Selfish Mining in Public Blockchains: A Quantitative Analysis

Daria Smuseva(✉), Andrea Marin, and Sabina Rossi

Università Ca' Foscari Venezia, Venice, Italy
{daria.smuseva,marin,sabina.rossi}@unive.it

Abstract. Blockchains are digital ledgers of transactions that aim to be decentralized, secure, and tamper-proof. To achieve this goal, they rely on a consensus algorithm, with the most well-known being the proof-of-work (PoW) algorithm. In PoW, a group of specialized users known as miners invest a significant amount of energy to secure the blockchain ledger. Miners are incentivized to participate in the network through the potential rewards they can earn, which are based on the number of blocks they are able to consolidate and add to the chain. An important characteristic of the PoW algorithm is that miners' rewards must be statistically proportional to the amount of computational power (and hence energy) invested in this process. In this work, we study the selfish miner attack by means of a stochastic model based on a quantitative process algebra. When a successful attack occurs, a miner or mining pool is able to receive more rewards than they should, at the expense of other miners. The model analysis allows us to derive the conditions under which the attack becomes convenient for the miners.

Keywords: Blockchain security · Stochastic process algebra · Selfish mining attack

1 Introduction

A blockchain is a digital ledger that is decentralized and distributed, where records are stored in blocks. Transactions are added to the network after being validated by a group of specialized users called *miners*, who bundle them into blocks. Once a transaction is added to the blockchain, it becomes highly resistant to alteration or deletion, making the records permanent and immutable.

Each blockchain network employs a consensus protocol to achieve consensus on the validity of transactions. The most commonly used consensus protocol is the Proof-of-Work (PoW) protocol, initially introduced in the original Bitcoin blockchain [14]. This protocol requires miners to compete to solve a complex computational problem, with the first miner to solve it receiving a reward. The primary advantage of PoW is that any attempt to modify transactions in the

E. Kalyvianaki and M. Paolieri (Eds.): VALUETOOLS 2023, LNICST 539, pp. 18–32, 2024.
https://doi.org/10.1007/978-3-031-48885-6_2

ledger becomes prohibitively expensive for malicious actors, as they must re-mine all subsequent blocks, making it computationally infeasible.

The Bitcoin blockchain features a lightweight scripting language that allows users to specify transaction validity conditions. Advanced blockchains, such as Ethereum [3], have introduced Turing-complete scripting languages that enable users to encode any computation as a script. The computational framework of scriptable cryptocurrencies has been referred to as *consensus computer* (see [10]). Miners within consensus computers are responsible for two primary functions: verifying the correct construction of blocks, and assessing the validity of transactions included within each block. While verifying the correct block construction is relatively straightforward, validating transactions within a block can take significantly more time due to the execution of corresponding code fragments. However, miners are incentivized to verify these scripted transactions to support the common good of the cryptocurrency.

In most of public blockchains with PoW consensus mechanism, miners who successfully solve a cryptopuzzle are given the chance to record a set of transactions and collect a cryptocurrency reward. The greater the mining power a miner employs, the higher the likelihood of being the first to solve the puzzle. This reward structure acts as an incentive for miners to allocate their resources towards the system, ultimately upholding the decentralized nature of the currency.

The PoW protocol requires that most miners operate honestly and follow the prescribed protocol. However, if a group of colluding miners controls the majority of the mining power in the network, the currency's decentralized nature is lost, and the colluding group governs it. This scenario could result in the prohibition of certain transactions, or even all transactions.

Empirical evidences and game theoretical models have revealed that Bitcoin miners are strategic in nature and frequently form mining pools [9]. These pools are created to reduce the variance of each member's income rate, given that rewards are dispersed at infrequent, random intervals. All members of the pool contribute to the solution of each cryptopuzzle and receive rewards in proportion to their level of involvement.

Initially, it was assumed that there is no advantage for colluding miners to organize into ever-increasing pools and it poses no threat to the system [2]. However, in [5] the authors have described a strategy that can be used by a minority pool to obtain more revenue than the pool's fair share, that is, more than its ratio of the total mining power.

The Selfish Mining strategy is based on the idea of a mining pool keeping its discovered blocks confidential and intentionally creating a fork, i.e., two branches, in the blockchain. Meanwhile, honest miners continue to mine on the public chain, unaware of the pool's private branch. Whenever the pool discovers more blocks, it strengthens its lead on the public chain and continues to keep its blocks secret. When the public branch approaches the pool's private branch in length, the selfish miners reveal blocks from their private chain to the public

one, causing the public chain to become the shorter branch and their private chain to become the longer one.

This strategy results in a wastage of resources for honest miners who follow the blockchain protocol by solving cryptopuzzles that serve no objective. Although both honest and selfish miners squander some resources, honest miners bear a higher percentage of the wastage. Moreover, the rewards obtained by the selfish pool exceed its portion of the network's mining power, providing it with a competitive edge and encouraging rational miners to join the selfish mining pool.

