \pi) = (r^1(x, y, \pi(x)), r^2(x, y, \pi(x)))$. 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.

Honest Mining and SM1. We now define two policies of prime interest that will serve as a baseline for future comparisons. Honest mining is the unique policy which adheres to the protocol at every state. It is defined by

$$\text{honest mining}\,(l_a,l_h,\cdot) = \left\{\begin{array}{cc} \mathit{adopt} & l_h > l_a \\ \mathit{override} & l_a > l_h \end{array}\right\}, \tag{2}$$

and *wait* otherwise. Notice that under our model, $REV(\text{honest mining},\alpha,\gamma)=\alpha$ for all γ.[7] Eyal and Sirer's selfish mining strategy, SM1, can be defined as

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

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 $\gamma = 1$.*

[7] Indeed, in networks without delays, honest mining is equivalent to the policy $\{$ *adopt* if $(l_a,l_h)=(0,1)$; *override* if $(l_a,l_h)=(1,0)\}$, as these are the only reachable states.

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:

$$REV(\pi) = \lim_{T\to\infty} \frac{\sum_{t=1}^{T} r_t^1}{\sum_{t=1}^{T} r_t^1 + \sum_{t=1}^{T} r_t^2} \leq \lim_{T\to\infty} \frac{\sum_{t=1}^{T} r_t^1}{\sum_{t=1}^{T} r_t^1 + l_T - k_T} = \tag{5}$$

$$\lim_{T\to\infty} \frac{1}{1 + (l_T - k_T) \,/\, \left(\sum_{t=1}^{T} r_t^1\right)} \leq \lim_{T\to\infty} \frac{1}{1 + (l_T - k_T)\,/k_T} = \lim_{T\to\infty} \frac{k_T}{l_T}. \tag{6}$$

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

We claim that when $\gamma = 1$ the bound is achieved by the following policy:

$$\pi\left(l_a, l_h, \cdot\right) := \left\{ \begin{array}{cc} adopt & l_h = 1, l_a = 0 \\ match & l_h = l_a \\ wait & \text{otherwise} \end{array} \right\}. \tag{7}$$

Indeed, under π the attacker only publishes its current number of blocks l_a when the honest network has the same amount, hence every block of the attacker overrides one block of the honest network. In addition, none of the attacker's blocks are overridden, since the policy never reaches a state where it needs to *adopt* except when $l_a = 0$. This turns both inequalities in (5)-(6) into equalities. □

4 Computing 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. Suppose we are given some value ρ as a candidate for the optimal value of the objective function. We show that one can construct a related MDP, M_ρ, which has one important characteristic: If the average reward of its optimal policy is negative, then ρ is above the optimal value, and if it is positive, then ρ is below it. When the optimal average reward is exactly 0, ρ is optimal in the original problem, and the policy that obtains this value in M_ρ, obtains a value of ρ in the original problem.

Formally, 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. The value of a policy

π is defined by $v_\rho^\pi = \mathbb{E}\left[\liminf_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} w_\rho(r_t(\pi))\right]$, and the optimal value by $v_\rho^* = \max_{\pi\in A}\left\{v_\rho^\pi\right\}$.[8] Our solution method is based on the following proposition:

Proposition 2. *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 $\rho = REV(\pi_\rho^*)$. Additionally, v_ρ^* is monotonically decreasing in ρ.*

Following these observations we can utilize the family M_ρ to obtain an optimal policy: We simply perform a **search for $\rho \in [0,1]$ such that the optimal solution of M_ρ has a value $v_\rho^* = 0$**. Since v_ρ^* is monotonically decreasing, this can be done efficiently, using binary search.

In practice, this method needs some refinement before it is applied, due to several limitations: First, the search domain $[0,1]$ is continuous, and we must account for some precision error. Second, standard MDP solvers can only deal with finite-state spaces, and even then, only to a limited degree of accuracy. The need to deal with finite state-spaces forces us to work with truncated versions of M_ρ whose optimal values serve as lower bounds to the real optimal one. To circumvent this we have constructed another family of (finite) MDPs which serve as upper bounds. Both bounds are tight, that is, they converge to the optimal value as the truncation increases. A full description of this construction, an algorithm to obtain these tight bounds, and a proof of the algorithm's correctness appear in the full version available online. Proposition 2 follows from the analysis therein.

5 Results

5.1 Optimal Values

We ran our algorithm for γ in $\{0, 0.5, 1\}$, with various values of α, using an MDP solver for MATLAB (an implementation of the relative value iteration algorithm developed by Chadés et al. [5]).[9] The values of ρ returned by the algorithm are depicted in Figure 1 above; additionally, the values for $\gamma = 0$ appear in Table 2. The graphs 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.

