Selfish Mining in Ethereum

Jianyu Niu, Chen Feng
School of Engineering, *University of British Columbia (Okanagan Campus)*, Canada
jianyu.niu@ubc.ca, chen.feng@ubc.ca

***Abstract*—As the second largest cryptocurrency by market capitalization and today's biggest decentralized platform that runs smart contracts, Ethereum has received much attention from both industry and academia. Nevertheless, there exist very few studies about the security of its mining strategies, especially from the selfish mining perspective. In this paper, we aim to fill this research gap by analyzing selfish mining in Ethereum and understanding its potential threat. First, we introduce a 2-dimensional Markov process to model the behavior of a selfish mining strategy inspired by a Bitcoin mining strategy proposed by Eyal and Sirer. Second, we derive the stationary distribution of our Markov model and compute long-term average mining rewards. This allows us to determine the threshold of computational power that makes selfish mining profitable in Ethereum. We find that this threshold is lower than that in Bitcoin mining (which is 25% as discovered by Eyal and Sirer), suggesting that Ethereum is more vulnerable to selfish mining than Bitcoin.**

***Index Terms*—Bitcoin, Blockchains, Ethereum, Selfish Mining**

I. Introduction

A. Motivation

The Proof-of-Work (PoW) is the most widely adopted consensus algorithm in blockchain platforms such as Bitcoin [1] and Ethereum [2]. By successfully solving math puzzles involving one-way hash functions, the winners of this PoW competition are allowed to generate new blocks that contain as many outstanding transactions as possible (up to the block size limit). As a return, each winner can collect all the transaction fees and earn a block reward (if its new block is accepted by other participants). This economic incentive encourages participants to contribute their computation power as much as possible in solving PoW puzzles (often called mining in the literature).

If all the miners follow the mining protocol, each miner will receive block rewards proportional to its computational power [1]. Interestingly, if a set of colluding miners deviate from the protocol to maximize their own profit, they may obtain a revenue larger than their fair share. Such a behavior is called *selfish mining* in the literature and has been first studied in a seminal paper by Eyal and Sirer in the context of Bitcoin mining [3]. The selfish mining poses a serious threat to any blockchain platform adopting PoW. If colluding miners occupy a majority of the computational power in the system, they can launch a so-called 51% attack to control the entire system.

Selfish mining in Bitcoin has been well studied with various mining strategies proposed (e.g., [4]–[6]) and numerous defenses mechanisms suggested (e.g., [7], [8]). In sharp contrast, selfish mining in Ethereum is much less understood. Ethereum differs from Bitcoin in that it provides the so-called uncle and nephew rewards in addition to the (standard) block rewards used in Bitcoin [9]. This complicates the analysis. As a result, many existing research results on Bitcoin cannot be directly applied to Ethereum.

B. Objective and Contributions

In this paper, we aim to fill this research gap by analyzing selfish mining in Ethereum and understanding its potential threat. We believe, to the best of our knowledge, that we are among the first to develop a mathematical analysis for selfish mining in Ethereum. First, we introduce a 2-dimensional Markov process to model the behavior of a selfish mining strategy inspired by [3]. Second, we derive the stationary distribution of our Markov model and compute long-term average mining rewards. This allows us to determine the threshold of computational power which makes selfish mining profitable in Ethereum. We find that this threshold is lower than that in Bitcoin. In other words, selfish mining poses a more serious threat to Ethereum due to the effect of uncle and nephew rewards. Finally, we perform extensive simulations to verify our mathematical results and obtain some engineering insights.

Note that although our mining strategy is similar to that by Eyal and Sirer [3], our analysis is different from theirs in two aspects. First, our Markov model is 2-dimensional whereas their model is 1-dimensional. Second, our analysis "track" block rewards in a probabilistic way whereas their analysis tracks rewards in a deterministic way. It turns out that our 2-dimensional model and the probabilistic tracking enable us to characterize the effect of uncle and nephew rewards, which is impossible with 1-dimensional model and deterministic tracking.

The main contributions of this paper are summarized as follows:

- To the best of our knowledge, this work is among the first to develop a mathematical analysis of a particular selfish mining strategy in Ethereum. We believe that our approach can be extended to study more advanced selfish mining strategies as well.
- Using our theoretical results, we evaluate the threshold of making selfish mining profitable under different versions of Ethereum proposals with a particular focus on EIP100, which is adopted by the released Byzantium [9]. We find that the threshold is lower than that in Bitcoin, suggesting that Ethereum is more vulnerable to selfish mining than Bitcoin.

Paper Structure. The rest of the paper is structured as follows. Section II provides necessary backgrounds for our work. Section III introduces a selfish mining strategy for Ethereum and Section IV gives the mathematical analysis and results. We conduct a series of simulations to verify our analysis in Section V and discuss how to improve the security of Ethereum in Section VI. After reviewing the related work in Section VII, we conclude the paper in Section VIII.

II. A Primer on Ethereum

Ethereum is a distributed blockchain-based platform that runs smart contracts. Roughly speaking, a smart contract is a set of functions defined in a Turing-complete environment. The users of Ethereum are called clients. A client can issue transactions to create new contracts, to send Ether (internal cryptocurrency of Ethereum) to contracts or to other clients, or to invoke some functions of a contract. The valid transactions are collected into blocks; blocks are chained together through each one containing a cryptographic hash of the previous block.

There is no centralized party to authenticate the blocks and to execute the smart contracts. Instead, a subset of clients (called miners in the literature) verifies the transactions, generates new blocks, and uses the PoW algorithm to drive consensus, receiving Ethers for their effort in maintaining the blockchain network.

A. Blockchain

Each block in the Ethereum blockchain contains three components: a block header, a set of transactions, and some reference links to certain previous blocks called uncle blocks (whose role will be explained in Sec. III-B) [9]. The block header includes a Keccak 256-bit hash value of the previous block, a time stamp and a nonce (which will explained shortly). See Fig. 1 for an illustration in which blocks are linked together by the hash references, forming a chain structure.

Such a chain structure has several desirable features. First, it is tamper free. Any changes of a block will lead to subsequent changes of all later blocks in the chain. Second, it prevents double-spending. All the clients will eventually have the same copy of the blockchain[1] so that any transactions involving double-spending will be detected and discarded.

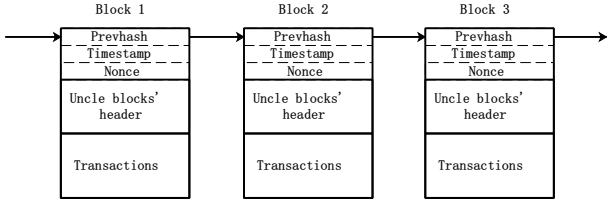


Fig. 1. An illustration of the blockchain structure in Ethereum.

[1]More precisely, all the clients will have a "common prefix" of the blockchain [10].

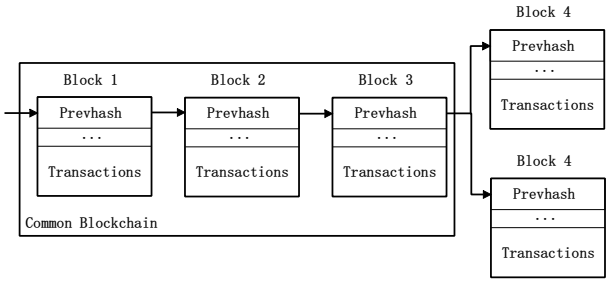


Fig. 2. An illustration of a forked blockchain.

B. PoW and Mining

In order for a miner to produce a new valid block in PoW, he or she needs to find a value of the nonce such that the hash value of the new block is below a certain threshold depending on the difficulty. This puzzle-solving process is often referred to as *mining*. Intuitively, the mining difficulty determines the chance of finding a new block in each try. By adjusting the mining difficulty, the blockchain system can maintain a stable chain growth.

Once a new block is produced, it will be broadcast to the entire network. In the ideal case, a block will arrive all the clients before the next block is produced. If this happens to every block, then each client in the system will have the same chain of blocks. In reality, the above ideal case doesn't always happen. For example, if a miner produces a new block *before* he or she receives the previous block, a fork will occur where two "child" blocks share a common "paren" block. See Fig. 2 for an illustration. In general, each client in the system observes a tree of blocks due to the forking. As a result, each client has to choose a main chain from the tree according to certain rules (e.g., the longest chain rule in Bitcoin and the heaviest subtree rule in the GHOST protocol) [2]. The common prefix of all the main chains is called the *system main chain*.

C. Ethereum Milestones

Ethereum has four milestones: Frontier, Homestead, Metropolis and Serenity. We are now in the third milestone where the mining difficulty level depends not only on the growth of main chains but also on the appearance of uncle blocks, as suggested in EIP100 which is adopted by the released Byzantium [9]. By contrast, the mining difficulty level of Bitcoin only depends on the growth of main chains. Such a difference motivates our work.