To completely understand the selfish miner attack, we must consider some important facts. Firstly, the selfish mining attack does not compromise the integrity of the blockchain, i.e., the validity and immutability of the data is still preserved. Secondly, several factors contribute to the success of a selfish miner attack, including the ability of the mining pool to efficiently and rapidly propagate its blocks. Therefore, even if the pool is unable to create a longer chain than the one mined by fair miners, there is still a chance that it can propagate its block quickly enough to make it the official branch of the fork.

In this work, we take a quantitative approach to analyze the selfish mining attack by presenting a model expressed with a Markovian process algebra, namely the Performance Evaluation Process Algebra (PEPA) [7]. With respect to previous studies, we consider the effect of the verification time on the selfish mining strategy. Since with the introduction of smart contacts this may be not negligible, as discussed in the seminal work [5], our model is applicable to a broader range of scenarios. Our findings suggest that slower verification times provide an advantage to the selfish miners.

The paper is structured as follows. In Sect. 2, we review the state of the art. Section 3 describes the mining process and the selfish mining attack. Moreover, we briefly introduce PEPA to keep the paper self-contained. In Sect. 4, we describe the model and in Sect. 5, we show the exact solution of its lumping. This allows us to derive the conditions under which the selfish mining attack is rationally convenient for miners. Finally, Sect. 6 concludes the paper.

To the best of our knowledge, this is the first work that models the selfish miner attack by means of a Markovian process algebra.

2 Related Work

The selfish mining attack in Bitcoin was studied in [4–6] where a Markovian model was proposed to demonstrate that a mining pool with a sufficiently large fraction of computational power f (but lower than 50%) can receive a higher fraction of rewards than f. Although the attack does not affect the integrity of the transactions stored in the ledger, the authors emphasize that the economic mechanism that supports the mining process is critical to the blockchain's security. Furthermore, the overproduction of chain forks caused by the selfish mining behaviour, causes a waste of hash power reducing the security of the ledger. With respect to this work, our results are more general since we account for the

block verification time which was regarded as negligible in [4–6]. We show that this parameter has a significant impact on selfish miners' expected reward.

Selfish mining has faced criticism in [17] where the author assumes that the attack becomes convenient when the selfish pool can mine two consecutive blocks while fair miners are still working on one block. However, this assumption overlooks the fact that the mining pool can announce its first block as soon as a fair miner announces a consolidated block. This gives the pool a chance to win the race for making its branch of the fork successful, even if it fails to mine the second block in time. However, the success of this strategy heavily relies on the network connectivity and the propagation time of the blocks announced by the pool.

The impact of selfish mining attacks on the Bitcoin network is extensively studied in [13]. The authors show that the impact of selfish mining on the network performance is noticeable and that the expected number of forks observed in the blockchain is higher than what would be expected if all miners were fair. This suggests that the selfish miner attack is still a major problem in blockchains.

Quantitative analysis has been widely employed in blockchain studies to gain insights into various aspects of blockchain systems such as performance, security, and scalability. Through quantitative analysis, researchers have been able to evaluate the behavior of blockchain networks, understand the dynamics of transaction processing, and identify potential vulnerabilities in the system. For instance, in [15] the authors propose a queueing model and apply it to study the performance of Bitcoin blockchain.

Building on [16], where PEPA process algebra was used to model and analyze the Verifier's Dilemma in the Ethereum Classic blockchain, we apply PEPA to model a public blockchain with PoW and investigate the Selfish Mining attack while taking into account the block verification time.

3 Background

In this section, we provide an overview of the selfish miner attack problem, as well as a brief introduction to the notation and important aspects of PEPA.

3.1 Selfish Mining

When two miners create a block with the same prior block, the chain splits into two branches, resulting in a *fork*. Miners can add valid blocks to either branch, mining on whichever they choose. However, to maintain a consistent chain, miners follow a globally-agreed upon sequence of transactions. The protocol requires miners to pick the longest chain and add blocks to it. If a miner's successful mining result is discarded from the network, they lose the potential reward.

Assuming a PoW-based blockchain network where the majority of miners follow the honest mining strategy, a colluding minority pool may adopt the selfish mining strategy. The key idea behind selfish mining is to compel honest miners to expend their resources on a branch of the blockchain that ultimately will not be included in the final chain.

In order to increase their revenue, selfish miners keep their mined blocks private, which results in the creation of a private branch causing the bifurcation of the blockchain. The pool gradually reveals blocks from the private branch, causing honest miners to switch to the newly revealed blocks and abandon the shorter public branch. This results in the invalidation of the honest miners' past efforts on the public branch and empowers the selfish pool to accumulate greater reward by incorporating a greater proportion of its blocks into the blockchain.

As the selfish mining pool has only a fraction of the total mining power, its private branch cannot remain longer than the public branch indefinitely. Therefore, when the public branch becomes longer, the selfish mining pool adopts it and starts mining on top of the current public head.

