Fully Automated Selfish Mining Analysis in Efficient Proof Systems Blockchains

Krishnendu Chatterjee
krishnendu.chatterjee@ist.ac.at
IST Austria
Austria

Amirali Ebrahimzadeh
ebrahimzadeh.amirali@gmail.com
Sharif University of Technology
Iran

Mehrdad Karrabi
mehrdad.karrabi@ist.ac.at
IST Austria
Austria

Krzysztof Pietrzak
krzysztof.pietrzak@ist.ac.at
IST Austria
Austria

Michelle Yeo
mxyeo@nus.edu.sg
National University of Singapore
Singapore

Đorđe Žikelić
dzikelic@smu.edu.sg
Singapore Management University
Singapore

ABSTRACT

We study selfish mining attacks in longest-chain blockchains like Bitcoin, but where the proof of work is replaced with efficient proof systems – like proofs of stake or proofs of space – and consider the problem of computing an optimal selfish mining attack which maximizes expected relative revenue of the adversary, thus minimizing the chain quality. To this end, we propose a novel selfish mining attack that aims to maximize this objective and formally model the attack as a Markov decision process (MDP). We then present a formal analysis procedure which computes an ϵ-tight lower bound on the optimal expected relative revenue in the MDP and a strategy that achieves this ϵ-tight lower bound, where $\epsilon > 0$ may be any specified precision. Our analysis is fully automated and provides formal guarantees on the correctness. We evaluate our selfish mining attack and observe that it achieves superior expected relative revenue compared to two considered baselines.

In concurrent work [Sarenche FC'24] does an automated analysis on selfish mining in *predictable* longest-chain blockchains based on efficient proof systems. Predictable means the randomness for the challenges is fixed for many blocks (as used e.g., in Ouroboros), while we consider *unpredictable* (Bitcoin-like) chains where the challenge is derived from the previous block.

CCS CONCEPTS

• **Security and privacy** → **Logic and verification**; **Cryptanalysis and other attacks**; **Distributed systems security**.

KEYWORDS

Blockchain, Formal Methods, Efficient Proof Systems, Selfish Mining, Markov Decision Process

ACM Reference Format:
Krishnendu Chatterjee, Amirali Ebrahimzadeh, Mehrdad Karrabi, Krzysztof Pietrzak, Michelle Yeo, and Đorđe Žikelić. 2024. Fully Automated Selfish Mining Analysis in Efficient Proof Systems Blockchains. In *ACM Symposium on Principles of Distributed Computing (PODC '24), June 17–21, 2024, Nantes, France.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3662158.3662769

This work is licensed under a Creative Commons Attribution International 4.0 License.
PODC '24, June 17–21, 2024, Nantes, France
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0668-4/24/06
https://doi.org/10.1145/3662158.3662769

1 INTRODUCTION

Bitcoin. Blockchain protocols were proposed as a solution to achieve consensus over some states (e.g., financial transactions) in a distributed and permissionless (i.e., everyone can participate in securing the chain) setting. Participants in blockchain protocols add data into blocks, which are then appended to the blockchain with some probability that depends on the underlying consensus protocol. The earliest and most commonly adopted consensus protocol is proof of work (PoW), which also forms the basis of the Bitcoin blockchain protocol [20].

Proof of work. The parties that maintain a PoW blockchain like Bitcoin are called miners. The general idea is that in order to add a block to the chain, the miners derive a computationally hard but easily verifiable puzzle from the tip of the chain. To add a block to the chain, this block must contain a solution to this puzzle. This mechanism ensures that attacking the chain, in particular rewriting past blocks in a double spending attack, is computationally very expensive. In Bitcoin, the puzzle is a hashcash style PoW [1], with parameters being a difficulty level D and a global hash function (e.g., SHA256). A newly generated block can only be added to the chain if the hash of the new block, the previous block, the miner's public key and some nonce give a value that is less than the current difficulty D. To mine a block in Bitcoin, miners will continuously generate different nonces and hash them until they find a nonce that passes the threshold. The lucky miner that first finds a block broadcasts it across the network, where other miners can easily verify the validity of the nonce. The Bitcoin protocol specifies that miners should always work towards extending the longest chain they are aware of, hence such chains are called "longest-chain blockchains". An alternative approach is chains based on Byzantine Fault Tolerance like Algorand [5] where randomly rotating committees approve blocks to be added.

Efficient proof systems. As an alternative to PoW, several other consensus protocols based on efficient proof systems have been proposed [7, 8, 21]. Generally, we say a "proof of X" is efficient if computing a proof for a random challenge is very cheap assuming one has the resource X. The most popular and best investigated efficient proofs are Proofs of Stake (PoStake) as e.g., used in Ouroboros [8] or post-merge Ethereum. Here the coins recorded in the blockchain are the resource. Another proposal of an actual physical resource are Proofs of Space (PoSpace) [9], where the resource is disk-space. The first proposal for a PoSpace based chain was Spacemint [21]. The first deployed chain was the Chia network [7], it uses PoSpace in combination with verifiable delay functions (VDFs) to address some security challenges, and thus is referred to as a Proofs of Space and Time (PoST) based chain. See Appendix B in our extended technical report [4] for more details of blockchains based on efficient proof systems that we consider in our work.

Selfish mining. There are various security properties we want from longest-chain blockchains, the most important ones being persistence and liveness [15]. Informally, persistence means that entries added to the chain will remain there forever, while liveness means that the chain remains available. A less obvious requirement is fairness, in the sense that a party that contributes a p fraction of the resource (hashing power in PoW, space in PoST, staked coins in PoStake) should contribute a p fraction of the blocks, and thus get a p fraction of the rewards.

It was first observed in [10] that Bitcoin is not fair in this sense due to *selfish mining* attacks. In selfish mining, attackers mine blocks but occasionally selectively withhold these (so the honest miners cannot mine on top of those) to later release them, overtaking the honest chain and thus orphaning honest blocks. In doing so, the attackers can reduce the *chain quality* of the blockchain [10], this measure quantifies the fraction of blocks contributed by an adversarial miner.

Although the analysis of selfish mining and its impact on chain quality is well-studied in Bitcoin and other PoW-based blockchains [10, 18, 24, 27], a thorough analysis of optimal selfish mining attacks in blockchains based on efficient proof systems is difficult due the fact that the computational cost of generating proofs is low in these systems. This gives rise to several issues that in the PoStake setting are generally referred to as the "nothing-at-stake" (NaS) problem. [1]

The security vs predictability dilemma. Assume one would naïvely replace PoW in Bitcoin with an efficient proof system like PoStake. As computing proofs is now cheap, an adversarial miner can try to extend blocks at different depths (not just on the tip of the longest chain), growing private trees at each depth. If they manage to create a tree of depth d that starts less than d blocks deep in the public chain, releasing the longest path in this tree will force the honest miners to switch to this path. Such "tree growing" attacks can be prevented by diverting from Bitcoin-like protocols, and instead of using the block at depth i to derive the challenge for the block at depth $i + 1$, one uses some fixed randomness for a certain number of consecutive blocks. Unfortunately, this creates a new security issue as an adversary can now predict when in the future they will be able to create blocks [2, 3]. We note that this approach is used in the PoStake-based Ouroboros [8] chain, where one only creates a fresh challenge every five days. An extreme in the other direction is the PoSpace-based Spacemint [21] chain or an early proposal of the PoST-based Chia chain [6] which, like Bitcoin, derive the challenge from the previous block (the deployed Chia design [7] uses a fresh challenge every 10 minutes, or 32 blocks).

[1]We will slightly abuse terminology in our work and continue to use the terms "mining" and "miners" from PoW-based chains also when discussing chains based on efficient proof systems even though when using PoStake this is sometimes referred to as "proposing" and "proposers" (but it is not coherent, some works like [12, 13] use mining) while in PoSpace it was suggested to use the term "farming" and "farmers".