5.2 Optimal Policies

We now present the ϵ-optimal policies returned by our algorithm, in two particular setups. Table 3 above describes the policy for an attacker with $\alpha = 1/3, \gamma = 0$

[8] The equivalence of this formalization of the value function and alternatives in which the order of expectation and limit is reversed is discussed in [3].

[9] The error parameter ϵ was set to be 10^{-5} and the truncation was set to $T = 75$. See full version.

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

α	$SM1$	ϵ-OPT	Upper-Bound
1/3	1/3	0.33705	0.33707
0.35	0.36650	0.37077	0.37080
0.375	0.42118	0.42600	0.42614
0.4	0.48372	0.48863	0.48976
0.425	0.55801	0.56788	0.57672
0.45	0.65177	0.66809	0.72203
0.475	0.78254	0.7987	0.90476

Table 3: Optimal actions for an attacker with $\alpha = 0.35, \gamma = 0$, in states (l_a, l_h) with $l_a, l_h \leq 7$. See legend in the text (Subsection 5.2).

l_a \ l_h	0	1	2	3	4	5	6	7
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
6	w	w	w	w	w	o	w	w
7	w	w	w	w	w	w	o	w

(here *match* is of no consequence, and no forks form). The row indices correspond to l_a and the columns to l_h. Table 4 corresponds to $\alpha = 0.45, \gamma = 0.5$. Each entry in it contains a string of three characters, corresponding to the possible status of $fork$: $irrelevant, relevant, active$. Actions are abbreviated to their initials: $\boldsymbol{a}dopt$, $\boldsymbol{o}verride$, $\boldsymbol{m}atch$, $\boldsymbol{w}ait$, while '$*$' represents an unreachable state.

Table 4: Optimal actions (abbreviated to their initials) for an attacker with $\alpha = 0.45, \gamma = 0.5$, for states $(l_a, l_h, \cdot)$ with $l_a, l_h \leq 7$. See legend in Subsection 5.2.[10]

l_a \ l_h	0	1	2	3	4	5	6	7
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**
6	w**	*mw	*mw	*mw	*mw	omw	wm*	w**
7	w**	*mw	*mw	*mw	*mw	*mw	ooo	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 l_h and l_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 [12], 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

Using the bounds provided by our algorithm, we are able to introduce lower bounds on the profit thresholds, i.e., on the minimal fraction of computational resources needed for a profitable attack. The exact methodology is described in the full version. 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 modifications to the Bitcoin protocol have been suggested. It is important to provably verify the resilience of such protocols to selfish mining, especially if their explicit goal is to counter this attack. This can be done by adapting our algorithm to the MDPs induced by these modifications. We demonstrate this below for three protocols.

GHOST. First, we make a short note regarding the GHOST protocol, an alternative to the longest-chain rule [18]. We point out that our entire analysis in this paper applies to the GHOST protocol as well, since GHOST coincides with the longest-chain rule when the network suffers no delays.

Freshness Preferred. The tie breaking rule in Bitcoin instructs a node to adopt the chain whose tip was received by it first, in case of ties. Freshness Preferred is an alternative suggested in [11]. It uses unforgeable timestamps, and dictates that if less than w seconds passed between the receiving of the two tips, the node adopts the chain which contains the most recent time-stamp. Freshness Preferred disincetivizes block withholding, and in particular counters the SM1 scheme. Unfortunately, when also considering deviations from the longest-chain rule, it can be seen that an attacker of any size can benefit from this protocol by waiting in state $(l_a, l_h) = (0, 1)$, rather than adopting, for a certain period. Such a consideration is also the basis for our following analysis of Eyal and Sirer's suggested tie breaking rule.

Uniform Tie Breaking. In case of ties, Eyal and Sirer suggest to instruct nodes to choose between the chains uniformly at random. 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

[10] For instance, the string "wm$*$" in entry $(l_a, l_h) = (3, 3)$ reads: "in case a fork is *irrelevant* (i.e., 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".

guarantees a positive lower bound on the threshold for profitability of SM1. On the other hand, it has the apparent downside of enhancing the power of poorly communicating attackers—it allows an attacker to *match* with a success-probability 1/2 even if its "real" γ is smaller than 1/2.

Unfortunately, there is an additional drawback. An attacker is able to *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. Indeed, by applying our algorithm to the setup induced by uniform tie breaking, we found that the profit threshold $\hat{\alpha}(0.5)$ 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 optimal policy is described in the online full version of this paper.