When the selfish miners discover a new block, instead of publishing it they keep it private and keep mining. If the selfish miners find a second block before the honest miners can reveal their first block, the selfish pool will have a longer private branch and will lead the public chain. Even if the honest miners reveal their first block at this point, the selfish pool will still receive a reward for publishing their two blocks on the private branch. However, since the selfish pool has less hash power, there is a high likelihood that the public branch will eventually become longer than the private branch.

When the selfish pool has a one-block lead and a honest miner discovers a block on the public branch, the selfish pool will immediately publish the private branch, triggering a race between the two branches. The selfish miners will extend their private branch while the honest miners will choose which branch to mine on based on notification propagation. At this point, there are three possible outcomes: the selfish pool successfully mines the second block on their first block, making their branch longer and earning revenue for both blocks; a honest miner builds a block after the pool's first block, allowing the pool to earn revenue from the first block and the honest miner to earn revenue from the second block; or the pool gets nothing if honest miners mine a block after their own block.

3.2 PEPA

We use the Performance Evaluation Process Algebra (PEPA) [7] to investigate the effects of Selfish mining on miners' behavior in public blockchains. PEPA is a compositional modeling language that is supported by a tool that can be applied to Eclipse, known as the PEPA Eclipse Plug-in. One of the main advantages of using PEPA is that it has an underlying stochastic process, which is a continuous time Markov process under certain assumptions.

In PEPA, a system specification comprises a group of active agents or components that work together through activities to accomplish the desired system behavior. The syntax of PEPA terms is determined by the following grammar:

$$S ::= (\alpha, r).S \;\mid\; S + S \;\mid\; A \qquad\qquad P ::= P \underset{L}{\bowtie} P \;\mid\; P/L \;\mid\; S$$

where S denotes a *sequential component*, while P denotes a *model component* which can be obtained as the cooperation of sequential terms. The meaning of

the operators is the following: $(\alpha, r).S$ performs the activity (α, r) with action type α and rate r and subsequently behaves as S. $S_1 + S_2$ specifies a system which may behave either as S_1 or as S_2. The meaning of a constant A is given by a defining equation such as $A \stackrel{def}{=} P$ which gives the constant A the behaviour of the component P. The component P/L behaves as P except that any activity of type within the set L is relabelled with the *unknown type* τ. The cooperation combinator $\underset{L}{\bowtie}$ represents an interaction between two components, which is determined by the *cooperation set* L of action types. Activities with action types in the cooperation set L, called *shared* activities, require the synchronisation of the components. It is assumed that each component proceeds independently with the activities whose types do not occur in the cooperation set L. The duration of a shared activity is reflected by the rate of the slower participant. If in a component an activity has the *unknown rate* $\top$, then the rate of the shared activity will be that of the other component.

The stationary throughput of action a, denoted as X_a, is calculated as the sum of the stationary probabilities of all states where action a is enabled, each multiplied by the respective rate.

4 Analysis of the Selfish Mining Strategy

In this section, we present a PEPA model for a blockchain network with PoW consensus mechanism with fair miners and one selfish miner with arbitrary hash power. We assume that all miners have the same computational power γ, and verification is performed in an exponentially distributed time with mean β^{-1}.

4.1 PEPA Model for Networks of All Fair Miners

We first inspect the behaviour of miners who follow the blockchain consensus protocol by verifying all new blocks in the network. Let $M_1, \ldots, M_K$ be a network of K fair miners. The specification of a single fair (verifying) miner, say M_1, can be expressed in PEPA as

$$M_1 \stackrel{def}{=} (m_1, \gamma).M_1 + (m_2, \top).V_1 + \ldots + (m_K, \top).V_1$$
$$V_1 \stackrel{def}{=} (v_2, \beta).M_1 + \ldots + (v_K, \beta).M_1 + (m_2, \top).V_1 + \ldots + (m_K, \top).V_1$$

Miner M_1 mines a new block with action type m_1 and rate γ. Then, it returns to its initial state and it starts to mine another block. Whenever M_1 gets a block from the network through an activity $(m_j, \top)$ with $j \in \{2, \ldots, K\}$, then it starts the verification process and moves to state V_1.

We assume that all miners possess an equal proportion of computational power, and thus they perform verification in a synchronous manner. The block announced by a miner M_i is verified by all the other miners with rate β and action type v_i. It is evident that miners verify all blocks on the network, except for the block they have created themselves. During the verification phase, M_1

Table 1. PEPA model of a network with K fair miners

M_i	$\stackrel{def}{=}$	$(m_i, \gamma).M_i + \sum_{j \neq i}(m_j, \top).V_i$
V_i	$\stackrel{def}{=}$	$\sum_{j \neq i}(v_j, \beta).M_i + \sum_{j \neq i}(m_j, \top).V_i$
Network	$\stackrel{def}{=}$	$(..((M_1 \underset{L \cup L_{12}}{\bowtie} M_2) \underset{L \cup L_3}{\bowtie}) \underset{L \cup L_4}{\bowtie} \ldots) \underset{L \cup L_K}{\bowtie} M_K$
where		$i, j \in \{1, \ldots, K\}$ and $L = \{m_1, \ldots, m_K\}, L_{12} = \{v_3, \ldots, v_K\}$, $L_j = \{v_1, \ldots, v_K\} \setminus \{v_j\}$ for $j \geq 3$

can still get new blocks from the rest of the network through an activity $(m_j, \top)$ with $j \in \{2, \ldots, K\}$.