Limitations of previous analyses. The persistence, i.e., security against *double spending* attacks, in predictable (Ouroboros-like) and unpredictable (Bitcoin-like) longest-chain blockchains is pretty well understood [2]. In particular, using the "tree growing" attack outlined above, an adversary can "amplify" its resources in an unpredictable protocol by a factor of $e \approx 2.72$. Thus, security requires that the adversary controls less than $\frac{1}{1+e} \approx 0.269$ fraction of the total resources. In a predictable protocol, it is sufficient for the adversary to have $< \frac{1}{2}$ of the resource. This is better, but being predictable opens new attack vectors like bribing attacks. Unfortunately, the security of blockchain protocols based on efficient proof systems in the face of *selfish mining* attacks is a lot less understood. In particular, previous analyses in this area suffer mainly from two limitations:

(1) (Model) Although there have also been some very recent works analysing *selfish mining* attacks in blockchains based on efficient proof systems [7, 11, 13, 18, 25, 26], these analyses so far have only focused on predictable protocols.
(2) (Methodology) Furthermore, they either consider different adversarial objectives [7, 11, 13, 26] or use deep reinforcement learning to obtain selfish mining strategies [18, 25]. However, deep reinforcement learning only empirically maximizes the objective and does not provide *formal guarantees* on the quality (lower or upper bound) of learned strategies.

Our approach. In this work, we present the first analysis of selfish mining in unpredictable longest-chain blockchains based on efficient proof systems. Recall that in such chains the challenge for each block is determined by the previous block. At each time step, the adversary has to decide whether to mine new blocks or to publish one of their private forks which is longer than the public chain. Our analysis is concerned with finding the optimal sequence of mining and fork reveal actions that the adversary should follow in order to maximize the expected relative revenue (i.e., the ratio of the expected adversarial blocks in the main chain when following a given strategy compared to the total number of blocks in the chain). Note that this is a challenging problem. Since at each time step the adversary can choose between mining new blocks and publishing *any* of its private chains, the strategy space of the adversary is exponential in the number of privately mined forks which makes the manual formal analysis intractable.

To overcome this challenge, we model our selfish mining attack as a *finite-state Markov Decision Process (MDP)* [23]. We then present a formal analysis procedure which, given a precision parameter $\epsilon > 0$, computes

- an *ϵ-tight lower bound on the optimal expected relative revenue* that a selfish mining strategy in the MDP can achieve, and
- a *selfish mining strategy* achieving this ϵ-tight lower bound.

At the core of our formal analysis procedure lies a reduction from the problem of computing an optimal selfish mining strategy under the expected relative revenue objective to computing an optimal strategy in the MDP under a *mean-payoff objective* for a suitably designed reward function. While we defer the details on MDPs and mean-payoff objectives to Section 2, we note that solving mean-payoff finite-state MDPs is a classic and well-studied problem within the formal methods community for which efficient (polynomial-time) algorithms exist [14, 23] and there are well-developed tools that implement them together with further optimizations [17, 19]. These algorithms are *fully automated* and provide *formal guarantees* on the correctness of their outputs, and the formal analysis of our selfish mining attack naturally inherits these desirable features.

Contributions. Our contributions can be summarized as follows:

(1) We study selfish mining in unpredictable efficient proof systems blockchain protocols where the adversary's goal is to maximize *expected relative revenue* and thus minimize chain quality.
(2) We propose a *novel selfish mining attack* that optimizes the expected relative revenue in unpredictable blockchain protocols based on efficient proof systems. We *formally model* the attack as an MDP.
(3) We present a *formal analysis procedure* for our selfish mining attack. Given a precision parameter $\epsilon > 0$, our formal analysis procedure computes an ϵ*-tight* lower bound on the optimal expected relative revenue in the MDP together with a *selfish mining* strategy that achieves it. The procedure is *fully automated* and provides *formal guarantees* on its correctness.
(4) Our formal analysis is *flexible* to changes in system model parameter values. For instance, it allows us to tweak system model parameters and study their impact on the optimal expected relative revenue that selfish mining can achieve in a *fully automated fashion* and without the need to develop novel analyses for different parameter values. This is in stark contrast to formal analyses whose correctness is proved manually, see Section 3.4 for a more detailed discussion.
(5) We implement the MDP model and the formal analysis procedure and *experimentally evaluate* the quality of the expected relative revenue achieved by the computed selfish mining strategy. We compare our selfish mining attack to two baselines: (1) honest mining strategy and (2) a direct extension of the PoW selfish mining strategy of [10] to the setting of blockchains based on efficient proof systems. Our experiments show that our selfish mining strategy achieves higher expected relative revenue compared to the two baselines.

A Remark on the Model. While the selfish mining attacks analyzed in this paper apply to PoStake, PoSpace and PoST based longest-chain blockchains, they capture PoST better than PoStake and PoSpace as we will shortly outline now. Due to the use of VDFs, PoST is "strongly unpredictable" in the sense that a miner cannot predict when they will be able to extend any block, while PoStake and PoSpace are "weakly unpredictable" in the sense that a miner can predict when they will find blocks on top of their own (but not other) blocks. Moreover, in PoST a malicious miner must run a VDF on top of every block they try to extend, while in PoStake or PoSpace this comes basically for free. The class of selfish mining attacks analyzed in this paper does not exploit "weak unpredictability" (a necessary assumption for PoST, but not PoStake or PoSpace) and, to be able to give automated bounds, we also assume a bound on the number of blocks one tries to extend, which is a realistic assumption for PoST (as each block requires a VDF) but less so for PoSpace or PoStake.

1.1 Related Work

Selfish mining in Bitcoin. One of the motivations behind the initial design of Bitcoin and other PoW blockchain systems is fairness. That is, a miner controlling $p \in [0, 1]$ proportion of resources should generate blocks at the rate of p. This led to the initial analysis claiming Bitcoin is fair when the total resource p owned by adversarial miners is bounded from above by $\frac{1}{2}$ [15]. Nevertheless, [10] outlined an attack called "selfish mining" in Bitcoin which shows that even for $p < \frac{1}{2}$ it is possible to generate blocks at a rate strictly greater than p, implying that Bitcoin is inherently "unfair". The attack secretly forks the main chain and mines blocks in a private, unannounced chain. These blocks are only revealed when the private chain is longer than the main chain, thus forcing the honest miners to switch from the main chain to the private chain. This causes the honest miners to waste their computational resources on the now shorter public chain. It is shown in [10] that due to selfish mining, the largest amount of adversarial resources that can be tolerated while ensuring the security of Bitcoin is $\frac{1}{3}$ in the optimistic setting where honest miners propagate their blocks first, and $\frac{1}{4}$ if both honest and adversarial miners propagate their blocks first with equal probability.

NaS selfish mining. NaS selfish mining can be studied under several different adversarial objectives, see Appendix A in our extended technical report [4] for a thorough discussion and comparison of these objectives. The objective we consider in our work is the expected relative revenue of the adversary, which is also considered in the analysis of [25] for PoStake. There have been earlier analyses that consider several other objectives, however. The works of [26] and [7] studied the probability of an adversarial coalition overtaking the honest chain and showed respectively that under this objective, the largest fraction of adversarial resources that can be tolerated in PoStake based blockchains is $\frac{1}{1+e} \approx 0.269$, and between $\frac{1}{1+e} < p < \frac{1}{2}$ for Chia, a PoST-based blockchain. The work of [11] studied selfish mining strategies in PoStake blockchains under the objective of finding strategies that give the adversarial coalition a larger payoff compared to following the honest strategy. They showed that the expected advantage of the adversary when growing a private tree from the genesis block is e times larger than the expected revenue of following the honest strategy, implying that the maximum fraction of adversarial resource that can be tolerated to be secure under this strategy and objective is $\frac{1}{e} \approx 0.37$.

Optimal selfish mining strategies. There have been some works that go beyond proposing and analysing specific selfish mining strategies to actually claiming the *optimality* of these strategies. The work of [24] modelled the Bitcoin protocol as an MDP and proposed a method using binary search to solve it approximately in order to find an optimal selfish mining strategy. [27] improved the search method in [24], resulting in a method that finds an optimal strategy but with an order of magnitude less computation. [18] and [25] use deep reinforcement learning to automate discovery of attack strategies in Bitcoin and Nakamoto-PoStake, respectively, but without guarantees on optimality. Finally, [12] suggested modelling mining strategies in PoStake as an MDP and using an MDP solver to find optimal selfish mining strategies. However, due to the infinite size of their MDP, finding even an approximately optimal solution is undecidable.