III. Selfish Mining on Ethereum

A. Mining Model

In this paper, we consider a system of n miners. The ith miner has m_i fraction of total hash power. Clearly, $\sum_{i=1}^{n} m_i = 1$. We assume miners are either honest (those who follow the protocol) or selfish (those who deviate from the protocol in order to maximize their own profit). Let $\mathcal{S}$ denote the set of

[2]Although Ethereum claimed to apply the heaviest subtree rule [11], it seems to apply the longest chain rule instead [6].

selfish miners and $\mathcal{H}$ denote the set of honest miners. Let α denote the fraction of total hash power controlled by selfish miners and β denote the fraction of total hash power controlled by honest miners . We have $\alpha = \sum_{i\in\mathcal{S}} m_i$ and $\beta = \sum_{i\in\mathcal{H}} m_i$. Clearly, $\alpha + \beta = 1$. Without loss of generality, we assume a single selfish mining pool with α fraction of hash power.

The PoW mining process can be viewed as a series of Bernoulli trails (Sec. II-B), each of which independently finds a valid nonce to complete a block with very low probability. Therefore, the number of trails to mine one block is a geometric random variables. Nowadays, miners in the network are conducting hash computations at a very large rate ($\approx 2.9 \times 10^{14}$ at August 2018). Thus the interval between two successfully mined blocks can be modeled by an exponential random variables. Following [1], [12], [13], we model the mining process of the ith miner as a Poisson process with rate fm_i. Here, f denotes the block mining rate of the entire system (which captures the total hashing power). That is, the ith miner generates new blocks at rate fm_i. Hence, the selfish pool generates blocks at rate $f\alpha$ and the honest miners generate blocks at rate $f\beta$.

B. Mining Rewards

There are three types of block rewards in Ethereum, namely, static block reward, uncle block reward and nephew block reward [2], [14], as outlined in Table I. The static reward is used in both Ethereum and Bitcoin. To explain static reward, we introduce the concepts of *regular* and *stale* blocks. A block is called regular if it is included into the system main chain, and is called stale block otherwise. Each regular block in Ethereum can bring its miner a reward of exactly 3.0 Ethers as an economic incentive.

The uncle and nephew rewards are unique in Ethereum. An uncle block is a stale block that is a "direct child" of the system main chain. In other words, the parent of an uncle block is always a regular block. An uncle block receives certain reward if it is referenced by some future regular block (through the use of reference links) called nephew block. See Fig. 3 for an illustration of uncle and nephew blocks. The values of uncle rewards depend on the "distance" between the uncle and nephew blocks. This distance is well defined because all the blocks form a tree. For instance, in Fig. 3, the distance between uncle block $B3$ (uncle block $D2$, resp.) and its nephew block is 1 (2, resp.). In Ethereum, if the distance is 1, the uncle reward is $\frac{7}{8}$ of the (static) block reward; if the distance is 2, the uncle reward is $\frac{6}{8}$ of the block reward and so on. Once the distance is greater than 6, the uncle reward will be zero. By contrast, the nephew reward is always $\frac{1}{32}$ of the block reward. Except blocks rewards, miners can also receive gas cost as a reward for verifying and executing all the transactions [2]. However, gas cost is dwarf with other rewards, we ignore it in our analysis.

In our later analysis, we use K_s, K_u, and K_n to denote static, uncle, and nephew rewards, respectively. Without loss of generality, we assume that $K_s = 1$ so that K_u (K_n, resp.) represents the ratio of uncle reward (nephew reward, resp.) to the block reward. As we explained before, in the current version of Ethereum, $K_n < K_u < 1$ and K_u is a function of the distance. As we will see later, our analysis allows K_u and K_n to be any functions of the distance.

TABLE I
MINING REWARDS IN ETHEREUM AND BITCOIN

	Ethereum	Bitcoin	Purpose
Static Reward	✓	✓	Compensate for miners' mining cost
Uncle Reward	✓	×	Reduce centralization trend of mining
Nephew Reward	✓	×	Encourage miners to reference uncle blocks
Transaction Fee (Gas Cost)	✓	✓	Transaction execution; Resist network attack

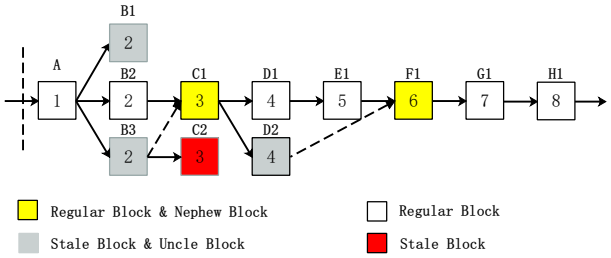


Fig. 3. Different block types in Ethereum. The block sets $\{A, B2, C1, D1, E1, F1, G1, H1\}$ and $\{B1, B3, C2, D2\}$ belong to regular blocks and stale blocks, respectively. Specially, the block sets $\{C1, F1\}$ and $\{B1, B3, D2\}$ also belong to nephew blocks and uncle blocks. Uncle block B3 (uncle block D2, resp.) is referenced with distance one (two, resp.).

C. Mining Strategy

We now describe the mining strategies for honest and selfish miners. The honest miners follow the protocol given in Sec. II-B. Each honest miner observes a tree of blocks. It chooses a main chain from the tree and mines new blocks on its main chain. Once a new block is produced, the miner broadcasts the block to everyone in the system. Also, it includes as many reference links as possible to (unreferenced) uncle blocks in the tree.

By contrast, the selfish pool can withhold its newly mined blocks and publish them strategically to maximize its revenue. The basic idea behind selfish mining is to increase the selfish pool's share of regular blocks and, at the same time, to gain as many uncle and nephew rewards as possible. Specifically, the selfish pool keeps its newly discovered blocks private, creating a fork on purpose. The pool then continues to mine on this private branch, while honest miners still mine on public branches (which are often shorter than the private branch).

Fig. 4 gives an example in which "circle" blocks are mined by the pool and "square" blocks are mined by honest miners. In this example, the private branch consists of 4 blocks $(D1, E1, F, G)$, all of which are mined by the pool with $(D1, E1)$ published and (F, G) still private. There are two public branches, namely, $(D1, E1)$ and $(D2, E2)$, because

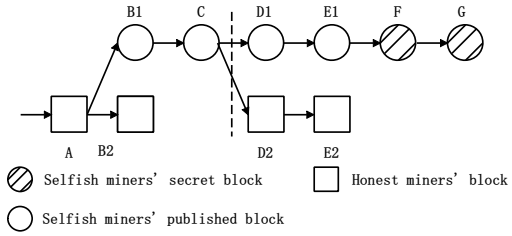


Fig. 4. An example to illustrate private branch length and public branch length

honest miners can see both branches. Each honest miner then chooses one public branch to mine new blocks according to certain rules (e.g., the longest chain rule).

Let $L_s(t)$ be the length of the private branch seen by the selfish pool at time t. Similarly, let $L_h(t)$ be the length of public branches seen by honest miners at time t. (Note that $L_h(t)$ is well defined because all public branches have the same length under our selfish mining strategy.) We are now ready to describe our selfish mining strategy which is based on the strategy in [3].

Algorithm 1 An selfish Mining Strategy in Ethereum

on The selfish pool mines a new block
1: reference all (unreferenced) uncle blocks based on its private branch
2: $L_s \leftarrow L_s + 1$
3: **if** $(L_s, L_h) = (2, 1)$ **then**
4: publish its private branch
5: $(L_s, L_h) \leftarrow (0, 0)$ (since all the miners achieve a consensus)
6: **else**
7: keep mining on its private branch

on Some honest miners mine a new block
8: The miner references all (unreferenced) uncle blocks based on its public branches
9: $L_h \leftarrow L_h + 1$
10: **if** $L_s < L_h$ **then**
11: $(L_s, L_h) \leftarrow (0, 0)$
12: keep mining on this new block
13: **else if** $L_s = L_h$ **then**
14: publish the last block of the private branch
15: **else if** $L_s = L_h + 1$ **then**
16: publish its private branch
17: $(L_s, L_h) \leftarrow (0, 0)$ (since all the miners achieve a consensus)
18: **else**
19: publish first unpublished block in its private branch
20: set $(L_s, L_h) = (L_s - L_h + 1, 1)$ if the new block is mined on a public branch that is a prefix of the private branch

Algorithm 1 presents the strategy. When the selfish pool mines a new block (see lines 1 to 7), it will keep this block private and continue mining on its private branch, unless its advantage is limited (i.e., $(L_s, L_h) = (2, 1)$) which will be discussed later.

When some honest miners mine a new block, the length of some public branch will be increased by 1. We have the following cases. Case 1) If the new public branch is longer than the private branch, the pool will adopt the public branch and mine on it. (That is why the pool will set $(L_s, L_h) = (0, 0)$.) Case 2) If the new public branch has the same length as the private branch, the pool will publish its private block immediately (so that the honest miners will see two branches of the same length) hoping that as many honest miners will choose its private branch as possible. Case 3) If the new public branch is shorter than the private branch by just 1, the pool will publish its private branch so that all the honest miners will adopt the private branch. Case 4) If the new public branch is shorter than the private branch by at least 2, the pool will publish the first unpublished block since the pool still have a clear advantage. Moreover, if the new block is mined on a public branch that is a prefix of the private branch, the pool will set $(L_s, L_h) = (L_s - L_h + 1, 1)$ due to a new forking (caused by the honest miner).