The specification of a network composed by $K \geq 3$ fair miners is reported in Table 1. Notice that, according to the operational semantics of PEPA, if miner M_i announces a block, m_i is received by all miners except M_i. Thus, all miners moves to state V_j with $j \neq i$, while M_i is still mining. Now, according to the synchronization operation specified in Table 1, all miners in V_j synchronize on the intersection of all synchronizing sets that is exactly equal to $\{v_i\}$, hence once the verification is completed, they all simultaneously move to the mining phase due to action v_i, as required.

The derivation graph of the model in Table 1 is depicted in Fig. 1.

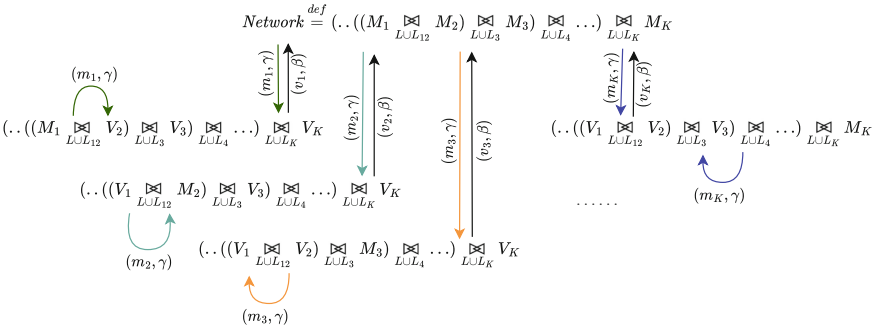


Fig. 1. Derivation graph of the model in Table 1

4.2 PEPA Model for Networks of Fair and Selfish Miners

Suppose the network contains a selfish mining pool that controls the amount of hash power $w\gamma$, and it uses action types m_{S_1} and m_{S_2} to mine blocks. The PEPA specification of a network with K honest miners and one selfish miner M_S, representing the mining pool, is presented in Table 2. The action type m_{S_1}

Table 2. PEPA model of a network with K fair miners and one selfish pool M_S

M_{F_i}	$\stackrel{def}{=}$	$(m_{F_i}, \gamma).M_{F_i} + \sum_{j \neq i}(m_{F_j}, \top).V_i + (m_{S_2}, \top).V_{i_S}$
V_i	$\stackrel{def}{=}$	$\sum_{j \neq i}(v_j, \beta).M_{F_i} + \sum_{j \neq i}(m_{F_j}, \top).V_i$
V_{i_S}	$\stackrel{def}{=}$	$(v_S, \beta).M_{F_i} + (m_{S_2}, \top).V_{i_S}$
M_S	$\stackrel{def}{=}$	$(m_{S_1}, w\gamma).C + \sum_i (m_{F_i}, \top).V_S$
C	$\stackrel{def}{=}$	$(m_{S_2}, w\gamma).M_S + \sum_i (m_{F_i}, \top).V_S$
V_S	$\stackrel{def}{=}$	$\sum_i (v_i, \beta).M_S + \sum_{j \neq i}(m_{F_j}, \top).V_S$
Network	$\stackrel{def}{=}$	$M_S \underset{L \cup V}{\bowtie} (..((M_{F_1} \underset{L \cup V_{12}}{\bowtie} M_{F_2}) \underset{L \cup V_3}{\bowtie} M_{F_3}) \dots) \underset{L \cup V_K}{\bowtie} M_{F_K})$
where		$i, j \in \{1, \dots, K\}$ and $L = \{m_{S_2}, m_1, \dots, m_K\}, V = \{v_1, \dots, v_K\},$
		$V_{12} = \{v_S, v_3, \dots, v_K\}, V_j = \{v_S, v_1, \dots, v_K\} \setminus \{v_j\}$ for $j \geq 3$

refers to the first block mined by the selfish pool, which is kept private, creating a separate branch, while m_{S_2} describes the second successful block mined by the selfish pool. Once the second block is produced, two blocks are revealed from the private branch to the public, causing the rest of the network to enter the verification phase and switch from the shorter public branch to the recently revealed blocks. This behavior is captured in the model by the fact that all nodes in the network synchronize on the action type v_S when the second block of the selfish pool is announced.

Observe that, for the sake of simplicity, in our model the selfish miner discloses two blocks from their private branch to the public. Nonetheless, the model can be extended to encompass a longer queue of blocks.