2 PRELIMINARIES

2.1 System Model

We define a formal model for blockchain systems which we will consider throughout this work. The model describes how block mining proceeds and is parametrized by several parameters. Different parameter value instantiations then give rise to formal models for blockchains based on PoW, PoStake, or PoST protocols. We assume the miners in the blockchain protocol are either adversarial or honest. Honest miners all follow a prescribed protocol, whereas the adversarial miners work (i.e. pool resources) together in a coalition.

System model. We assume that block mining proceeds in discrete time steps, as in many discrete-time models of PoW [16, 22] and PoStake [8] protocols. For a miner that owns a fraction $p \in [0, 1]$ of the total resources in the blockchain and that can mine up to $k > 0$ blocks at any given time step, we define (p, k)-mining as follows: the probability of mining a block at the given time step is proportional to $p \cdot k$, and the maximum number of blocks which are available for mining is k. One can think of k as some further constraints on the number of blocks a miner can mine on at any given point in time, e.g. due to the number of VDFs they own in PoST. Hence, $(p, 1)$-mining corresponds to mining in PoW based blockchains, (p, k)-mining for $k < \infty$ to mining in PoST based blockchains with k VDFs, and (p, ∞)-mining to mining in PoStake based blockchains.

Adversarial and broadcast model. Let $p \in [0, 1]$ denote the fraction of resources in the chain owned by the adversarial coalition. We assume the adversarial coalition participates in (p, k)-mining, and the honest miners participate in $(1 - p, 1)$-mining, where the only block the honest miners mine on at any time step is the most recent block on the longest public chain. When there is a tie, i.e., two longest chains are gossiped through the network, the honest miners will mine on the chain which is gossiped to them first. In such situations, $\gamma \in [0, 1]$ denotes the *switching probability*, i.e., the probability of honest miners switching to the adversary's chain.

2.2 Selfish Mining Objective

Chain quality. Chain quality is a measure of the number of honest blocks in any consecutive segment of the blockchain. A blockchain protocol is said to satisfy (μ, ℓ)-chain quality if for any segment of the chain of length ℓ, the fraction of honest blocks in that segment is at least μ [15].

Selfish mining objective. Let σ denote an adversarial mining strategy. The selfish mining objective we analyse in our work is the *expected relative revenue* of the adversary. Formally, let $\textsc{revenue}_{\mathcal{A}}$ and $\textsc{revenue}_{\mathcal{H}}$ denote the number of adversarial and honest blocks in the main chain respectively. The expected relative revenue (ERRev) of the adversary under strategy σ is defined as

$$\mathsf{ERRev}(\sigma) = \mathbb{E}^{\sigma}\Big[\frac{\textsc{revenue}_{\mathcal{A}}}{\textsc{revenue}_{\mathcal{A}} + \textsc{revenue}_{\mathcal{H}}}\Big]$$

Note that the chain quality of the blockchain under an adversarial mining strategy σ is simply $1 - \mathsf{ERRev}(\sigma)$, hence our selfish mining objective captures the expected change in the chain quality.

2.3 Markov Decision Processes

As mentioned in Section 1, we will reduce the problem of computing optimal selfish mining strategies for the expected relative revenue objective to solving MDPs with mean-payoff objectives. In what follows, we recall the necessary notions on MDPs and formally define the mean-payoff objectives. For a finite set X, we use $\mathcal{D}(X)$ to denote the set of all probability distributions over X.

Markov decision process. A *Markov Decision Process (MDP)* [23] is a tuple $\mathcal{M} = (S, A, P, s_0)$ where

- S is a finite set of *states*, with $s_0 \in S$ being the *initial state*,
- A is a finite set of *actions*, overloaded to specify for each state $s \in S$ the set of *available actions* $A(s) \subseteq A$, and
- $P : S \times A \to \mathcal{D}(S)$ is a *transition function*, prescribing to each $(s, a) \in S \times A$ a probability distribution over successor states.

A *strategy* in $\mathcal{M}$ is a recipe to choose an action given a history of states and actions, i.e. it is a function $\sigma : (S \times A)^* \times S \to \mathcal{D}(A)$. In general, strategy can use randomization and memory. A *positional strategy* uses neither randomization nor memory, i.e. it is a function $\sigma : S \to A$. We denote by Σ and Σ^p the set of all strategies and all positional strategies in $\mathcal{M}$. Given a strategy σ, it induces a probability measure $\mathbb{P}^{\sigma}[\cdot]$ in $\mathcal{M}$ with an associated expectation operator $\mathbb{E}^{\sigma}[\cdot]$.

Mean-payoff objective. A *reward function* in $\mathcal{M}$ is a function $r : S \times A \times S \to \mathbb{R}$ which to each state-action-state triple prescribes a real-valued reward. Given an MDP $\mathcal{M}$, a reward function r and a strategy σ, we define the *mean-payoff* of σ with respect to r via

$$\mathsf{MP}(\sigma) = \mathbb{E}^{\sigma}\Big[\liminf_{N\to\infty} \frac{\sum_{n=1}^{N} r_n}{N}\Big],$$

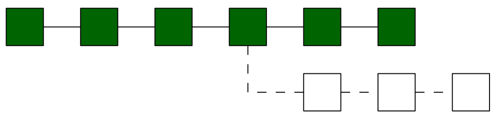
where $r_n = r_n(s_n, a_n, s_{n+1})$ is the reward incurred at step n.

The mean-payoff MDP problem is to compute the maximal mean-payoff that a strategy in the MDP can achieve. A classic result in MDP analysis is that there always exists a positional strategy σ^* that achieves maximal mean-payoff in the MDP, i.e. there is $\sigma^* \in \Sigma^p$ such that $\mathsf{MP}(\sigma^*) = \max_{\sigma\in\Sigma^p} \mathsf{MP}(\sigma) = \sup_{\sigma\in\Sigma} \mathsf{MP}(\sigma)$ [23]. Furthermore, the optimal positional strategy and the mean-payoff that it achieves can be efficiently computed in polynomial time [14, 23].

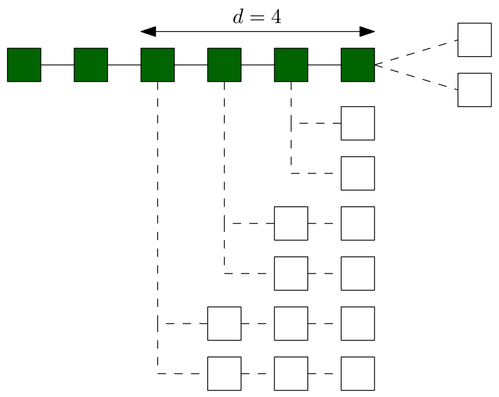
3 SELFISH MINING ATTACK

3.1 Overview

Motivation. In order to motivate our selfish mining attack, we first recall the classic selfish mining attack strategy in Bitcoin [10]. Recall, the goal of the selfish mining strategy is to mine a private chain that overtakes the public chain (see Figure 1a). Selfish miners secretly fork the main chain and mine privately, adding blocks to a private, unannounced chain. These blocks are only revealed when the length of the private chain catches up with that of the main chain, forcing the honest miners to switch from the main to the private chain and waste computational resources. While in PoW blockchains each party mines on one block, in blockchains based on efficient proof systems parties can mine on multiple blocks due to ease of generating proofs. Our selfish mining attack exploits this observation by creating multiple private forks concurrently.



(a) Classic selfish mining attack in Bitcoin.



(b) An illustration of our selfish mining attack with depth $d = 4$ and $f = 2$. Green boxes denote the main chain and white boxes denote potential blocks that can be mined under attack strategy.

Outline of the attack. Our selfish mining attack proceeds with the adversary creating several private forks at different blocks in the main chain, see Figure 1b for an illustration. Rather than forking on the most recent block alone, the adversary creates up to f forks on each of the last d blocks on the main chain. Here, f and d are parameters of the attack where f is the number of forks created at each public block and d represents the depth of the adversary's attack on the chain. One can view d as the persistence parameter of the blockchain, which represents the depth at which earlier blocks are practically guaranteed to remain in the main chain [15]. At each time step, the adversary can either perform a

(1) *mining action*, i.e. attempt to mine a new block, or
(2) *fork reveal action*, i.e. publicly announce one of the private forks whose length is greater than or equal to that of the main chain.