To better illustrate the selfish mining strategy, we provide an example in Fig. 5. In Step 1, we have $(L_s, L_h) = (3, 0)$. In Step 2, some honest miners publish block $A2$ and we have $(L_s, L_h) = (3, 1)$. This corresponds to Case 4). Hence, the pool immediately publishes block $A1$, still having an advantage of 2 blocks. In Step 3, some honest miners publish block $B2$, leading to $(L_s, L_h) = (3, 2)$. This corresponds to Case 3). Thus, the pool publishes its private branch, making honest miners' blocks ($A2$ and $B2$) become stale.

Remark 1. *The selfish mining strategy presented above isn't optimal. But by exploring it, we can reveal some characteristics of the selfish mining in Ethereum.*

D. Mining Pool

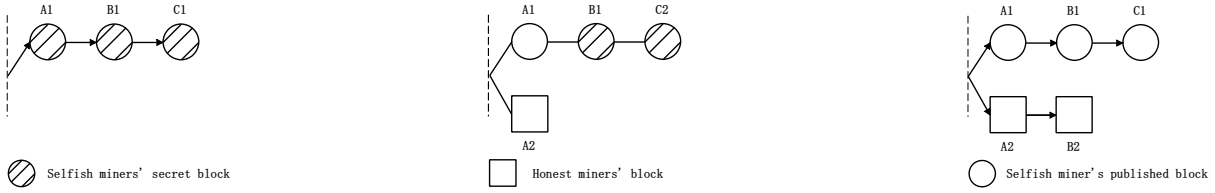
In Ethereum, individual miners can form mining pools to mine blocks together and share the revenue according to individuals' hash power. Fig. 6 presents the fractions of the hash power of various mining pools in Ethereum [15]. The largest mining pool (called Ethermine) has dominated 26.34% of the total hash power. The top two mining pools have dominated 48.8% of the total hash power. The top five mining pools have more than 81% of the total hash power. Note that these mining pools are not necessarily selfish. Due to the presence of these pools, it is important to understand the impact of selfish mining in Ethereum.

IV. Analysis of Selfish Mining

In this section, we will study the long-term behavior of the selfish mining strategy using a Markov model with a particular focus on the mining revenue.

A. Network Model

To simplify our analysis, we follow the network model of [3], [4], [6] in which the time it takes to broadcast a block is



a) Step 1: selfish pool withholds 3 blocks b) Step 2: selfish pool publishes 1 block b) Step 3: selfish pool overrides 2 blocks

Fig. 5. A simple example of the mining strategy.

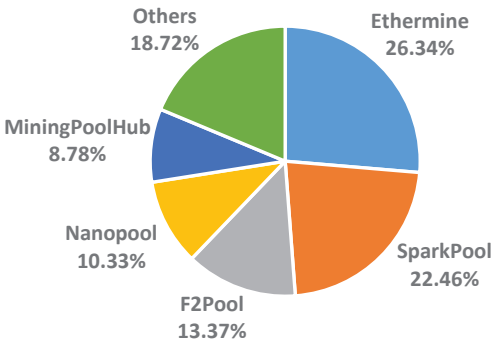


Fig. 6. The top 5 mining pools' hash power in Ethereum (2018.09)

negligible. In particular, we introduce the same parameter γ as in [3], which denotes the ratio of honest miners that are mining on blocks produced by the selfish pool (rather than by the honest miners) whenever they observe a fork of two branches of equal length (because the pool publishes some unpublished blocks in its private branch). Recall that the selfish pool has to publish its unpublished block (if any) immediately after some honest miners produce a new block. So, the pool has to listen to the network all the time and publish its block as soon as possible when some honest miners publish blocks. In this sense, the parameter γ captures the pool's communication capability.

If the honest miners apply the uniform tie-breaking rule when they observe a fork of two branches of equal length, then $\gamma = \frac{1}{2}$. On the other hand, if the pool can launch a network attack to influence honest miners' block propagation, then only a few honest miners can see any new block produced by some honest miner. In this case, the parameter γ is close to 1. In this paper, we assume that γ takes values in the interval $[0, 1]$.

B. Markov Process

For ease of presentation, we re-scale the time axis so that the honest miners generate new blocks at rate α and the selfish pool generates new blocks at rate β. We are now ready to define the system state. Recall that $L_s(t)$ is the length of the private branch and $L_h(t)$ is the length of the public branches at time t. Clearly, $(L_s(t), L_h(t))$ captures the system state at time t. The state space $\mathcal{S}$ contains the following states: $(0,0)$, $(1,0)$, $(1,1)$, as well as (i,j) with $i - j \geq 2$ and $j \geq 0$. It is easy to verify that $(L_s(t), L_h(t))$ evolves as a Markov process according to our selfish mining strategy and the network model, as illustrated in Fig. 7. Moreover, we can show that the process $(L_s(t), L_h(t))$ is positive recurrent and so it has a unique stationary distribution.

C. The Stationary Distribution

To compute the stationary distribution of the process $(L_s(t), L_h(t))$, we need to derive the transition rates for the state evolution as follows.

- $q_{(0,0),(0,0)} = \beta$
  This transition happens if any honest miner produces a new block, broadcasts it to everyone. Then, the selfish pool adopts the public branch and mines on it. Thus, the rate is β.
- $q_{(0,0),(1,0)} = \alpha$
  This transition happens if the pool produces a new block and keeps it private. Thus, the rate is α.
- $q_{(1,0),(2,0)} = \alpha$
  This transition happens if the pool produces a new block and keeps it private. Thus, the rate is α.
- $q_{(1,0),(1,1)} = \beta$
  This transition happens if any honest miner produces a new block and the pool immediately publishes its private block (because the new public branch has the same length as the private branch). Thus, the rate is β.
- $q_{(1,1),(0,0)} = \alpha + \beta = 1$
  This transition happens if any of the following events happens. 1) The pool produces a new block and publishes its private branch (because $(L_s, L_h) = (2, 1)$); 2) Any honest miner produces a new block and the pool has to publish its private branch (because $L_s = L_h + 1$). Thus, the rate is $\alpha + \beta = 1$.
- $q_{(i,j),(i+1,j)} = \alpha$ for $i \geq 2$ and $j \geq 0$
  This transition happens if the selfish pool produces a new block and keeps it private. Thus, the rate is α.
- $q_{(i,j),(i-j,1)} = \beta\gamma$ for $i - j \geq 3$ and $j \geq 1$
  This transition happens if any honest miner mines a new block on a public branch which is a prefix of the private branch. Then, the pool publishes a private block accordingly. Thus, the rate is $\beta\gamma$.
- $q_{(i,j),(0,0)} = \beta$ for $i - j = 2$ and $j \geq 1$
  This transition happens if any honest miner produces a new block and then the pool publishes its private branch (because $L_s = L_h + 1$). Thus, the rate is β.
- $q_{(2,0),(0,0)} = \beta$
  This transition happens if any honest miner produces a

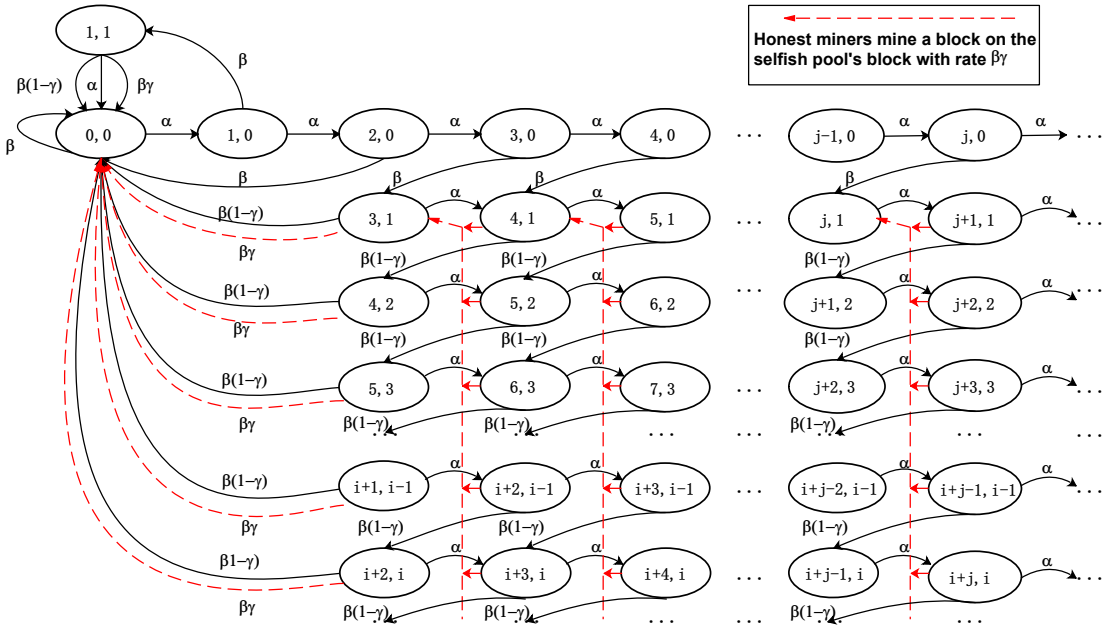


Fig. 7. The Markov process of the selfish mining in Ethereum.

new block and then the pool publishes its private branch (because $L_s = L_h + 1$). Thus, the rate is β.