Figure 2 represents the derivation graph of the model in Table 2, while Fig. 3 represents its underlying Markov chain.

4.3 Lumped Model

The detailed model that represents the state of each miner is not scalable, as the total number of miners in a blockchain is typically very large. To address this issue, we propose using a representation based on aggregation through lumping [1,8,11,12] all fair miners into an environment.

Let K be the number of fair miners. The PEPA specification of the lumped model is represented in Table 3, its derivation graph is depicted in Fig. 4 and the underlying CTMC is reported in Fig. 5.

Component E_F represents the behaviour of K fair miners which mine new blocks with action type m_{E_F} and rate $K\gamma$. Hence, $K\gamma$ is the total hash power held by fair miners. When a new block is produced by one fair miner, the environment

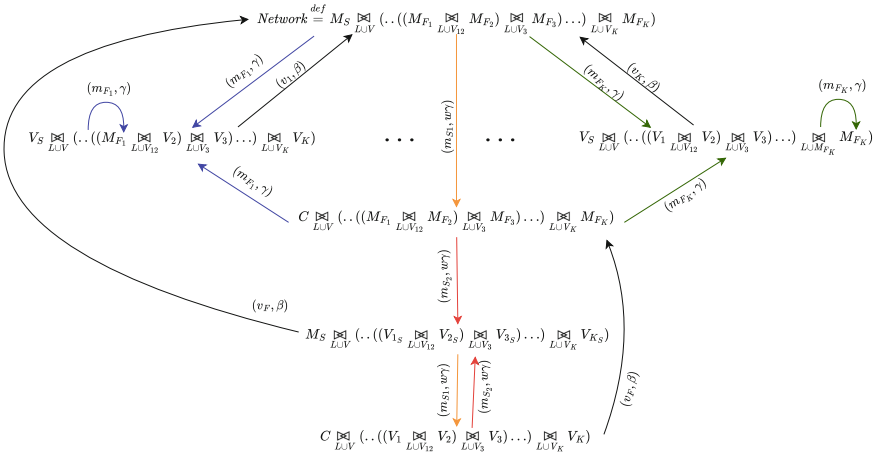


Fig. 2. Derivation graph of the model in Table 2

moves to state V_{E_F} where the miner that has proposed the block starts mining the next block, while the others verify the newly shared block. When the selfish pool reveals a block with action m_{S_2} then the environment moves to state V_{E_S} and verifies it with action v_S. The PoW race policy requires summing the rates of miners. However, the verification phase is not subject to the race policy and operates at the single miner rate β. Since all miners perform the same operations with equal computational power, the aggregated speed remains β. Additionally, since the time for solving PoW is independent and exponentially distributed by design, the aggregation of the rates is exact.

Meanwhile, the selfish miner M_S creates its first block with action type m_{S_1} and rate $w\gamma$. However, it keeps its mined block private, secretly creating a fork. The second block is created in the cheating phase C with action type m_{S_2} and rate $w\gamma$. At this point the selfish miner strategically reveals two blocks from the private branch to the public, causing the honest miners to switch to the recently revealed blocks and abandon the shorter public branch. Once the selfish miner gets a block from the network, it starts the verification process V_S. The verification of a block occurs with action v_{E_F} and rate β.

The *Lumped_Network* component is obtained as the cooperation of E_F and M_S over the set of actions L.

4.4 Performance Indices

We assume that the reward that miners gain is proportional to their *effective throughput*, which represents the proportion of their computational power that is used for mining after taking into account the necessary validation of new blocks. This effective throughput is lower than the originally invested hash power, as all miners in our model are fair and validate new blocks.

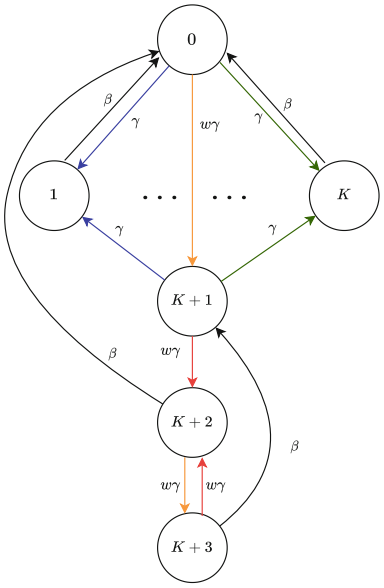


Fig. 3. Markov chain of the model underlying the derivation graph in Fig. 2

Table 3. Lumped PEPA model of a network with K fair miners aggregated in E_F and one selfish miner