Deciding on the optimal order of mining and fork reveal actions that the adversary should perform at each time step towards maximizing its expected relative revenue is a highly challenging problem. This is because, at each time step, the party to mine the next block is chosen probabilistically (see our System Model in Section 2.2), and the process results in a system with an extremely large number of states. For any given precision parameter $\epsilon > 0$, our analysis will provide both an *ϵ-optimal strategy* among selfish mining strategies that the adversary can follow, together with the *exact value* of the expected relative revenue guaranteed by this strategy.

Formal model of the attack. The goal of our analysis is to find an optimal selfish mining strategy which maximizes the expected relative revenue of the adversary. Note that this is a sequential decision making problem, since the optimal strategy under the above selfish mining setting must at every decision step optimally choose whether to perform a mining or a fork release action. Hence, in order to analyze this problem, in Section 3.2 we formally model our problem as an MDP. The state space of the MDP consists of all possible configurations of the main chain and private forks with the initial state corresponding to the time step at which the selfish mining attack is initiated. The action space of the MDP consists of the mining action as well as one fork release action for each private fork whose length is greater than or equal to that of the main chain. Finally, the probabilistic transition function of the MDP captures the probabilistic process that generates and determines ownership (honest or adversarial) of new blocks to be added to the blockchain, as well as the process of determining the new main chain whenever the adversary publishes one or more private forks of equal length.

Formal analysis of the attack. Recall, the objective of our selfish mining attack is to maximize the expected relative revenue of the adversary. To do this, in Section 3.3 we define a class of reward functions in the MDP constructed in Section 3.2. We show that, for any $\epsilon > 0$, we can compute an ϵ-optimal selfish mining strategy in the MDP and the exact value of the expected relative revenue that it guarantees by solving the mean-payoff MDP with respect to a reward function belonging to the class of constructed reward functions. We solve mean-payoff MDPs by using off-the-shelf tools as mentioned in Section 2.3. Our analysis yields a *fully automated* method for computing ϵ-optimal strategies and values of the expected relative revenue that provides *formal guarantees* on the correctness of its results.

3.2 Formal Model

We now formally model our selfish mining attack as an MDP. Recall, an MDP $\mathcal{M} = (S, A, P, s_0)$ is an ordered tuple where S is a finite set of states, A is a finite set of actions overloaded to specify for each state $s \in S$ the set of available actions $A(s) \subseteq A$, $P : S \times A \to \mathcal{D}(S)$ is a probabilistic transition function, and $s_0 \in S$ is the initial state. Hence, in order to formally model our attack as an MDP, we need to define each of these four objects.

Model parameters. Our MDP model uses as a basis the System Model that we defined in Section 2.2. Thus, it is parameterized by the parameters p and γ, but also by three additional parameters specific the selfish mining attack itself. We use $\mathbb{N}$ to denote the set of all positive integers:

- *Relative resource of the adversary.* $p \in [0, 1]$ denotes the fraction of resources in the blockchain owned by the adversary.
- *Switching probability.* $\gamma \in [0, 1]$ denotes the probability of an honest miner switching to a newly revealed adversarial chain that is of the same length as the main chain.

- *Attack depth.* $d \in \mathbb{N}$ denotes the depth of the adversary's attack on the chain, i.e. the number of last blocks on the main chain on which the adversary mines private forks.
- *Forking number.* $f \in \mathbb{N}$ denotes the number of private forks that the adversary creates at each of the last d public blocks.
- *Maximal fork length.* $l \in \mathbb{N}$ denotes the maximal private fork length. Introducing this parameter ensures that our MDP model consists of finitely many states, which is necessary since existing mean-payoff MDP solvers are applicable to finite state MDPs [17, 19]. We discuss the implications of this in Section 3.4.

MDP definition. We define the MDP $\mathcal{M} = (S, A, P, s_0)$ as follows:

- *States.* The state space is defined via

$$S = \Big\{(C, O, \textsc{type}) \,\Big|\, C \in \{0, \dots, l\}^{d\times f}, O \in \{\textsc{honest}, \textsc{adversary}\}^{d-1}, \textsc{type} \in \{\textsc{mining}, \textsc{honest}, \textsc{adversary}\}\Big\},$$

i.e. each state is defined as a triple $(C, O, \textsc{type})$ where C defines the current blockchain configuration (i.e. topology) up to depth d, O specifies who owns each block in the main chain up to depth d (honest miners or the adversary), and TYPE specifies whether the parties are still mining or some party has mined a new block and is supposed to add it to the blockchain. In particular:
  - For each $1 \le i \le d$ and $1 \le j \le f$, we use $C[i, j]$ to denote the length of the j-th private fork mined by the adversary at the depth i block on the main chain. Each private fork has length at most l, so we impose that each $C[i, j] \in \{0, \dots, l\}$.
  - For each $1 \le i < d$, we use $O[i]$ to denote who owns the block at depth i in the main chain. In particular, $O[i] = \textsc{honest}$ if the block is owned by honest miners and $O[i] = \textsc{adversary}$ if the block is owned by the adversary.
  - Finally, TYPE specifies whether a new block to be added to the blockchain is still being mined in which case we set TYPE = MINING, or if some party has generated the proof and gets to add a new block in which case we set TYPE = HONEST and TYPE = ADVERSARY, respectively.
- *Initial State.* Initial state $s_0 = (C_0, O_0, \textsc{type}_0)$ corresponds to the time step at which the selfish mining attack is initiated. Hence, we set $C_0 = [0]^{d\times f}$ since the length of each private fork is initially 0, $O_0 = [\textsc{honest}]^d$ and $\textsc{type}_0 = \textsc{mining}$.
- *Actions.* The action space is defined via

$$A = \{\textsc{mine}\} \cup \Big\{\textsc{release}_{i,j,k} \,\Big|\, 1 \le i \le d, 1 \le j \le f, 1 \le k \le l\Big\}.$$

Intuitively, MINE prescribes that the adversary should not release any private forks and should simply continue mining new blocks. On the other hand, $\textsc{release}_{i,j,k}$ prescribes that the adversary should publish the first k blocks of the j-th private fork mined on the block at depth i in the main chain. The set of available actions at each MDP state $s = (C, O, \textsc{type})$ is defined as follows:
  - If TYPE = MINING, then $A(s) = \{\textsc{mine}\}$.
  - If $\textsc{type} \in \{\textsc{honest}, \textsc{adversary}\}$, then $A(s) = \{\textsc{mine}\} \cup \{\textsc{release}_{i,j,k} \mid k \le C[i, j]\}$, where the condition $k \le C[i, j]$ simply ensures that the length of the published part of the private fork cannot exceed the total length of the private fork.
- *Transition Function.* Finally, we define the transition function $P : S \times A \to \mathcal{D}(S)$. Let $s = (C, O, \textsc{type})$ and $a \in A(s)$ be an action available in s:
  - If TYPE = MINING, then we must have $a = \textsc{mine}$ as the only available action. The next state in the MDP is chosen probabilistically, based on who mines the next block and where. Recall, the honest miners own $1-p$ fraction of resources in the blockchain and are mining on the main chain only, whereas the adversary owns p fraction of resources but mines on at most $d \cdot f$ private forks. Note that the adversary will concurrently mine on top of each private fork starting at public blocks up to depth d in the main chain, as well as on top of each public block up to depth d at which not all f private works were initiated. Hence:
    * Denote by σ the total number of blocks that the adversary is mining on. For each $1 \le i \le d$ and $1 \le j \le f$, if the j-th private fork on the public block at depth i in the main chain is not empty, adversary mines a new block on it and the MDP moves to $s^{i,j}_{\text{ADV}} = (C', O, \textsc{adversary})$ with probability