- $q_{(i,0),(i,1)} = \beta$ for $i \geq 3$
  This transition happens if any honest miner produces a new block and the pool publishes a private block. Thus, the rate is β.
- $q_{(i,j),(i,j+1)} = \beta(1-\gamma)$ for $i - j \geq 3$ and $j \geq 1$
  This transition happens if any honest miner mines a new block on a public branch which is not a prefix of the private branch. Thus, the rate is $\beta(1-\gamma)$.

Let $\{\pi_{i,j}\}$ be the steady-state distribution of the Markov process $(L_s(t), L_h(t))$. Then, the set $\{\pi_{i,j}\}$ satisfies the following global balance equations according to the above transition rates.

$$\begin{cases} \alpha\pi_{0,0} = \pi_{1,1} + \beta\sum_{j=0}^{\infty}\pi_{2+j,j}, \\ \pi_{1,1} = \beta\pi_{1,0}, \\ \pi_{3,1} = \beta\pi_{3,0} + \sum_{j=1}^{\infty}\beta\gamma\pi_{3+j,j}, \\ \pi_{i,0} = \alpha\pi_{i-1,0}, \text{for } i \geq 1, \\ \pi_{i,1} = \beta\pi_{i,0} + \alpha\pi_{i-1,1} + \sum_{j=1}^{\infty}\beta\gamma\pi_{i+j,j}, \text{for } i \geq 4, \\ \pi_{i,i-2} = \beta(1-\gamma)\pi_{i,i-1}, \text{for } i \geq 4, \\ \pi_{i,j} = \alpha\pi_{i-1,j} + \beta(1-\gamma)\pi_{i,j-1}, \text{for } j \geq 2, i \geq 5. \end{cases} \tag{1}$$

Solving the global balance equations, we obtain the following results for the stationary distribution:

$$\begin{aligned} \pi_{0,0} &= \frac{1-2\alpha}{2\alpha^3 - 4\alpha^2 + 1}, \\ \pi_{i,0} &= \alpha^i\pi_{0,0} \text{ for } i \geq 1, \\ \pi_{1,1} &= \left(\alpha - \alpha^2\right)\pi_{0,0}, \\ \pi_{i,j} &= \alpha^i(1-\alpha)^j(1-\gamma)^j f(i,j,j)\pi_{0,0} + \\ &\quad \alpha^{i-j}\gamma(1-\gamma)^{j-1}\left(\frac{1}{(1-\alpha)^{i-j-1}} - 1\right)\pi_{0,0} - \\ &\quad \gamma(1-\gamma)^{j-1}\sum_{k=1}^{j}\alpha^{i-k}(1-\alpha)^{j-k}f(i,j,j-k)\pi_{0,0} \end{aligned}$$

for $i \geq j+2$ and $j \geq 1$, where the function $f(x,y,z)$ is defined as

$$f(x,y,z) = \begin{cases} \underbrace{\sum_{s_z=y+2}^{x}\sum_{s_{z-1}=y+1}^{s_z}\cdots\sum_{s_1=y-z+3}^{s_2}}_{z} 1, & z \geq 1, x \geq y+2, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

The function $f(x,y,z)$ is a multiple summations. See Appendix A for a detailed introduction.

We have the following remarks for the stationary distribution in Equation 2.

Remark 2. *When $0 < \alpha < \frac{1}{2}$, we have $0 < \pi_{0,0} < 1$. Specially, the distribution $\pi_{0,0}$ only depends on the parameter α and is monotonically decreasing. The tendency of $\pi_{0,0}$ reflects that with more hash power, the selfish pool can obtain more lead blocks and stay in state $(0,0)$ less frequently.*

Remark 3. *The distributions $\pi_{i,0}$ with $i \geq 1$ are decreasing geometrically and are less than 10^{-6} when $i \geq 15$ given $\alpha =$*

0.4. *Thus, in later analysis truncating the states won't make the analysis lose too much accuracy, but can greatly simplify the computation.*

D. Reward Analysis

In this subsection, we mainly provide some key insights of our reward analysis. With the insights, it's easy to figure out the transition reward for each state, and thereafter to compute the revenues for the selfish pool and honest miners. For a better presentation flow, we leave the explicit analysis in Appendix B.

Challenge Outline Compared with Bitcoin, the biggest difference of selfish mining issue in Ethereum is that we have to take uncle and nephew rewards into account. However, it's a non-trivial challenge to the existed approaches [3], [5] in Bitcoin. This is because, on the one side, the existed one-dimensional Markov model can't provide enough state information to tracking the uncle and nephew rewards. On the other side, recall Sec. III-B, in order to compute the uncle and nephew rewards won by the selfish pool and honest miners, we have to distinguish which blocks are uncle blocks and nephew blocks, tell which miners the corresponding blocks belong to, record how many uncle blocks haven't been referenced, and know each referencing distance between the uncle block and associated nephew block. So it's obvious just directly extending the state space of Markov model in [3], [5] will make the whole analysis complicated and hard to be solved.

Proof Outline For the sake of scalability and simplicity, our analysis provides a new perspective to deal with the various rewards of selfish mining. Specifically speaking, we "track" the reward received by each new block associated with a state transition whereas [3] tracks published block(s) associated with a state transition. However, it's unintuitive to see the benefits from the transformation. Usually, we are more likely to decide the rewards when blocks are published and their corresponding type are determined. But as said previously, in this method we need to track all the information of blocks, which will make the analysis too complicated to be solved. On the contrary, if we can decide the reward when blocks are generated based on probability, which will greatly reduce the amount of tracked information. Thus in our previous model, although we need to deal with more complicated uncle and nephew rewards, we can use $(L_s(t), L_h(t))$ to captures the selfish mining process at arbitrary time t. Owning the simplicity, our analysis can deal with more optimal and complicated mining strategy.

Here, we provide a simple example to better understand the differences between the two perspectives. We assume the selfish pool has already mined two blocks and kept them privately at time t. Then some honest miners mine a new block. By the mining Algorithm 1, the pool publishes its private branch at once. It's easy to see the block mined by honest miners is uncle block. But we still can't figure out which nephew block will reference it and what' the referencing block distance (used to compute the uncle reward). Using the previous approach, we need to record the information of the newly mined uncle block and assign its miner reward when it's referenced. However, see the *Case 7* in Appendix B, using our approach we can decided the uncle block will be referenced with block distance two and the nephew reward will be received by honest miners with probability $\beta(1+\alpha\beta(1-\gamma))$ and by the pool with probability $1-\beta(1+\alpha\beta(1-\gamma))$ without storing the massive information.

E. Revenue Analysis

In Appendix B, we provide the various block rewards for every state transitions. In this subsection, we combine them together to compute various rewards received by the selfish pool and honest miners. This calculation is straightforward.

1) Revenue Computing: First, we compute the block rewards for the selfish pool (denoted as r_b^s) and honest miners (denoted as r_b^h). We have the following results:

$$\begin{aligned} r_b^s &= \big(\alpha(1-\pi_{0,0}) + (\alpha^2+\alpha^2\beta+\alpha\beta^2\gamma)\pi_{0,0}\big) \\ &= \alpha - \alpha\beta^2(1-\gamma)\pi_{0,0}, \\ &= \frac{\alpha(1-\alpha)^2(4\alpha+\gamma(1-2\alpha))-\alpha^3}{2\alpha^3-4\alpha^2+1} \end{aligned} \tag{3}$$

and

$$\begin{aligned} r_b^h &= \beta(\pi_{0,0}+\pi_{1,1}) + \beta^2(1-\gamma)\pi_{1,0} \\ &= \frac{(1-2\alpha)(1-\alpha)(\alpha(1-\alpha)(2-\gamma)+1)}{2\alpha^3-4\alpha^2+1} \end{aligned} \tag{4}$$

Note that r_b^s and r_b^h represent the long-term average static rewards per time unit. Since all the miners generate new blocks at rate 1, the maximum long-term average reward is 1 per time unit. Hence, we have $r_b^s + r_b^h \leq 1$.