E_F	$\stackrel{def}{=}$	$(m_{E_F}, K\gamma).V_{E_F} + (m_{S2}, \top).V_{E_S}$
V_{E_F}	$\stackrel{def}{=}$	$(v_{E_F}, \beta).E_F + (m_{E_F}, \gamma).V_{E_F}$
V_{E_S}	$\stackrel{def}{=}$	$(v_S, \beta).E_F + (m_{S2}, \top).V_{E_S}$
M_S	$\stackrel{def}{=}$	$(m_{S1}, w\gamma).C + (m_{E_F}, \top).V_S$
C	$\stackrel{def}{=}$	$(m_{S2}, w\gamma).M_S + (m_{E_F}, \top).V_S$
V_S	$\stackrel{def}{=}$	$(v_{E_F}, \beta).M_S + (m_{E_F}, \top).V_S$
Lumped_Network	$\stackrel{def}{=}$	$E_F \underset{L}{\bowtie} M_S$
where		$L = \{m_{E_F}, m_{S2}, v_{E_F}\}$

Note that in our model two scenarios of successful selfish mining are captured. Action type m_{S1} refers to the production of the first block in the private branch. With certain probabilities p and $(1-p)$ the block is accepted or rejected by the network, respectively. Thus, for the calculation of the effective throughput we need to consider the throughput of m_{S1} being able to impose the block without producing the second, meaning $X_p = p(X_{m_{S1}} - X_{m_{S2}})$. Action type m_{S2} refers

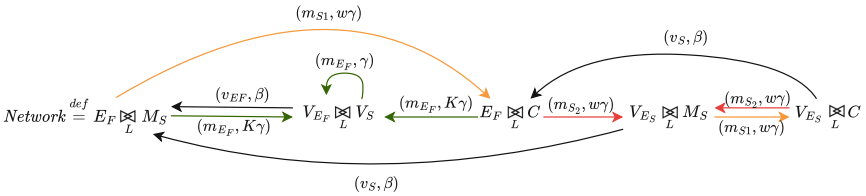


Fig. 4. Derivation graph of the model in Table 3

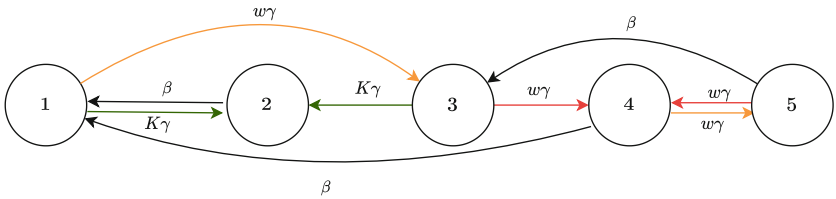


Fig. 5. Markov chain of the model underlying the derivation graph in Fig. 4

to the production of the second block, and it should be noted that whenever the selfish pool successfully propagates their private branch with two blocks, they receive a reward for both of them. Therefore, the total reward of the selfish pool is given by $X_S = 2X_{m_{S2}} + X_p = (2-p)X_{m_{S2}} + pX_{m_{S1}}$.

Every time the selfish pool successfully propagates its private branch with two blocks, one block mined by a fair miner in the network is discarded. Thus, the fair environment has an effective throughput given by

$$X_{EF} = X_{m_{EF}} - X_{m_{S2}} - X_p = X_{m_{EF}} - (1-p)X_{m_{S2}} - pX_{m_{S1}}.$$

Consequently, the effective throughput of one fair miner is given by

$$X_F = \frac{X_{EF}}{K} = \frac{X_{m_{EF}} - (1-p)X_{m_{S2}} - pX_{m_{S1}}}{K}.$$

The effective throughput of the selfish pool per unit of hash power invested in the mining process can be obtained as:

$$X_S^N = \frac{X_S}{w} = \frac{(2-p)X_{m_{S2}} + pX_{m_{S1}}}{w}.$$

Finally, we introduce the revenue of the selfish pool that is calculated as

$$R = \frac{X_S}{X_S + X_{EF}}.$$

As we may see, R measures the fraction of rewards obtained by a selfish pool.

Table 4. Parameters used for the case study

Parameter	Value
K	100 fair miners
γ	8.3×10^{-4} blocks/s
β	0.314 s^{-1}
w	100

5 Stationary Analysis and Numerical Evaluation

In this section, we give explicit expressions for the steady-state probabilities and reason about the conditions under which a selfish mining pool has a advantage over fair miners.

5.1 Steady-State Probabilities

The limited number of states of the lumped process allows us to derive the steady-state probability distribution explicitly. The symbolic expression of the steady-state probabilities of the Markov chain underlying the *Lumped_Network* depicted in Table 3 are:

$$\pi_1 = \frac{\beta(\beta(K+w)+\gamma w(2\,K+w))}{G}\,, \quad \pi_2 = \frac{\gamma K(\beta(K+2w)+\gamma w(2\,K+3w))}{G}$$

$$\pi_3 = \frac{\beta w(\beta+2\gamma w)}{G}\,, \qquad \pi_4 = \frac{\gamma w^2(\beta+\gamma w)}{G}\,, \qquad \pi_5 = \frac{\gamma^2 w^3}{G}\,,$$

where the normalizing constant is

$$G = \gamma K^2(\beta+2\gamma w) + K(\beta+\gamma w)(\beta+3\gamma w) + 2w(\beta+\gamma w)^2\,.$$

For instance, if we use the parameters specified in Table 4, we obtain:

$$\pi_1 \approx 0.475964,\ \pi_2 \approx 0.1946725,\ \pi_3 \approx 0.260505,\ \pi_4 \approx 0.0569526,\ \pi_5 \approx 0.011907.$$

Notice that, from the steady-state probability distribution, we can compute the performance metrics described in Sect. 4.4.