    $$P(s, a)(s^{i,j}_{\text{ADV}}) = \frac{p}{1 - p + p \cdot \sigma}.$$

    Here, C' coincides with C on all entries except $C'[i, j] = \min\{C[i, j] + 1, l\}$, where the new block is found on top of the j-th private fork on the block at depth i. The minimum ensures that new block is not added if the length of the fork would exceed l.
    Moreover, if at least one private fork on the public block at depth i in the main chain is empty, adversary mines a block on it to start a new private fork and the MDP moves to $s^{i,j^*}_{\text{ADV}} = (C'', O, \textsc{adversary})$ with probability $\frac{p}{1-p+p\cdot\sigma}$, where j^* is the smallest index of a private fork that is currently empty at the depth i public block and C'' coincides with C on all entries except $C''[i, j^*] = 1$.
    * Honest miners add a new block to the main chain and the MDP moves to state $s_{\text{HONEST}} = ([0]^f \cdot C[1 : d-1], [\textsc{honest}] \cdot O[1 : d-2], \textsc{honest})$ with probability

    $$P(s, a)(s_{\text{HONEST}}) = \frac{1 - p}{1 - p + p \cdot \sigma}.$$

    where σ is as defined above. We use $[0]^f \cdot C[1 : d-1]$ to denote the $d \times f$ matrix whose first column consists of zeros (representing empty private forks on the newly added block) and the remaining columns are the first $d-1$ columns of C. Similarly, $[\textsc{honest}] \cdot O[1 : d-2]$ denotes the vector whose first component is HONEST corresponding to the new block, followed by the first $d-2$ components of O.
  - If $\textsc{type} \ne \textsc{mining}$ but $a = \textsc{mine}$, then the adversary continues mining new blocks and the MDP moves to $s_{\text{MINE}} = (C, O, \textsc{mining})$ with probability

  $$P(s, a)(s_{\text{MINE}}) = 1.$$

  - If TYPE = HONEST and $a = \textsc{release}_{i,j,k}$, then the adversary publishes the first k blocks of the j-th private fork mined on the block at depth i in the main chain. Hence, the next MDP state is determined by the chain which gets accepted as the main chain by honest miners. If the newly published fork is strictly longer than the main chain, the honest miners accept it as the new main chain with probability 1. Otherwise, if the main chain and the published fork have the same length, then a "race" between the chains happens and the published fork becomes the main chain with switching probability γ. Hence:

* If the published fork is longer than the main chain, the honest miners switch to the fork and the MDP moves to $s_{\text{ACCEPT}} = (C' \cdot [0]^{(min(d,k)-1)\times f} \cdot C''$, $[\text{ADVERSARY}]^{min(d-1,k)} \cdot O[k+1 : d-1]$, MINING) with probability
$$P(s,a)(s_{\text{ACCEPT}}) = 1.$$
Here, C' is a $(1\times f)$-dimensional vector with 0 entries except that $C'[1,1] = C[i,j] - k$. So, the adversary keeps the part of the published private fork consisting of the last $C[i,j] - k$ blocks that have not been revealed. On the other hand, C'' coincides with $C[k+1 : d]$ on all entries except that $C''[i,j] = 0$. Hence, the adversary keeps and continues mining on all usable private forks, with one new private fork of initial length 0 being created to replace the published fork. Note, $O[k+1 : d-1]$ and $C[k+1 : d]$ are empty if $k+1 > d-1$ and $k+1 > d$, respectively.
* Otherwise, if the published fork and the main chain are of the same length, a race happens and honest miners switch to the published fork with probability γ. and the new block will be mined by adversary with probability p. So, the MDP moves to $s_{\text{ACCEPT}} = (C' \cdot [0]^{(min(d,k)-1)\times f} \cdot C''$, $[\text{ADVERSARY}]^{min(d-1,k)} \cdot O[k+1 : d-1]$, MINING) with probability
$$P(s,a)(s_{\text{ACCEPT}}) = \gamma,$$
where C' and C'' are as above, and moves to $s_{\text{REJECT}} = (C, O, \text{MINING})$ with probability
$$P(s,a)(s_{\text{REJECT}}) = 1 - \gamma.$$

– Finally, if TYPE = ADVERSARY, $a = \text{RELEASE}_{i,j,k}$ and the published private fork is longer than the main chain, the MDP moves to
$s_{\text{ACCEPT}} = (C' \cdot [0]^{(min(d,k)-1)\times f} \cdot C''$, $[\text{ADVERSARY}]^{min(d-1,k)} \cdot O[k+1 : d-1]$, MINING) with probability
$$P(s,a)(s_{\text{ACCEPT}}) = 1.$$
where C' and C'' are defined as the previous case. Note that if the published fork and the main chain are of the same length, the race cannot happen as the last block was mined by the adversary and enough time has passed for all the miners to receive the public chain.

3.3 Formal Analysis

Goal of the analysis. We now show how to compute the optimal expected relative revenue together with an adversarial strategy achieving it in the MDP $\mathcal{M} = (S, A, P, s_0)$, up to an arbitrary precision parameter $\epsilon > 0$. Formally, for each strategy σ in $\mathcal{M}$, let

$$\mathsf{ERRev}(\sigma) = \mathbb{E}^{\sigma}\Big[\frac{\text{REVENUE}_{\mathcal{A}}}{\text{REVENUE}_{\mathcal{A}} + \text{REVENUE}_{\mathcal{H}}}\Big]$$

be the expected relative revenue under adversarial strategy σ, i.e. the relative ratio of the number of blocks accepted on the main chain belonging to the adversary and to honest miners. Moreover, let

$$\mathsf{ERRev}^* = \sup_{\text{strategy } \sigma \text{ in } \mathcal{M}} \mathsf{ERRev}(\sigma)$$

be the optimal expected relative revenue that an adversarial strategy can attain. Given a precision parameter $\epsilon > 0$, our goal is to compute

(1) a lower bound $\overline{\mathsf{ERRev}} \in [\mathsf{ERRev}^* - \epsilon, \mathsf{ERRev}^*]$, and
(2) a strategy $\overline{\sigma}$ in $\mathcal{M}$ such that $\mathsf{ERRev}(\overline{\sigma}) \in [\mathsf{ERRev}^* - \epsilon, \mathsf{ERRev}^*]$.

We do this by defining a class of reward functions in the MDP $\mathcal{M}$ and showing that, for any value of the precision parameter $\epsilon > 0$, we can compute the above by solving the mean-payoff MDP with respect to a reward function belonging to this class. Our analysis draws insight from that of [24], which considered selfish mining in PoW blockchains and also reduced reasoning about expected relative revenue to solving mean-payoff MDPs with respect to suitably defined reward functions. However, in contrast to [24], we consider selfish mining in efficient proof systems in which the adversary can mine on multiple blocks, meaning that our design of reward functions and the analysis require additional care.

Reward function definition. The key challenge in designing the reward function is that the main chain and the blocks on it may change whenever the adversary publishes a private fork. Hence, we design the reward function to incur positive (resp. negative) reward whenever a block owned by the adversary (resp. honest miners) is accepted at the depth strictly greater than d in the main chain. Since the adversary only mines and publishes private forks mined on blocks up to depth d in the main chain, this means that blocks beyond depth d are *guaranteed* to remain on the main chain.

Formally, for each $\beta \in [0,1]$, we define $r_\beta : S \times A \times S \to \mathbb{R}$ to be a reward function in $\mathcal{M}$ which to each state-action-state triple (s, a, s') assigns the reward:

- $1-\beta$, for each block belonging to *the adversary* accepted at depth greater than d as a result of performing the action;
- $-\beta$, for each block belonging to *honest miners* accepted at depth greater than d as a result of performing the action.

This definition can be formalized by following the same case by case analysis as in the definition of the transition function in Section 3.2. For the interest of space, we omit the formal definition. For each β and strategy σ in the MDP, let $\mathsf{MP}_\beta(\sigma)$ be the mean-payoff under σ with respect to the reward function r_β, and let $\mathsf{MP}^*_\beta = \sup_\sigma \mathsf{MP}_\beta(\sigma)$. Note that, since for each state-action-state triple the value of the reward $r_\beta(s,a,s')$ is monotonically decreasing in $\beta \in [0,1]$, we have that MP^*_β is also monotonically decreasing in $\beta \in [0,1]$.