Remark 4. *If we only consider static rewards, the above results are the same as those in [3], though our approach is different from that in [3].*

Next, we can compute the uncle block rewards for the selfish pool (denoted as r_u^s):

$$\begin{aligned} r_u^s &= \alpha\beta^2(1-\gamma)K_u(1)\pi_{0,0} \\ &= \frac{(1-2\alpha)(1-\alpha)^2\alpha(1-\gamma)}{2\alpha^3-4\alpha^2+1}K_u(1) \end{aligned} \tag{5}$$

Remark 5. *Note that r_u^s is zero in Bitcoin. In other word, the involved selfish pools' blocks without rewards can be viewed as the cost of launching selfish mining attack. Thus the additional reward in Ethereum will reduce the cost of selfish mining and make it easier. Moreover, the involved uncle blocks of the pool are always referenced with distance one, the minimum distance means the selfish pool can obtain the maximum uncle reward for its uncle blocks.*

Similarly, we can compute the uncle block rewards for the honest miners (denoted as r_u^h):

$$\begin{aligned} r_u^h = (\alpha\beta+\beta^2\gamma)K_u(1)\pi_{1,0} + \sum_{i=2}^{\infty}\beta K_u(i)\pi_{i,0} + \\ + \sum_{i=2}^{\infty}\sum_{j=1}^{\infty}\beta\gamma K_u(i)\pi_{i+j,j}. \end{aligned} \tag{6}$$

Remark 6. *In the current version of Ethereum, the function $K_u(\cdot)$ is given below:*

$$K_u(l) = \begin{cases} (8-l)/8, & 1 \le l \le 6 \\ 0, & \textit{otherwise}. \end{cases} \quad (7)$$

Our analysis applies to an arbitrary function of $K_u(\cdot)$.

Then, we can compute the nephew block rewards for the selfish pool (denoted as r_n^s) and honest miners (denoted as r_n^h):

$$r_n^s = \alpha\beta K_s(1)\pi_{1,0} + \sum_{i=2}^{\infty}\sum_{j=1}^{\infty} \beta^{i-1}\gamma(\alpha - \alpha\beta^2(1-\gamma))K_s(i)\pi_{i+j,j} \quad (8)$$

$$r_n^h = \alpha\beta^2(1-\gamma)K_s(1)\pi_{0,0} + \beta^2\gamma K_s(1)\pi_{1,0} + + \sum_{i=2}^{\infty}\sum_{j=1}^{\infty} \beta^{i}\gamma(1+\alpha\beta(1-\gamma))K_s(i)\pi_{i+j,j} \quad (9)$$

Remark 7. *In the current version of Ethereum, the function $K_n(\cdot)$ is always equal to $\frac{1}{32}$. Our analysis applies to an arbitrary function of $K_n(\cdot)$.*

Finally, we can obtain the total mining revenue r_{total} as

$$r_{\text{total}} = r_b^s + r_b^h + r_u^s + r_u^h + r_n^s + r_n^h. \quad (10)$$

Hence,

$$R_s = \frac{r_b^s + r_u^s + r_n^s}{r_{\text{total}}}$$

gives the share of the mining revenue by the selfish pool.

2) Absolute Revenue: We now define the absolute revenue U_s for the selfish pool. As we will soon see, although the absolute revenue is equivalent to the relative revenue (i.e., the share R_s) in Bitcoin, it is different from the relative revenue in Ethereum due to the presence of uncle and nephew rewards.

Recall that Bitcoin adjusts the mining difficulty level so that the regular blocks are generated at a stable rate, say 1 block per time unit. Thus, the long-term average total revenue is fixed to be 1 block reward per time unit with or without selfish mining. This makes the absolution revenue equivalent to the relative revenue. The situation is different in Ethereum. Even if the regular blocks are generated at a stable rate, the average total revenue still depends on the generation rate of uncle blocks, which is affected by selfish mining as we will see shortly. Indeed, Ethereum didn't take into account the generation rate of uncle blocks when adjusting the difficulty level until its third milestone. This motivates us to consider two scenarios in our analysis: 1) the regular block generation rate is 1 block per time unit, and 2) the regular and uncle block generation rate is 1 block per time unit.

In our previous analysis, the regular block generation rate is $r_b^s + r_b^h$, which is smaller than 1 as explained before. Thus, we can re-scale the time to make the regular block generation rate to be 1 block per time unit. In this scenario, the long-term absolute revenue for the selfish pool is

$$U_s = \frac{r_b^s + r_u^s + r_n^s}{r_b^s + r_b^h}, \quad (11)$$

and the long-term absolute revenue for honest miners is

$$U_h = \frac{r_b^h + r_u^h + r_n^h}{r_b^s + r_b^h}. \quad (12)$$

Similarly, we can re-scale the time to make the regular and uncle block generation rate to be 1 block per time unit and define long-term absolute revenues for the selfish pool and honest miners accordingly.

3) Threshold Analysis: First of all, if the selfish pool follows the mining protocol, its long-term average absolute revenue will be α, since the network delay is negligible (and so no stale blocks will occur). On the other hand, if the pool applies the selfish mining strategy proposed in this paper, its long-term absolute revenue is given by U_s.

Let α^* be the smallest value such that $U_s \ge \alpha$. Clearly, α^* is the threshold of computational power that makes selfish mining profitable in Ethereum. We can determine α^* for both scenarios through numerical calculations.

V. Evaluation

In this section, we build an Ethereum selfish mining simulator to validate our theoretical analysis. In particular, we simulate a system with $n = 1000$ miners, each with the same block generation rate. In our simulations, the selfish pool controls at most 450 miners (i.e., $\alpha \le 0.45$) and runs the Algorithm 1. In contrast, the honest miners follow the designed protocol (Sec. III-C). We consider the block propagation delay by introducing the stale rate δ in our simulation [6], [16]. Our simulation results are based on the average of 10 runs, where each run generates $100,000$ blocks.

A. Validation of the Theory Results

In this subsection, we validate the long-term average absolute revenues for the selfish pool and honest miners. Fig. 8 plots the results obtained from analysis and simulations. From the results, we can see when $\gamma = 0.5$, $K_u = 4/8K_s$ and α changes from 0 to 0.45, the simulation results match our theoretical results[3]. In addition, when α is above 0.163, the selfish pool can always gain higher revenue from selfish mining than following honest mining. More importantly, when α is below the threshold 0.163, the selfish pool lose just a small amount of revenue due to the additional uncle block rewards, which is quite different with the results in Bitcoin [3].

B. Impact of the Uncle Reward

In this subsection, we explore the impact of the uncle block rewards on the selfish pool's and honest miners' revenues. To this end, we first use the uncle reward function $K_u(\cdot)$ (Sec. IV-E) in Ethereum, then set the uncle reward as a fixed value regardless of the distance, ranging from $2/8K_s$ to $7/8K_s$. Here the fixed uncle reward value can directly show its impacts and simplify our understanding.

[3]To simplify the numerical calculations of our results, we only consider the states (i, j) with i and j less than 200. This approximation turns out to be accurate when $\alpha \le 0.45$.

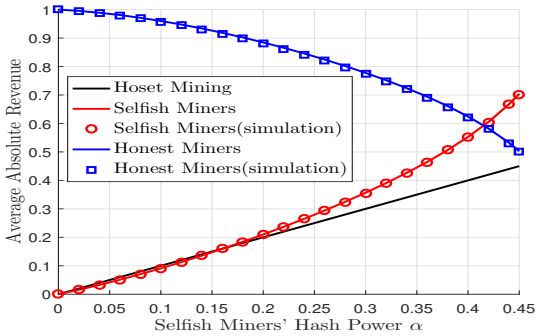


Fig. 8. Revenue rate for the selfish pool and honest miners when $\gamma = 0.5$, $K_u = 4/8K_s$ and α changes from 0 to 0.45.

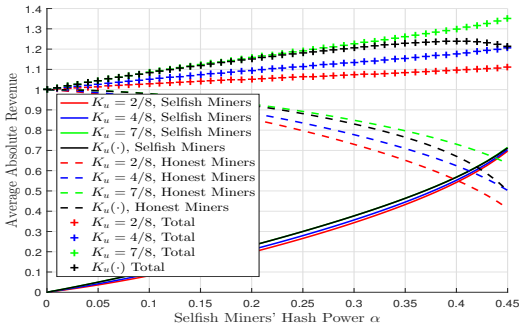


Fig. 9. Revenue rate for the selfish pool, honest miners under different uncle block reward K_u.

Fig. 9 shows that the higher uncle reward, the more absolute revenue for both the selfish pool and honest miners, which is quite obvious. It also reveals that the total revenue increases with the selfish pool's computation power α and soars to 135% of the revenue without selfish mining, when $K_u = 7/8K_s$ and $\alpha = 0.45$. This is because, without the consideration of uncle blocks into difficulty adjust, the selfish mining can produce additional uncle and nephew rewards, resulting in the fluctuation of total revenue. Additionally, the uncle reward function $K_u(\cdot)$ used in Ethereum has the same effect with the setting $K_u = 7/8K_s$ for selfish pool's revenue (explained in Sec. IV-E). Whereas, $K_u(\cdot)$ functions complicatedly for the honest miners' revenue. When α is small, its impact is similar with $K_u = 7/8K_s$, and when α is close to 0.45, its impact is similar with $K_u = 4/8K_s$. This is because with the increase of α, the average referencing distances of honest miners' uncle blocks will increase, which further leads to the decrease of honest miners' average uncle rewards when using function $K_u(\cdot)$. That finding motivates our discussion in Sec. VI.

C. Comparison with Bitcoin

In this subsection, we compare the hash power thresholds of making selfish mining profitable in Ethereum and Bitcoin under different values of γ. In Ethereum, We use function $K_u(\cdot)$ to compute the uncle rewards. Particularly, we compute the thresholds for the two scenarios (recall Sec. IV-E2): 1) the regular block generation rate is 1 block per time unit, and 2) the regular and uncle block generation rate is 1 block per time unit.