5.2 The Convenience of Selfish Behaviors

It should be clear that it is not easy for a mining pool to understand if a selfish behaviour is profitable. Intuitively, if the probability to generate two consecutive blocks in the time in which fair miners consolidate a single block is negligible, then the selfish pool would have a lower revenue than fair miners. Moreover, it is also unclear how the verification time impacts on this considerations.

In this subsection, we try to unveil some of the aspects that play a role in driving the decision of a rational mining pool about behaving in a selfish manner.

To make a fair comparison, let v be the fraction of hash power controlled by the selfish mining pool, i.e., $v = w/(w + K)$. We must assume that $v < 0.5$, otherwise the blockchain would be vulnerable to the 50% attack and hence would not be considered secure for more severe reasons than the selfish miner attack. In Figs. 6a, 6c, 6e, we compare the effective throughput of a single fair miner with the normalized effective throughput of a selfish miner for $p = 0.2$, $p = 0.5$ and $p = 0.8$. The other network parameters are those of Table 4. We notice that when the mining pool is able to quickly propagate its blocks as soon as a fair block is announced, the fraction of hash power that it has to control to overtake fair miners' profit becomes smaller.

Figures 6b, 6d, 6f show the revenue of the selfish pool as function of the fraction of controlled hash power. This is compared with the revenue it would obtain in a network where everyone (including the pool itself) is fair, i.e., when its revenue is equal to the fraction of controlled hash power. This experiments are coherent with the ones in [5,13] where the authors do not consider the impact of the verification times. This is somehow expected since, in our case, $\beta >> \gamma$.

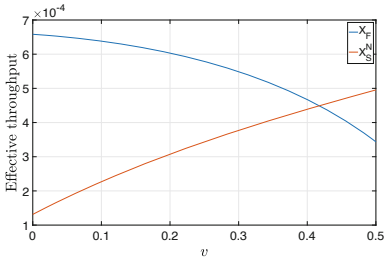
It is useful now to investigate the impact of the verification time on the point in which the selfish mining pool has an advantage over the fair miners. Formally, let w^* be the minimum positive solution in w of the equation $X_F = X_S^N$ and let $v^* = w^*/(w^* + K)$, i.e., the minimum fraction of hash power that must be controlled by the selfish pool to take advantage by the attack.

Figure 7 shows v^* as function of β. This analysis shows that slower verification times drastically reduce the demand of hash power for the greedy miners. As a consequence, blockchains with smart contracts that require longer block verification times are even more exposed to the selfish miner attack than Bitcoin, where β is very high.

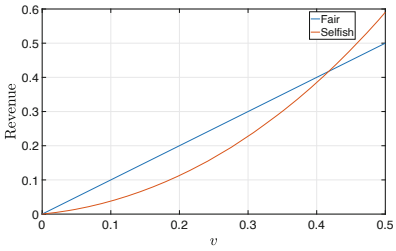
6 Conclusion

In conclusion, our study provides a quantitative analysis of the selfish miner attack in blockchain systems based on a stochastic model expressed with the Performance Evaluation Process Algebra. We have shown that the verification time of the transactions affects the rationality of the attacker, and we have derived the conditions under which the attack becomes convenient for selfish miners. This work contributes to the understanding of the selfish miner attack and can help in the development of more robust and efficient blockchain systems. Further research can explore the extension of our model to consider more complex scenarios and the application of our findings in real-world blockchain systems.

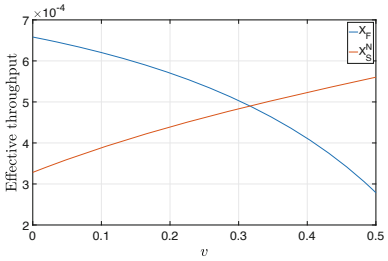
Acknowledgements. This work is partially supported by the Project NiRvAna: Noninterference and Reversibility Analysis in Private Blockchains (20202FCJMH) funded by MUR Progetti di Ricerca di Rilevante Interesse Nazionale (PRIN 2020), and by the project SERICS (PE00000014) under the