Formal analysis. Our formal analysis is based on the following theorem. For clarity of exposition, we defer the proof of the theorem to Appendix C in our extended technical report [4] . For every $\epsilon > 0$, the theorem shows how to relate the optimal expected relative revenue in the MDP and ϵ-optimal strategies to the optimal mean-payoff and ϵ-optimal strategies under the reward function r_β for a suitably chosen value of β.

THEOREM 3.1. *We have* $\mathsf{MP}^*_{\beta^*} = 0$ *if and only if* $\beta^* = \mathsf{ERRev}^*$. *Moreover, if* $\epsilon > 0$ *and* $\beta \in [\mathsf{ERRev}^* - \epsilon, \mathsf{ERRev}^*]$, *then for any strategy* σ *such that* $\mathsf{MP}_\beta(\sigma) = \mathsf{MP}^*_\beta$ *we have* $\mathsf{ERRev}(\sigma) \in [\mathsf{ERRev}^* - \epsilon, \mathsf{ERRev}^*]$.

Algorithm 1 Formal analysis procedure

```
Input Precision parameter $\epsilon > 0$, MDP parameters $p, d, f, l, \gamma$
$\mathcal{M} \leftarrow$ MDP constructed as in Section 3.2
$\beta_{\text{low}}, \beta_{\text{up}} \leftarrow 0, 1$
while $\beta_{\text{up}} - \beta_{\text{low}} \geq \epsilon$ do
    $\beta \leftarrow (\beta_{\text{low}} + \beta_{\text{up}})/2$
    $\mathsf{MP}^*_\beta, \sigma_\beta \leftarrow$ solve mean-payoff MDP $\mathcal{M}$ with reward $r_\beta$
    if $\mathsf{MP}^*_\beta < 0$ then
        $\beta_{\text{up}} \leftarrow \beta$
    else
        $\beta_{\text{low}} \leftarrow \beta$
    end if
end while
$\overline{\mathsf{ERRev}} \leftarrow \beta_{\text{low}}$
$\overline{\sigma} \leftarrow$ solve mean-payoff MDP $\mathcal{M}$ with reward $r_{\beta_{\text{low}}}$
Return Expected relative revenue $\overline{\mathsf{ERRev}}$, strategy $\overline{\sigma}$
```

Following Theorem 3.1, we compute $\overline{\mathsf{ERRev}}$ and $\overline{\sigma}$ for a given precision $\epsilon > 0$ as follows. Algorithm 1 shows the pseudocode of our formal analysis. The algorithm performs binary search in $\beta \in [0, 1]$ in order to find a value of β for which $\beta \in [\mathsf{ERRev}^* - \epsilon, \mathsf{ERRev}^*]$. In each iteration of the binary search, the algorithm uses an off-the-shelf mean-payoff MDP solver to compute the optimal mean-payoff MP^*_β and a strategy σ_β attaining it. Upon binary search termination, we have $\mathsf{ERRev}^* \in [\beta_{\text{low}}, \beta_{\text{up}}]$, since MP^*_β is a monotonically decreasing function in β and since $\mathsf{MP}^*_{\beta^*} = 0$ if and only if $\beta^* = \mathsf{ERRev}^*$ by the first part of Theorem 3.1. Hence, as the binary search terminates when $\beta_{\text{up}} - \beta_{\text{low}} < \epsilon$, we conclude that $\overline{\mathsf{ERRev}} = \beta_{\text{low}} \in [\mathsf{ERRev}^* - \epsilon, \mathsf{ERRev}^*]$. Moreover, since $\overline{\sigma}$ is optimal for the mean-payoff objective with reward $r_{\beta_{\text{low}}}$ and since $\beta_{\text{low}} \in [\mathsf{ERRev}^* - \epsilon, \mathsf{ERRev}^*]$, by the second part of Theorem 3.1 we have $\mathsf{ERRev}(\overline{\sigma}) \in [\mathsf{ERRev}^* - \epsilon, \mathsf{ERRev}^*]$.

Corollary 3.2 (Correctness of the analysis). *Let $\epsilon > 0$. Suppose that Algorithm 1 returns a value $\overline{\mathsf{ERRev}}$ and a strategy $\overline{\sigma}$ in $\mathcal{M}$. Then, we have $\overline{\mathsf{ERRev}}, \mathsf{ERRev}(\overline{\sigma}) \in [\mathsf{ERRev}^* - \epsilon, \mathsf{ERRev}^*]$.*

Corollary 3.3 (Formal lower bound). *Suppose that Algorithm 1 returns a value $\overline{\mathsf{ERRev}}$ and a strategy $\overline{\sigma}$. Then, there exists a selfish mining attack in the blockchain that achieves the expected relative revenue of at least $\overline{\mathsf{ERRev}}$.*

3.4 Key Features and Limitations

Key features. The key features of our selfish mining attack and formal analysis are as follows:

(1) *Fully automated analysis.* Manual (i.e. non-automated) analysis of optimal selfish mining attacks is already challenging and technically involved for PoW blockchains, where the adversary only grows a single private fork [10]. Hence, it would be even more difficult and potentially intractable in blockchains based on efficient proof systems. By modelling our selfish mining attack as an MDP and reducing the analysis to solving mean-payoff MDPs, we leverage existing methods for formal analysis of MDPs to obtain a *fully automated analysis* procedure, thus avoiding the necessity for tedious manual analyses.

(2) *Formal guarantees on correctness.* Our analysis provides *formal guarantees* on the correctness of its output. Again, this is achieved by formally reducing our problem to solving mean-payoff MDPs for which exact algorithms with formal correctness guarantees are available [17, 19].

(3) *Flexibility of the analysis.* Our analysis is agnostic to the values of system model and attack parameters and it is *flexible* to their changes. Hence, it allows us to tweak the parameter values and study their impact on the optimal expected relative revenue, while preserving formal guarantees on the correctness. To illustrate the flexibility, observe that:
   - If the attack depth d, forking number f or maximal fork length l of the attack change, then both the state space and the action space of the MDP change.
   - If the relative resource of the adversary p or the switching probability γ change, then the transition function of the MDP changes.
   - As we show in our experiments in Section 4, a change in any of these parameter values results in a change in the optimal expected relative revenue that the adversary can achieve.

   The flexibility of our analysis is thus a significant feature, since it again avoids the need for tedious manual analyses for different parameter values that give rise to different MDPs.

Limitations. While our formal analysis computes an optimal selfish mining strategy in the MDP up to a desired precision, note that there still exist selfish mining attacks that do not correspond to any strategy in our MDP model. Hence, the strategy computed by our method is optimal only with respect to the *subclass* of strategies captured by the MDP model. There are two key reasons behind the incompleteness of our MDP model:

(1) *Bounded forks.* In order to ensure finiteness of our MDP model, we impose an upper bound l on the maximal length of each private fork. This means that the adversary cannot grow arbitrarily long private forks. Since the probability of the adversary being able to grow extremely long private forks is low, we believe that this limitation does not significantly impact the expected relative revenue of selfish mining strategy under this restriction.

(2) *Disjoint forks vs fork trees.* Our attack grows private forks on different blocks in the main chain. However, rather than growing multiple disjoint private forks, a more general class of selfish mining attacks would be to allow growing *private trees.* We stick to disjoint private forks in order to preserve *computational efficiency* of our analysis, since allowing the adversary to grow private trees would result in our MDP states needing to store information about each private tree topology, which would lead to a huge blow-up in the size of the MDP. In contrast, storing disjoint private forks only requires storing fork lengths, resulting in smaller MDP models.

We conclude by noting that, while our formal analysis is incomplete due to considering a subclass of selfish mining attacks, the formal guarantees provided by our analysis still ensure that we compute a *true lower bound* on the expected relative revenue that a selfish mining attack achieves.

4 EXPERIMENTAL EVALUATION

We implement the MDP model and the formal analysis procedure presented in Section 3 and perform an experimental evaluation towards answering the following research questions (RQs):

RQ1 What is the expected relative revenue that our selfish mining strategy achieves? How does it compare to direct extensions of classic selfish mining attacks on PoW blockchains [10] or to mining honestly?