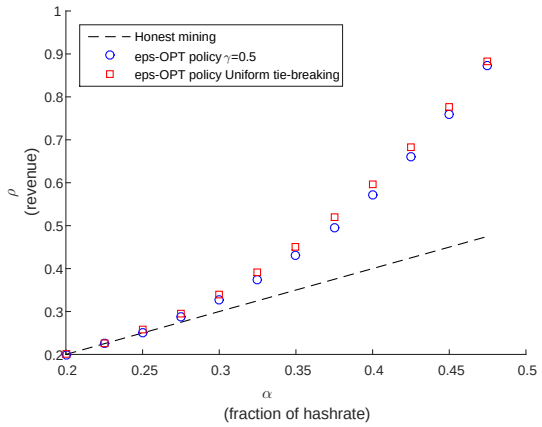


Fig. 3: The attacker's optimal revenue under uniform tie breaking, compared to that under the original protocol (with $\gamma = 1/2$) and to honest mining.

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), with the attacker following 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 during the propagation of blocks released earlier. 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 γ, it is better to consider the 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.

We now adjust our model to incorporate communication delays. For simplicity, and to demonstrate how the profit threshold breaks down even when a delay on a single link is introduced, we assume that only messages from the attacker to the honest network suffer delays. We denote this delay by $d_{a,h}$. Messages between honest nodes and from them to the attacker are assumed to be delivered instantaneously, as in the original model. Apart from l_a and l_h, the process now encodes also information about the timings of recent messages that are currently propagating on the link from the attacker's node to the honest network's node. Our following result states that under delays the profit threshold is 0. A node of *any* size can benefit from occasional deviation from the longest-chain rule:

Proposition 3. *Under communication delays (as modeled above), the attacker has a strict better-response strategy to honest mining, for any $\alpha > 0$.*

Proof. Part I: Fix $k \in \mathbb{N}$ and let $T > k + 1$. Below we describe a policy which instructs the attacker whether to adopt the honest network's chain (*adopt*) or not (*wait*). Our policy involves no block withholding at all; it thus deviates from the longest-chain rule, but adheres to immediate-publication. Consequently, the honest network adopts the attacker's chain if and only if the latter is longer by the time it arrived at the honest node. Our policy uses only (l_a, l_h) to decide between *wait* and *adopt* (although the state embeds additional information, e.g., timing of recent messages). When state $(l_a, l_h) = (k-1, k)$ is reached, our policy deviates from the honest protocol: In $(k-1, k)$ it waits, instead of adopting, until the $(k+1)$ block of the attacker or of the honest network is created; it then terminates the attack, and continues mining on the honest network's updated chain. Upon termination, the honest network's chain contains the $(k+1)$ blocks of the attacker only if they have propagated to the honest node in time. This possibility will be incorporated into the immediate reward function below.

Policies as the one described are technically not considered stationary, and a common circumvent is applied in such cases. We add a third bit to our representation of states, which denotes whether $(k-1, k)$ has been reached during this epoch. Thus, for any state with (l_a, l_h) we write $(l_a, l_h, 0)$, corresponding to the original states. Then, we add four new states that are only reachable from $(k-1, k, 0)$ under the *wait* action: $(k, k, 1), (k+1, k, 1), (k, k+1, 1), (k-1, k+1, 1)$. The original copies of these four states, with the third bit set to 0, are still reachable via $(k, k-1, 0)$. The states with the bit set to 1 are unreachable under honest mining, hence honest mining can still be seen as living inside this modified state-space.

When the process arrives at either of the states $(\cdot, k+1, 1)$ or $(k+1, k, 1)$ it renews (i.e., it transits to $(1,0,0)$ or $(0,1,0)$). Importantly, the immediate reward in M_ρ^T gained by arriving at $(\cdot, k+1, 1)$ is $(-\rho \cdot (k+1))$, and by arriving at $(k+1, k, 1)$ is $q \cdot (1-\rho) \cdot (k+1) - (1-q) \cdot \rho \cdot (k+1)$, where $q = e^{-(1-\alpha)\cdot 2 \cdot d_{a,h} \cdot \lambda}$. This will be justified shortly. Our policy acts as follows:

$$\pi_k(l_a, l_h, m) := \left\{ \begin{array}{cc} \mathit{wait} & k-1 \leq l_a \leq k, l_h = k, m = 1 \\ \mathit{adopt} & l_h = k+1, m = 1 \\ \text{honest mining}(l_a, l_h) & m = 0 \end{array} \right\}.$$

Since $(\cdot, k+1, 1)$ and $(k+1, k, 1)$ are terminating states, there are two possibilities: (a) The attacker managed to create the next two blocks in the network (corresponding to $(k+1, k, 1)$) *and* to propagate them in time to the honest network. Thence the honest network adopts the attacker's longer chain, and it gains an immediate reward of $(1-\rho) \cdot (k+1)$. In reality this happens w.p. $\geq q$, since q is the probability that the honest network created no conflicting blocks during $2 \cdot d_{a,h}$ seconds.[11] (b) State $(\cdot, k+1, 1)$ was reached, or $(k+1, k, 1)$ was reached but the attacker's blocks didn't make it in time to the honest node before the latter created its next one. The immediate reward described above assumes the worst case for the attacker, i.e., that it suffers a loss of $(-\rho \cdot (k+1))$ (in (b)).