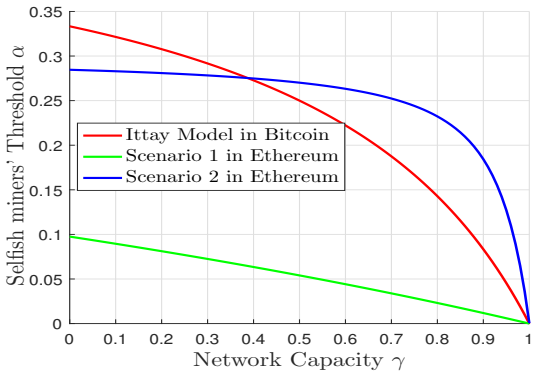


Fig. 10. The profitable threshold of hash power in Bitcoin and Ethereum.

From Fig. 10, we can see that the higher γ is, the lower hash power needed for a profitable selfish mining. Specially, when $\gamma = 1$, the selfish mining in Bitcoin and Ethereum can always be profitable regardless of their hash power. Besides that, the results show that the hash power thresholds of Ethereum in scenario 1 are always lower than Bitcoin. Whereas, the hash power thresholds in scenario 2 are higher than Bitcoin when $\gamma \geq 0.39$. This is because the larger γ is, the more blocks mined by honest miners are uncle blocks. However, in scenario 2 the additional referenced uncle blocks will reduce the generation rate of regular block, resulting in the decrease of selfish pools' block rewards. Thus the selfish pool needs to have higher hash power to make selfish mining profitable.

D. Honest miners' Discounted Hash Power

In reality, new blocks can't immediately arrive to every miners in the network, which means miners cannot always mine on the top of the latest public branch [16]. In order to evaluate the impact, we introduce a parameter δ as in [6], [16] to denote the fraction of honest miners' hashing power that mines on a stale block rather than the latest public branch. Note that the parameter δ allows us to account for the impact of the network propagation delay (which is missing in our analysis). We assume that the selfish pool doesn't suffer from the propagation delay because of its powerful networking ability. Fig. 11 compares the selfish pool' revenue under different value of δ. It's clear that the higher δ is, the more honest miners' hash power is wasted and the lower threshold of making selfish mining profitable is needed.

VI. Discussion

In Ethereum, uncle and nephew rewards are initially designed to solve the mining centralization bias—miners form or join in some big mining pools (Sec. III-D). This is because due to propagation delay [17] individually mining with less hash power is more likely to generated stale blocks and win less block reward. However, as analyzed previously, the uncle reward (computing by function $K_u(\cdot)$) can greatly reduce the

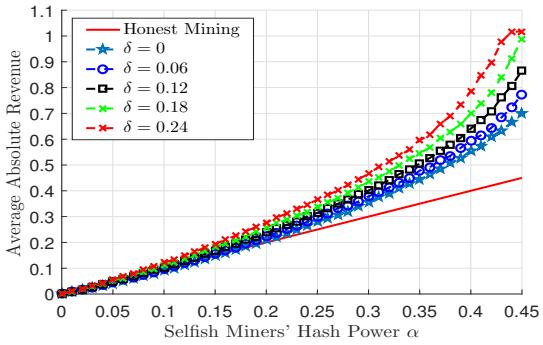


Fig. 11. The revenue rate for the selfish pool under different stale rate δ.

TABLE II
THE DISTRIBUTION OF HONEST MINERS' UNCLE BLOCK WITH DIFFERENT REFERENCING BLOCK DISTANCES

Referencing distance	$\alpha = 0.3$	$\alpha = 0.45$
1	0.5270	0.2841
2	0.2954	0.2485
3	0.1109	0.1709
4	0.0429	0.1252
5	0.0170	0.0958
6	0.0068	0.0755

loss of launching selfish mining. Here we propose a new uncle reward computing function to balance the two aspects.

One insight from our analysis is that the uncle blocks mined by the selfish pool can always be reference with block distance one, i.e, the maximum uncle reward $7/8K_s$, while the honest miners' uncle blocks are referenced with longer average block distance (see analysis in Sec. IV-E1). Here we give the distribution of the honest miners' uncle block with different referencing block distance in Table II when $\gamma = 0.5$. As we can see, with the increase of α, the average referencing distance of honest miners' blocks are increasing. What's more, it's obvious the existed uncle reward function $K_u(\cdot)$) are more beneficial for the selfish pool. Thus we propose to set the uncle reward function $K_u(\cdot)$) as uniform distribution to mitigate the selfish pool's advantage, i.e, the uncle block reward is fixed at $4/7K_s$ when its referencing block distance is between 1 and 6. From our previous results, when $\gamma = 0.5$, we can see the threshold of selfish mining promote from 0.06 to 0.163 in scenario 1, and from to in scenario 2.

VII. Related Work

The research of selfish mining is mostly focused on Bitcoin with roughly two directions: 1) optimizing the selfish mining strategies in order to increase the revenue and lower the threshold of launching selfish mining attacks; 2) proposing defense mechanisms. In [3], Eyal and Sirer developed a Markov process to model the Selfish-Mine Strategy and to evaluate the selfish pool's relative revenue. Moreover, they proposed a uniform tie-breaking defense against selfish mining, which is adopted in Ethereum. Inspired by this seminal paper, Sapirshtein et al. [4] and Nayak et al. [5] demonstrated that by adopting the optimized strategies, the threshold of the hashing power to make selfish mining profitable can be reduced to 23.2% even when honest miners adopt the uniform tie-breaking defense. Furthermore, the authors in [18] took the propagation delay into the analysis of selfish mining.

As for defense mechanisms,Heilman proposed a defense mechanism called Freshness Preferred [8], in which by using the latest unforgeable timestamp issued by a trusted party, the threshold can be increased to 32%. Bahack in [19] introduced a fork-punishment rule to make selfish mining unprofitable. Specially, each miner in the system can include a fork evidence in their block. Once confirmed, the miner can get half of the total rewards of the wining branch. Solat and Potop-Butucaru [20] propose a solution called ZeroBlock, which can make selfish miners' block expire and be rejected by all the honest miners without using forgeable timestamps. In [7], the authors proposed a backward-compatible defense mechanism called weighted FRP which considers the weights of the forked chains instead of their lengths. This is similar in spirit to the GHOST protocol [11].

However, there exists very few studies about the selfish mining in Ethereum. The author in [21] first proposed to exploit the flaw of difficulty adjustment to mine additional uncle blocks, which is shown less profitable than our selfish mining strategy. In [16] the authors built a Monte Carlo simulation platform to quantify the security of the Ethereum after EIP100. However, their paper contains no mathematical analysis and can't directly observe the effects of uncle block rewards and nephew rewards.

VIII. Conclusion And Future Work

In this paper, we have proposed a Markov model to analyze a selfish mining strategy in Ethereum. Our model enables us to evaluate the impact of the uncle and nephew rewards, which is missing in the previous analysis for selfish mining in Bitcoin. In particular, we have shown how these rewards influence the security of Ethereum mining. Additionally, we have computed the hashing power threshold of making selfish mining profitable, which is essential for us to evaluate the security of Ethereum mining.

As a major finding, we notice that it is important to consider uncle blocks when adjusting the mining difficulty level. Otherwise, Ethereum would be much more vulnerable than Bitcoin to selfish mining. This finding supports the emendation adopted by the third milestone of Ethereum. However, once the mining mechanism is changed, the selfish pool is likely to adopt some new mining strategies. We leave the exploration of the optimal selfish mining strategy using the proposed approach as future work.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Working Paper*, 2008.