RQ2 How do different values of the System Model parameters impact the expected relative revenue that our selfish mining attack can achieve? The System Model parameters include the relative resource of the adversary $p \in [0,1]$ and the switching probability $\gamma \in [0,1]$.

Baselines. To answer these RQs, we compare our selfish mining attack against two baselines:

(1) *Honest mining strategy.* This is the strategy extending only the leading block of the main chain.
(2) *Single-tree selfish mining attack.* This is the attack strategy that exactly follows the classic selfish mining attack in Bitcoin proposed in [10], however it grows a private tree fork rather than a private chain. Analogously as in [10], the adversary publishes the private tree whenever the length of the main chain catches up with the depth of the private tree. We omit the formal model of this baseline due to space limitations. We also use this baseline to empirically evaluate how severe is the second limitation discussed in Section 3.4.

Experimental setup. We perform an experimental comparison of our attack and the two baselines for the values of the adversarial relative resource $p \in [0, 0.3]$ (in increments of 0.01) and the switching probability $\gamma \in \{0, 0.25, 0.5, 0.75, 1\}$. As for the parameters of each selfish mining attack:

- For our selfish mining attack, we set the maximal length of private forks $l = 4$ and consider all combinations $(d, f) \in \{(1,1),(2,1),(2,2),(3,2),(4,2)\}$ of the values of the attack depth d and the forking number f.
- For the single-tree selfish mining attack baseline, we set the maximal depth of the private tree $l = 4$ to match our maximal private fork length, and the maximal width of the private tree $f = 5$.

All experiments were run on Ubuntu 20, 2.80GHz 11th Gen Intel(R) Core(TM) i7-1165G7 CPU, 16 GB RAM, and 16 GB SWAP SSD Memory. For solving mean-payoff MDPs, we use the probabilistic model checker Storm [17], a popular MDP analysis tool within the formal methods community[2].

Results. Table 1 shows the runtimes of both our selfish mining attack as well as the single-tree selfish mining attack given various parameter settings and for a fixed switching parameter of $\gamma = 0.5$. We only show timings for $\gamma = 0.5$ as we found the runtimes of our experiments to be very similar across all γ parameter settings. As can be seen from Table 1, increasing the depth of the attack increases the runtime of our evaluation by an order of magnitude due to the exponential increase in state space.

[2]Refer to our github repository for our implementation details: https://github.com/mehrdad76/Automated-Selfish-Mining-Analysis-in-EPS-Blockchains

Attack Type	Parameters	Time (s)
Our Attack	$d = 1, f = 1$	3.8
Our Attack	$d = 2, f = 1$	5.4
Our Attack	$d = 2, f = 2$	61.7
Our Attack	$d = 3, f = 2$	2295.3
Our Attack	$d = 4, f = 2$	77761.7
Single-tree Selfish Mining	$f = 5$	2406.8

Table 1: Runtimes for various attack types and parameter settings and $\gamma = 0.5$.

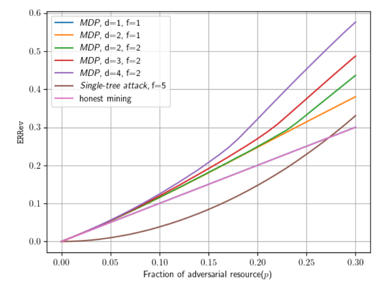
Experimental results are shown in Figure 2, showing plots for each $\gamma \in \{0, 0.25, 0.5, 0.75, 1\}$. As we can see from the plots, our selfish mining attack consistently achieves *higher expected relative revenue* ERRev than both baselines for each value of γ, except when $d = 1$ and $f = 1$. Indeed, already for $d = 2$ and $f = 1$ when the adversary grows a single private fork on the first two blocks in the main chain, our attack achieves higher ERRev than both baselines. This shows that growing private forks at two different blocks already provides a more powerful attack than growing a much larger private tree at a single block. Hence, our results indicate that growing disjoint private forks rather than trees is not a significant limitation, justifying our choice to grow private forks towards making the analysis computationally tractable.

The attained ERRev grows significantly as we increase d and f and allow the adversary to grow more private forks. In particular, for $d = 4$, $f = 2$, and relative adversarial resource $p = 0.3$, our attack achieves ERRev that is larger by at least 0.2 than that of both baselines, for all values of the switching probability γ. This indicates a *significant advantage* of selfish mining attacks in efficient proof systems blockchains compared to PoW, as the ability to simultaneously grow multiple private forks on multiple blocks translates to a much larger ERRev. Our results suggest that further study of techniques to reduce the advantage of the adversary when mining on several blocks is important in order to maintain reasonable chain quality for efficient proof systems blockchains.

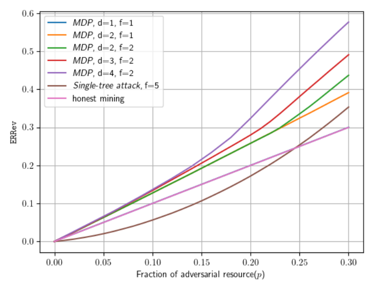
Finally, we notice that larger γ values correspond to larger ERRev in our strategies. This is expected, as larger γ values introduce bias in the likelihood of the adversarial chain becoming the main chain. This is most pertinently observed in the case of $d = f = 1$: since $d = f = 1$ corresponds to a strategy that only mines a private block on the leading block in the main chain, the only way to deviate from honest mining is to withhold a freshly mined block and reveal it together with the occurrence of a freshly mined honest block. As we can see in the plots, for $\gamma < 0.5$ the achieved ERRev of the strategy with $d = f = 1$ corresponds to that of honest mining and the two lines in plots mostly overlap, whereas this strategy only starts to pay off for $\gamma > 0.5$ and for the proportion of resource $p > 0.25$. Altogether, this suggests that further and careful analysis of the control of the adversary over the broadcast network as well as the fork choice breaking rule is necessary.

Key takeaways. The key takeaways of our experimental evaluation are as follows:

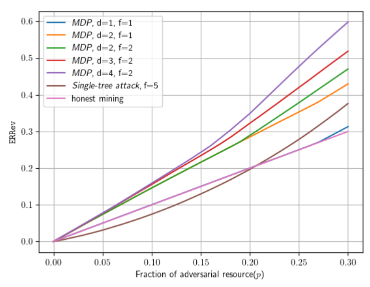
- Our selfish mining attack achieves *significantly higher* ERRev than both baselines, reaching up to 0.2 difference in ERRev. Thus, our



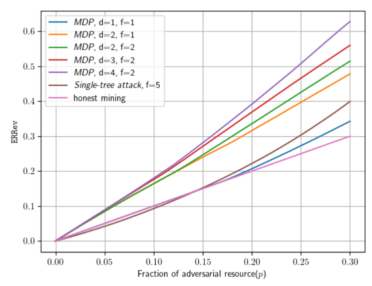
(a) $\gamma = 0$



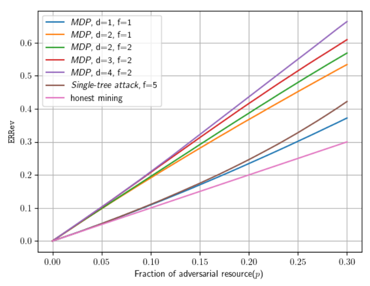
(b) $\gamma = 0.25$



(c) $\gamma = 0.5$



(d) $\gamma = 0.75$



(e) $\gamma = 1$

Figure 2: Comparison of expected relative revenue ERRev as a function of adversarial resource achieved by our attack and the baselines for different values of γ.

results strongly suggest that growing private forks at multiple blocks is much more advantageous than growing all forks on the first block in the main chain.

- Our results suggest that growing private trees rather than disjoint private forks would *not* lead to a significant improvement in the adversary's ERRev. Hence, the second limitation of our attack discussed in Section 3.4 does not seem to be significant.
- Our results suggest that enhancing security against selfish mining attacks in efficient proof system blockchains requires further and careful analysis of the control that the adversary has over the broadcast system. In particular, for large values of the switching probability γ, even the simplest variant of our attack with $d = 1$ and $f = 1$ starts to pay off when $p > 0.25$.