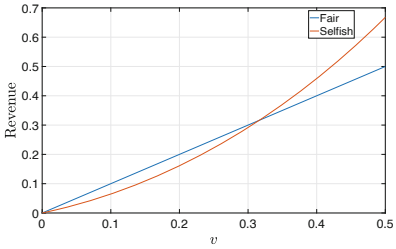
(a) Normalized effective throughput



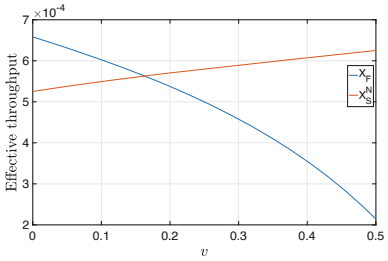
(b) Revenue of a totally fair network



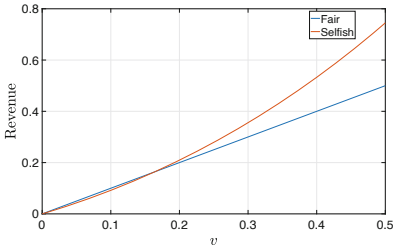
(c) Normalized effective throughput



(d) Revenue of a totally fair network

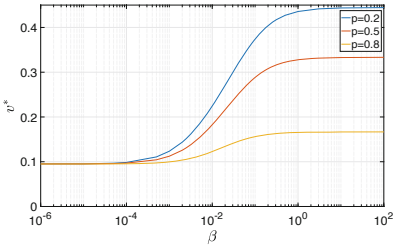


(e) Normalized effective throughput

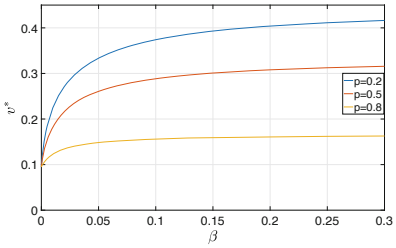


(f) Revenue of a totally fair network

Fig. 6. Selfish mining pool with $p = 0.2$ (a)-(b), $p = 0.5$ (c)-(d), $p = 0.8$ (e)-(f)



(a) Semi-logarithmic scale



(b) Linear scale with $\beta \in [0, 0.3]$

Fig. 7. How different β affects the minimum fraction of selfish miners at which they overtake the revenue with the different probability

MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU.

References

1. Alzetta, G., Marin, A., Piazza, C., Rossi, S.: Lumping-based equivalences in Markovian automata: Algorithms and applications to product-form analyses. Inf. Comput. **260**, 99–125 (2018)
2. Barber, S., Boyen, X., Shi, E., Uzun, E.: Bitter to better-how to make bitcoin a better currency. In: Financial Cryptography and Data Security Conference, FC 2012, pp. 399–414. Springer (2012)
3. Buterin, V., et al.: Ethereum: a next-generation smart contract and decentralized application platform (2014)
4. Carlsten, M., Kalodner, H., Weinberg, S.M., Narayanan, A.: On the instability of bitcoin without the block reward. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 154–167 (2016)
5. Eyal, I., Sirer, E.G.: Majority is not enough: Bitcoin mining is vulnerable. Commun. ACM **61**(7), 95–102 (2018)
6. Göbel, J., Keeler, H.P., Krzesinski, A.E., Taylor, P.G.: Bitcoin blockchain dynamics: The selfish-mine strategy in the presence of propagation delay. Perform. Eval. **104**, 23–41 (2016)
7. Hillston, J.: A Compositional Approach to Performance Modelling. Cambridge University Press (1996)
8. Kemeny, J.G., Snell, J.L.: Finite Markov Chains. Springer (1976)
9. Lewenberg, Y., Bachrach, Y., Sompolinsky, Y., Zohar, A., Rosenschein, J.S.: Bitcoin mining pools: a cooperative game theoretic analysis. In: Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, pp. 919–927 (2015)
10. Luu, L., Teutsch, J., Kulkarni, R., Saxena, P.: Demystifying incentives in the consensus computer. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 706–719 (2015)
11. Marin, A., Rossi, S.: On the relations between lumpability and reversibility. In: MASCOTS, pp. 427–432. IEEE Computer Society (2014)
12. Marin, A., Rossi, S.: On the relations between Markov chain lumpability and reversibility. Acta Informatica **54**(5), 447–485 (2017)
13. Motlagh, S.G., Mišić, J., Mišić, V.B.: The impact of selfish mining on bitcoin network performance. IEEE Trans. Netw. Sci. Eng. **8**(1), 724–735 (2021)
14. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system. Decentralized Business Review (2008)
15. Rossi, S., Malakhov, I., Marin, A.: Analysis of the confirmation time in proof-of-work blockchains. Futur. Gener. Comput. Syst. **147**, 275–291 (2023)
16. Smuseva, D., Malakhov, I., Marin, A., van Moorsel, A., Rossi, S.: Verifier's dilemma in ethereum blockchain: A quantitative analysis. In: Quantitative Evaluation of Systems: 19th International Conference, QEST 2022, pp. 317–336. Springer (2022)
17. Wright, C.S.: The fallacy of the selfish miner in bitcoin: an economic critique. Available at SSRN: https://ssrn.com/abstract=3151923 (2018)