[2] V. Buterin *et al.*, "A next-generation smart contract and decentralized application platform," *white paper*, 2014. [Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper
[3] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," *Commun. ACM*, vol. 61, no. 7, pp. 95–102, Jun. 2018.
[4] A. Sapirshtein, Y. Sompolinsky, and A. Zohar, "Optimal selfish mining strategies in bitcoin," in *International Conference on Financial Cryptography and Data Security*. Berlin, Heidelberg: Springer, 2016, pp. 515–532.
[5] K. Nayak, S. Kumar, A. Miller, and E. Shi, "Stubborn mining: Generalizing selfish mining and combining with an eclipse attack," in *2016 IEEE European Symposium on Security and Privacy (EuroS P)*. Saarbrucken, Germany: IEEE, March 2016, pp. 305–320.
[6] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. Vienna, Austria: ACM, 2016, pp. 3–16.
[7] R. Zhang and B. Preneel, "Publish or perish: A backward-compatible defense against selfish mining in bitcoin," in *Cryptographers' Track at the RSA Conference*. Cham: Springer, 2017, pp. 277–292.
[8] E. Heilman, "One weird trick to stop selfish miners: Fresh bitcoins, a solution for the honest miner," in *International Conference on Financial Cryptography and Data Security*. Berlin, Heidelberg: Springer, 2014, pp. 161–162.
[9] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger byzantium version," *Ethereum project yellow paper*, pp. 1–32, 2018. [Online]. Available: https://github.com/ethereum/yellowpaper
[10] J. A. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and applications," in *Advances in Cryptology - EUROCRYPT 2015*. Berlin, Heidelberg: Springer, 2015, pp. 281–310.
[11] Y. Sompolinsky and A. Zohar, "Secure high-rate transaction processing in bitcoin," in *International Conference on Financial Cryptography and Data Security*. Berlin, Heidelberg: Springer, 2015, pp. 507–527.
[12] N. Papadis, S. Borst, A. Walid, M. Grissa, and L. Tassiulas, "Stochastic models and wide-area network measurements for blockchain design and analysis," in *Proc. of INFOCOM*. Honolulu, HI,: IEEE, April 2018, pp. 2546–2554.
[13] V. Bagaria, S. Kannan, D. Tse, G. Fanti, and P. Viswanath, "Deconstructing the blockchain to approach physical limits," *arXiv preprint arXiv:1810.08092*, 2018.
[14] Ethereum, "Mining rewsards," 2018. [Online]. Available: https://github.com/ethereum/wiki/wiki/Mining
[15] Ethereumscan, "Ethereum top 25 miners by blocks," 2018. [Online]. Available: https://etherscan.io/stat/miner?range=7&blocktype=blocks
[16] F. Ritz and A. Zugenmaier, "The impact of uncle rewards on selfish mining in ethereum," in *2018 IEEE European Symposium on Security and Privacy Workshops (EuroS PW)*. London, UK: IEEE, April 2018, pp. 50–57.
[17] Ethereum, "Design rationale: Uncle incentivization," *github, Aug*, 2018. [Online]. Available: https://github.com/ethereum/wiki/wiki/Design-Rationale#uncle-incentivization
[18] J. Gbel, H. Keeler, A. Krzesinski, and P. Taylor, "Bitcoin blockchain dynamics: The selfish-mine strategy in the presence of propagation delay," *Performance Evaluation*, vol. 104, pp. 23 – 41, 2016.
[19] L. Bahack, "Theoretical bitcoin attacks with less than half of the computational power (draft)," *arXiv preprint arXiv:1312.7013*, 2013.
[20] S. Solat and M. Potop-Butucaru, "Zeroblock: Preventing selfish mining in bitcoin," Sorbonne Universites, UPMC University of Paris 6, Amherst, MA, USA, Tech. Rep., 2016.
[21] S. Lerner, "Uncle mining, an ethereum consensus protocol flaw," *Bitslog blog, Apr*, 2016.

APPENDIX

A. The Multiple Summations Function

The function $f(x, y, z)$ used in Sec. IV-C involves multiple summations when $z > 1$. We provide several examples to explain this function.

Example 1 ($z = 1$, $x \geq y + 2$)**.**

$$\begin{aligned} f(x, y, 1) &= \sum_{s_1=y+2}^{x} 1 \\ &= x - y - 1. \end{aligned}$$

Example 2 ($z = 2$, $x \geq y + 2$)**.**

$$\begin{aligned} f(x, y, 2) &= \sum_{s_2=y+2}^{x} \sum_{s_1=y+1}^{s_2} 1 \\ &= \sum_{s_2=y+2}^{x} (s_2 - y) \\ &= 2 + \cdots + (x - y) \\ &= \frac{(x - y - 1)(x - y + 2)}{2}. \end{aligned}$$

B. Reward Analysis

Lemma 1. *Consider a new block associated with a state transition from state (i, j) with $i - j \geq 2$. It will be a regular block with probability 1 if and only if it is mined by the selfish pool.*

Proof. Suppose that the current system state is (i, j) with $i - j \geq 2$. If the selfish pool mines a new block, then the state becomes $(i + 1, j)$. Now, the private branch has an advantage of at least 3 blocks (since $i + 1 - j \geq 3$) over public branches. As the system evolves, the private branch will be published with probability 1 and become part of the system main chain, according to Algorithm 1. In other words, the new block will be a regular block with probability 1. On the other hand, if some honest miners mine a new block, we consider two cases.

1) $i - j = 2$. In this case, we have $L_h = j + 1$ and $L_s = L_h + 1$. Hence, the pool will publish its private branch and the new block becomes a stale block.
2) $i - j \geq 3$. In this case, the system state becomes either $(i, j + 1)$ or $(i - j, 1)$. The private branch has an advantage of at least 3 blocks (since $i - j \geq 3$). Hence, it will be published with probability 1. In other words, the new block will be a stale block with probability 1.

□

We are now ready to analyze every state transition. We call a new block associated with a transition a *target* block.

Case 1: $(0, 0) \xrightarrow{\beta} (0, 0)$

In this case, the target block generated by some honest miners will be adopted by all the miners. Thus, it will be a regular block and receive a static reward K_s.

Case 2: $(0, 0) \xrightarrow{\alpha} (1, 0)$

In this case, the selfish pool produces the target block, keeps it private, and continues mining on it. First, we analyze the static reward by determining whether the target block will be a regular block or not. To this end, we consider the following two subcases, which are illustrated in Fig. 12.

1) *Subcase 1*: The subsequent block is mined by the pool, which happens with probability α. As a result, the pool

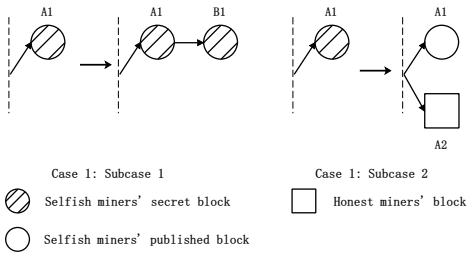


Fig. 12. The two subcases of Case 1: 1) the selfish pool mines a subsequent block; 2) some honest miners mine a new block.

owns a lead of two blocks. By Lemma 1, the target block will be a regular block and receive a static reward of K_s.

2) *Subcase 2*: The subsequent block is mined by some honest miner, which happens with probability β. Then, the pool will publish this target block. To determine whether it will be a regular block, we need to consider the following three subsubcases. See Fig. 13 for an illustration.

   Subsubcase 1: The pool mines a new block on its private branch and publishes it immediately. (This happens with probability α.) Now, the target block becomes a regular block.

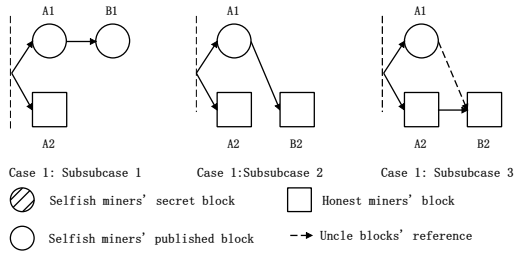


Fig. 13. The three subsubcases of Case 1: 1) the selfish pool mines a new block and publishes it; 2) the honest miners find a new block on the target block; 3) some honest miners mine a new block and reference the target block.

   Subsubcase 2: Some honest miners mine a new block on the target block. (This happens with probability $\beta\gamma$.) Now, the target block becomes a regular block.

   Subsubcase 3: Some honest miners mine a new block and references the target block. (This happens with probability $\beta(1-\gamma)$.) Now, the target block becomes an uncle block.

To sum up, the target block in *Case 2* will eventually be a regular block with probability $\alpha + \alpha\beta + \beta^2\gamma$ and be an uncle block with probability $\beta^2(1-\gamma)$. (Note that these two probabilities sum up to 1.)

Next, we analyze the uncle and nephew rewards associated with the target block. Based on our previous case-by-case discussion, only in subsubcase 3, the target block will be an uncle block. Also, the distance between the target block and its nephew block is 1. As such, the target block will bring the pool an uncle reward of $K_u(1)$ and some honest miners will receive a nephew reward of K_n. (This happens with probability $\beta^2(1-\gamma)$.)

Case 3: $(1,0) \xrightarrow{\alpha} (2,0)$

In this case , the pool produces the target block, keeps it private, and continues mining on it. By Lemma 1, the target block will be a regular block and receive a static reward of K_s.

Case 4: $(1,0) \xrightarrow{\beta} (1,1)$

In this case, some honest miners mine the target block, then the pool publishes its private block. First, we analyze the static reward by determining whether the target block is a regular block. To this end, we consider the following subcases. See Fig. 14 for an illustration.

1) *Subcase 1*: The pool mines a new block on its private branch, references the target block, and publishes its private branch. (This happens with probability α.) Now, the target block becomes an uncle block.
2) *Subcase 2*: Some honest miners mine a new block not on the target block and reference the target block. (This happens with probability $\beta\gamma$.) Now, the target block becomes an uncle block.
3) *Subcase 3*: Some honest miners mine a new block on the target block. (This happens with probability $\beta(1-\gamma)$.) Now, the target block becomes a regular block.

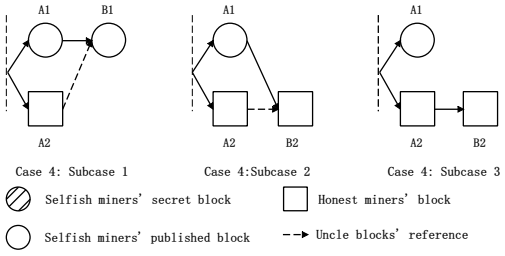


Fig. 14. The three subcases of Case 4: 1) the selfish pool mines a new block on its branch and references the target block; 2) some honest miners find a new block not on the target block and reference the target block; 3) some honest miners mine a new block on the target block.