Part II: Let $\rho = \rho_h := REV$ (honest mining) represent the revenue of the attacker under honest mining. In the full version it is shown that it is sufficient to prove that the expected sum of all immediate rewards accumulated in $M_{\rho_h}^T$, in a single epoch, is greater with π_k than with honest mining. Now, the two policies differ only after reaching state $(k-1, k, 0)$, and this state is reached under both policies with positive probability (as will be shown later). It thus suffices to prove this assertion w.r.t the rewards that result from reaching this state. The attacker's expected immediate reward, under π_k, upon reaching $(k-1, k, 0)$, is $\alpha^2 \cdot q \cdot (1-\rho_h) \cdot (k+1) - (1-\alpha^2 \cdot q) \cdot \rho_h \cdot (k+1)$; this stems from combining the rewards in scenarios (a) and (b). In contrast, under honest mining it gains $(-\rho_h \cdot k)$, since it adopts immediately. In conclusion, it suffices to show that $q \cdot \alpha^2 (1-\rho_h) \cdot (k+1) - (1 - q \cdot \alpha^2) \cdot \rho_h \cdot (k+1) - (-\rho_h \cdot k) > 0$. It is easy to see that this is satisfied by any $k > \frac{\rho_h}{q \cdot \alpha^2} - 1$. In particular, there exists a policy under which the expected immediate rewards of an attacker exceed those it would have gained mining honestly. As this holds for any $\alpha > 0$, we conclude that an attacker of any size can benefit from deviating from honest mining.

Part III: To complete the proof we show that after each renewal of the process, state $(k-1, k, 0)$ is reached with positive probability, under both policies. Indeed, there is a probability of α^{k-1} to arrive at $(k-1, 0, 0)$, with the attacker creating the first $(k-1)$ blocks. There is then a probability of $(1-\alpha)^k \cdot \left(1 - e^{-(1-\alpha)\cdot\lambda\cdot d_{a,h}}\right)^{k-1}$ to transit to $(k-1, k, 0)$, with the honest network creating the next k blocks before learning of any of the attacker's blocks. □

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

[11] After $2 \cdot d_{a,h}$ we are guaranteed that the link from the attacker to the honest network is empty.

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 debated 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 one 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, e.g., [17, 18], reach similar conclusions.

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 spend, and

[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.

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 [15]. 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. [12] 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 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.

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 [7, 14, 18].

A recent work by Nayak et al. [16], which was independently conducted in parallel to our own, discusses specific selfish mining strategies which outperform SM1 (that are not necessarily optimal), and analyzes the combination of selfish mining and a network-level attack. Our own work provides provably optimal strategies, which allows us to compute the profit thresholds, and to evaluate the worst attack given various protocol modifications (which can only be done with optimal strategies).

Additional work on selfish mining via block withholding appears in [6]. Another approach to mitigate selfish mining appears in [11]. 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 [13], or when considering attacks between pools [8]. Further discussion on Bitcoin's stability can be found in a recent survey by Bonneau et al. [4].

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. K-J Bierth. An expected average reward criterion. *Stochastic processes and their applications*, 26:123–140, 1987.
4. 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.
5. 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.
6. Nicolas T Courtois and Lear Bahack. On subversive miner strategies and block withholding attack in bitcoin digital currency. *arXiv preprint arXiv:1402.1718*, 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. Ethan Heilman. One weird trick to stop selfish miners: Fresh bitcoins, a solution for the honest miner. *IACR Cryptology ePrint Archive*, 2014:7, 2014.
12. 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.
13. 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.
14. Yoad Lewenberg, Yonatan Sompolinsky, and Aviv Zohar. Inclusive block chain protocols. *Financial Cryptography and Data Security*, 2015.
15. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Consulted*, 1(2012):28, 2008.
16. Kartik Nayak, Srijan Kumar, Andrew Miller, and Elaine Shi. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack.
17. Meni Rosenfeld. Analysis of hashrate-based double spending. *arXiv preprint arXiv:1402.2009*, 2014.
18. Yonatan Sompolinsky and Aviv Zohar. Secure high-rate transaction processing in bitcoin. *Financial Cryptography and Data Security*, 2015.