5 CONCLUSION

We initiated the study of optimal selfish mining strategies for unpredictable blockchain protocols based on efficient proof systems. To this end, we considered a selfish mining objective corresponding to changes in chain quality and proposed a novel selfish mining attack that aims to maximize this objective. We formally modeled our attack as an MDP strategy and we presented a formal analysis procedure for computing an ϵ-tight lower bound on the optimal expected relative revenue in the MDP and a strategy that achieves it for a specified precision $\epsilon > 0$. The procedure is fully automated and provides formal guarantees on the correctness of the computed bound.

We believe that our work opens several exciting lines for future research. We highlight two particular directions. First, our formal analysis only allows us to compute *lower bounds* on the expected relative revenue that an adversary can achieve. A natural direction of future research would be to consider computing *upper bounds* on the optimal expected relative revenue for fixed resource amounts. Second, as discussed in Section 3.4, our formal analysis only computes ϵ-tight lower bounds on the expected relative revenue by following a strategy in our MDP model. However, our model in Section 3.2 introduces assumptions such as growing private forks instead of trees and bounding the maximal length of each fork for tractability purposes. It would be interesting to study whether these assumptions could be relaxed while still providing formal correctness guarantees.

ACKNOWLEDGMENTS

This work was supported in part by the ERC-2020-CoG 863818 (FoRM-SMArt) grant and the MOE-T2EP20122-0014 (Data-Driven Distributed Algorithms) grant.

REFERENCES

[1] Adam Back. 1997. Hashcash. http://hashcash.org/.
[2] Vivek Kumar Bagaria, Amir Dembo, Sreeram Kannan, Sewoong Oh, David Tse, Pramod Viswanath, Xuechao Wang, and Ofer Zeitouni. 2022. Proof-of-Stake Longest Chain Protocols: Security vs Predictability. In *Proceedings of the 2022 ACM Workshop on Developments in Consensus, ConsensusDay 2022, Los Angeles, CA, USA, 7 November 2022*, Jorge M. Soares, Dawn Song, and Marko Vukolic (Eds.). ACM, 29–42. https://doi.org/10.1145/3560829.3563559
[3] Jonah Brown-Cohen, Arvind Narayanan, Christos-Alexandros Psomas, and S. Matthew Weinberg. 2018. Formal Barriers to Longest-Chain Proof-of-Stake

Protocols. *CoRR* abs/1809.06528 (2018). arXiv:1809.06528 http://arxiv.org/abs/1809.06528

[4] Krishnendu Chatterjee, Amirali Ebrahimzadeh, Mehrdad Karrabi, Krzysztof Pietrzak, Michelle Yeo, and Djordje Zikelic. 2024. *Fully Automated Selfish Mining Analysis in Efficient Proof Systems Blockchains*. Technical Report 2405.04420. arXiv. https://doi.org/10.48550/arXiv.2405.04420 Full version of this paper.

[5] Jing Chen and Silvio Micali. 2019. Algorand: A secure and efficient distributed ledger. *Theor. Comput. Sci.* 777 (2019), 155–183.

[6] Bram Cohen and Krzysztof Pietrzak. 2019. The chia network blockchain. https://docs.chia.net/assets/files/Precursor-ChiaGreenPaper-82cb50060c575f3f71444a4b7430fb9d.pdf

[7] Bram Cohen and Krzysztof Pietrzak. 2023. Chia Greenpaper. https://docs.chia.net/green-paper-abstract

[8] Bernardo Machado David, Peter Gazi, Aggelos Kiayias, and Alexander Russell. 2017. Ouroboros Praos: An adaptively-secure, semi-synchronous proof-of-stake protocol. *IACR Cryptol. ePrint Arch.* (2017), 573. http://eprint.iacr.org/2017/573

[9] Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, and Krzysztof Pietrzak. 2015. Proofs of Space. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 9216)*, Rosario Gennaro and Matthew Robshaw (Eds.). Springer, 585–605. https://doi.org/10.1007/978-3-662-48000-7_29

[10] Ittay Eyal and Emin Gün Sirer. 2018. Majority is not enough: bitcoin mining is vulnerable. *Commun. ACM* 61, 7 (2018), 95–102. https://doi.org/10.1145/3212998

[11] Lei Fan and Hong-Sheng Zhou. 2017. iChing: A Scalable Proof-of-Stake Blockchain in the Open Setting (or, How to Mimic Nakamoto's Design via Proof-of-Stake). *IACR Cryptol. ePrint Arch.* (2017), 656. http://eprint.iacr.org/2017/656

[12] Matheus V. X. Ferreira, Ye Lin Sally Hahn, S. Matthew Weinberg, and Catherine Yu. 2022. Optimal Strategic Mining Against Cryptographic Self-Selection in Proof-of-Stake. In *EC*. ACM, 89–114.

[13] Matheus V. X. Ferreira and S. Matthew Weinberg. 2021. Proof-of-Stake Mining Games with Perfect Randomness. In *EC '21: The 22nd ACM Conference on Economics and Computation, Budapest, Hungary, July 18-23, 2021*, Péter Biró, Shuchi Chawla, and Federico Echenique (Eds.). ACM, 433–453. https://doi.org/10.1145/3465456.3467636

[14] Jerzy Filar and Koos Vrieze. 2012. *Competitive Markov decision processes.* Springer Science & Business Media.

[15] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. 2015. The Bitcoin Backbone Protocol: Analysis and Applications. In *EUROCRYPT (2) (Lecture Notes in Computer Science, Vol. 9057)*. Springer, 281–310.

[16] Peter Gazi, Aggelos Kiayias, and Alexander Russell. 2020. Tight Consistency Bounds for Bitcoin. In *CCS*. ACM, 819–838.

[17] Christian Hensel, Sebastian Junges, Joost-Pieter Katoen, Tim Quatmann, and Matthias Volk. 2022. The probabilistic model checker Storm. *Int. J. Softw. Tools Technol. Transf.* 24, 4 (2022), 589–610. https://doi.org/10.1007/s10009-021-00633-z

[18] Charlie Hou, Mingxun Zhou, Yan Ji, Phil Daian, Florian Tramèr, Giulia Fanti, and Ari Juels. 2021. SquirRL: Automating Attack Analysis on Blockchain Incentive Mechanisms with Deep Reinforcement Learning. In *NDSS*. The Internet Society.

[19] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. 2011. PRISM 4.0: Verification of Probabilistic Real-Time Systems. In *CAV (Lecture Notes in Computer Science, Vol. 6806)*. Springer, 585–591.

[20] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf.

[21] Sunoo Park, Albert Kwon, Georg Fuchsbauer, Peter Gazi, Joël Alwen, and Krzysztof Pietrzak. 2018. SpaceMint: A Cryptocurrency Based on Proofs of Space. In *Financial Cryptography (Lecture Notes in Computer Science, Vol. 10957)*. Springer, 480–499.

[22] Rafael Pass, Lior Seeman, and Abhi Shelat. 2017. Analysis of the Blockchain Protocol in Asynchronous Networks. In *EUROCRYPT (2) (Lecture Notes in Computer Science, Vol. 10211)*. 643–673.

[23] Martin L. Puterman. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* Wiley.

[24] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. 2016. Optimal Selfish Mining Strategies in Bitcoin. In *Financial Cryptography (Lecture Notes in Computer Science, Vol. 9603)*. Springer, 515–532.

[25] Roozbeh Sarenche, Svetla Nikova, and Bart Preneel. 2024. Deep Selfish Proposing in Longest-Chain Proof-of-Stake Protocols. In *Financial Cryptography and Data Security*.

[26] Xuechao Wang, Govinda M. Kamath, Vivek Kumar Bagaria, Sreeram Kannan, Sewoong Oh, David Tse, and Pramod Viswanath. 2019. Proof-of-Stake Longest Chain Protocols Revisited. *CoRR* abs/1910.02218 (2019).

[27] Roi Bar Zur, Ittay Eyal, and Aviv Tamar. 2020. Efficient MDP Analysis for Selfish-Mining in Blockchains. In *AFT*. ACM, 113–131.