To sum up, the target block will eventually be a regular block with probability $\beta(1-\gamma)$, and be an uncle block with probability $\alpha + \beta\gamma$.

Next, we analyze the uncle and nephew rewards associated with the target block. Based on our previous case-by-case discussion, the target block will become an uncle block only in subcases 1 and 2, where the distance is 1. Thus, the target block will bring honest miners an uncle reward of $K_u(1)$. As for the nephew reward, in *Subcase 1*, the pool receives it, and in *Subcase 2*, some honest miners receive it. To sum up, honest miners will receive an uncle block reward of $K_u(1)$ with probability $\alpha+\beta\gamma$, receive a nephew reward of K_n with probability $\beta\gamma$, and the pool will receive a nephew reward of K_n with probability α.

Case 5: $(1,1) \xrightarrow{1} (0,0)$

In this case, the target block will always be a regular block no matter who mines it. Hence, the pool receives a static reward with probability α, and some honest miners receive a static reward with probability β.

Case 6: $(i,j) \xrightarrow{\alpha} (i+1,j)$ *with* $i \geq 2$ *and* $j \geq 0$

In this case , the pool mines the target block, keeps it private, and continues mining. By Lemma 1, the target block will eventually become a regular block, receiving a static reward of K_s.

Case 7: $(i,j) \overset{\beta\gamma}{\rightarrow} (i-j,1)$ *with* $i-j \geq 3$ *and* $j \geq 1$

In this case, some honest miners mine the target block on a public branch that is a prefix of the private branch. Then, the pool publishes its first unpublished block. By Lemma 1, the target block will eventually become an uncle block.

Next, we analyze the uncle and nephew rewards associated with the target block. We begin with the special case of $(4,1) \rightarrow (3,1)$ before discussing the general case. We consider the following three subcases.

1) *Subcase 1*: The pool mines a subsequent block and references the target block. See Fig. 15 for an illustration. (This happens with probability α.) By Lemma 1, the target block will become an uncle block, receiving an uncle reward of $K_u(3)$. The pool will receive a nephew reward.

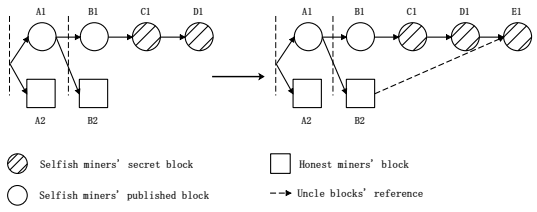


Fig. 15. The subcase 1 of Case 7, in which the pool mines a subsequent block, references the target block and eventually wins the associated nephew reward.

2) *Subcase 2*: Some honest miners mine a subsequent block on the target block. (This happens with probability $\beta(1-\gamma)$.) Then, the pool will publish its private branch. See Fig. 16 for an illustration. To determine the uncle and nephew rewards, we need to consider the following subsubcases.

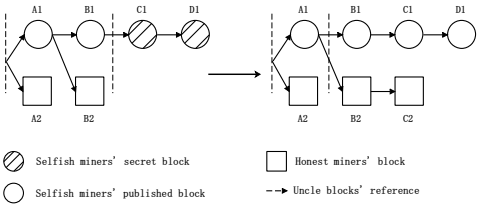


Fig. 16. The subcase 2 of Case 7, in which some honest miners mine a subsequent block on the target block.

Subsubcase 1: Some honest miners mine a new block and references the target block. See Fig. 17. This subsubcase happens with probability $\beta(1-\gamma)\beta$. The honest miner receives a nephew reward, and the target block receives an uncle reward of $K_u(3)$ since the distance is 3.

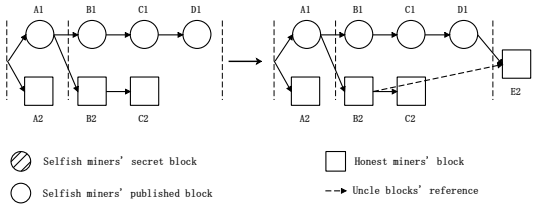


Fig. 17. The subsubcase 1 of Case 7, in which some honest miners mine a block in state $(0,0)$ and win the associated nephew reward.

Subsubcase 2: The pool mines a new block and keeps it private. This subsubcase happens with probability $\beta(1-\gamma)\alpha$. Now, if the new block later becomes a regular block (with probability $\alpha+\alpha\beta+\beta^2\gamma$ due to the discussion for *Case 2*), the pool will receive a nephew reward and the target block will receive an uncle reward of $K_u(3)$. Otherwise, if the new block later becomes a stale block (with probability $\beta^2(1-\gamma)$ due to the discussion for *Case 2*), some honest miners will receive a nephew reward and the target block will again receive an uncle reward of $K_u(3)$.

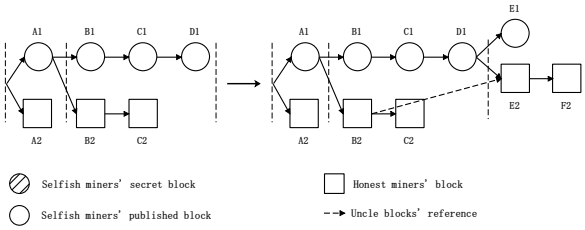


Fig. 18. The subsubcase 2 of Case 7, in which the selfish pool mines a new block in state $(0,0)$, but finally loses to the nephew reward.

3) *Subcase 3*: Some honest miners mine a subsequent block not on the target block. (This happens with probability $\beta\gamma$.) In terms of the analysis of uncle and nephew rewards, *Subcase 3* is the same as *Subcase 2*.

To sum up, for the special case of $(4,1) \rightarrow (3,1)$, the target block will always receive an uncle reward of $K_u(3)$. As for the nephew reward, one can draw a tree diagram summarizing all the subcases discussed above and conclude that it will be received by honest miners with probability $\beta^2(1+\alpha\beta(1-\gamma))$ and by the pool with probability $1-\beta^2(1+\alpha\beta(1-\gamma))$.

Here, we note that the probability $\beta^2(1+\alpha\beta(1-\gamma))$ (that the nephew reward will be received by honest miners) can be explained as follows. First, the honest miners have to "push" the system state from $(3,1)$ to $(0,0)$ while the pool mines nothing. (Otherwise, the pool will receive the nephew reward by Lemma 1.) This happens with probability β. Second, starting from $(0,0)$, the honest miners can win the nephew reward with probability $\beta(1+\alpha\beta(1-\gamma))$. This interpretation allows us to analyze the general case.

First, the honest miners have to "push" the system state from $(i-j,1)$ to $(0,0)$ while the pool mines nothing. This happens with probability β^{i-j-2}. Second, starting from $(0,0)$,

the honest miners can win the nephew reward with probability $\beta(1+\alpha\beta(1-\gamma))$. Therefore, the nephew reward will be received by honest miners with probability $\beta^{i-j-1}(1+\alpha\beta(1-\gamma))$ and by the pool with probability $1-\beta^{i-j-1}(1+\alpha\beta(1-\gamma))$.

Case 8: $(i,j) \xrightarrow{\beta\gamma} (0,0)$ *with* $i-j=2$ *and* $j \geq 1$

In this case, some honest miners mine the target block. Then, the pool publishes its private branch. Now, the target block becomes an uncle block. Similar to *Case 7*, the target block will receive an uncle reward of $R_u(2)$, and the nephew reward will be received by honest miners with probability $\beta(1+\alpha\beta(1-\gamma))$ and by the pool with probability $1-\beta(1+\alpha\beta(1-\gamma))$.

Case 9: $(2,0) \xrightarrow{\beta} (0,0)$ *with* $i \geq 2$

In this case, some honest miners mine the target block. Then, the pool publishes its private branch. The remaining discussion is the same as *Case 8*.

Case 10: $(i,0) \xrightarrow{\beta} (i,1)$ *with* $i \geq 3$

In this case, some honest miners mine the target block. Then, the pool publishes its first unpublished block. By Lemma 1, the target block will eventually become an uncle block. Similar to the discussion for *Case 7*, we conclude that the target block will receive an uncle reward of $K_u(i)$, and the nephew reward will be received by honest miners with probability of $\beta^{i-1}(1+\alpha\beta(1-\gamma))$ and by the pool with probability $1-\beta^{i-1}(1+\alpha\beta(1-\gamma))$.

Case 11: $(i,j) \xrightarrow{\beta(1-\gamma)} (i,j+1)$ *with* $i-j \geq 3$ *and* $j \geq 1$

In this case, some honest miners mine the target block. Then, the pool publishes its first unpublished block. By Lemma 1, the target block will eventually become a stale block. Since its parent block is not in the system main chain, the target block will not be an uncle block.

Case 12: $(i,j) \xrightarrow{\beta(1-\gamma)} (0,0)$ *with* $i-j=2$ *and* $j \geq 1$

In this case, some honest miners mine the target block. Then, the pool publishes its private branch. Similar to *Case 11*, the target block will neither be a regular block nor an uncle block.