Detective Mining: Selfish Mining Becomes Unrealistic under Mining Pool Environment

Suhyeon Lee*† and Seungjoo Kim*‡
*CIST(Center for Information Security Technologies), Korea University, Korea
†ADD(Agency for Defense Development), Korea

***Abstract*—One of Bitcoins core security guarantees is that, for an attacker to be able to successfully interfere with the Bitcoin network and reverse transactions, they need to control 51% of total hash power. Eyal et al., however, significantly reduces Bitcoins security guarantee by introducing another type of attack, called "*Selfish Mining*". The key idea behind selfish mining is for a miner to keep its discovered blocks private, thereby intentionally forking the chain. As a result of a selfish mining attack, even a miner with 25% of the computation power can bias the agreed chain with its blocks. After Eyal's original paper, the concept of selfish mining has been actively studied within the Bitcoin community for several years. This paper studies a fundamental problem regarding the selfish mining strategy under the existence of mining pools. For this, we propose a new attack strategy, called "*Detective Mining*", and show that selfish mining pool is not profitable anymore when other miners use our strategy.**

***Keywords*—Blockchain, Bitcoin, Selfish mining, Mining pool, Security analysis**

I. Introduction

A. Motivation

Bitcoin uses Proof-of-Work (PoW) to its block generation mechanism [9]. It was designed to make fair mining competition which relies on each miner's computing power. However, for several years, PoW mechanism has met with some challenges. Especially, the "*Selfish Mining*" strategy by Eyal [4] makes an adversary who has only more than 25% - this value depends on the network environment - can get revenue over its hashrate proportion. The existence of the selfish mining not only means it is unfair to solve PoW puzzles but also a severe flaw in the integrity of blockchain.

After Eyal's paper, many researchers have studied various aspects of the selfish mining strategy. Nayak [10] and Sapirshtein [14] optimized Eyal's strategy respectively. Ritz [13] and Niu [11] applied the selfish mining strategy in Ethereum [2] which uses modified PoW mechanism with orphan blocks. Grunspan [6] studied the selfish mining's profitability under the more realistic premises. Also some researches studied the defense mechanisms against selfish mining [12], [15], [18]. However, most of the proposed methods are not practical because they require a lot of modifications of the Bitcoin protocol itself, and so they are not widely used. For this reason, the selfish mining attack is still destructive.

In this paper, we study a fundamental problem regarding the selfish mining strategy under the existence of mining pool. Though the selfish mining strategy is studied in various aspects, we think that a more realistic mining environment should be considered. In real environment, most of mining is done by mining pools [17]. Therefore, when discussing selfish mining strategy, we should consider the structure of mining pool and the information shared among pool members for more accurate analysis.

This paper is organized as follows. An overview of the selfish mining researches will be given in section II. In section III, we will analyze how the mining pool works, what information is shared among its members, and how to get it. Based on these, we propose a detective mining strategy against the selfish mining in section IV. Our method can easily detect selfish mining pool, and neutralize it. And then, we show the simulation result on our strategy in section V. Discussion and conclusions are given in the last sections.

B. Contributions

The main contributions of this paper are summarized as follows:

- We propose the strongest mining strategy, named "*Detective Mining*", in pooled mining environment. It gives big extra revenue to miners even under the existence of selfish miners.
- We show that the selfish mining is not feasible in the real environment of Bitcoin where pooled mining power is dominant.

II. Selfish Mining Strategy

A. Single selfish miner case

Eyal[4]'s model of selfish mining is illustrated in figure 1. To clarify our description, we must define terms and relevant states. There are two parameters, α and γ. Each definition is given below the figure. Eyal's research [4] fixed γ for ease of evaluation. Of course, the γ value depends on the state of the network, so it cannot be constant. However, our paper also fixes this value for ease of evaluation.

The basic idea of the selfish mining is that one adversary does not publish valid blocks to make others waste their mining on the puzzle that was already solved. The blocks which the adversary keeps in secret is referred as a private chain. We use the term, $lead$ when an adversary who does selfish mining has a longer private chain. The lead is used only when attacker's chain is ahead, and can have negative values in some optimal selfish mining models [10], [14].

‡ Corresponding author

Now we describe an ordinary selfish mining strategy with the figure 1. When the adversary does not lead, it is the state S0 without fork. So miners are working on the same block. If there are forked chains, it is the state S0'. The state S0' indicates they have different chains which have the same length and these chains compete with each other.

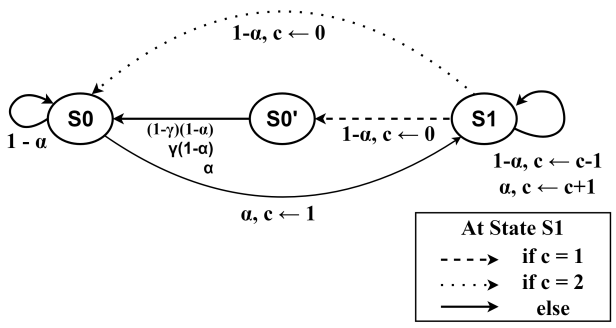

Figure 1: State machine of selfish mining

where

α is hashrate of the selfish miner. ($0 \leq \alpha \leq 0.5$)
γ is a proportion of the miners who mines on selfish miner's chain during fork. ($0 \leq \gamma \leq 1$)

Each state in the above figure means :

- State S0: The adversary mines the block on the public chain with the highest height. If the adversary finds a block, the adversary gets one lead by not publishing it. If the others find a block, the adversary accepts this block, and the state is still 0.
- State S0': This state happens when the other miners find a block A and the adversary had mined a block B unpublished on lead 1. Then the adversary publishes the block B to compete with the block A published by others. All miners except the adversary freely mine among two chains and one of two chains wins finally.
- State S1: If the adversary finds a block at S0, the adversary gets lead by not publishing it. The number of blocks the adversary leads is counted by variable c. If the others find a block, the adversary's next state is decided by reference to the variable c. Provided that the variable c is 1, the adversary's state goes to S0 and the adversary's chain competes with another chain. Provided that the variable c is 2, the adversary published all blocks and get all revenue from them. At the other positive c values, the adversary does not change the state and just decrease c. If the adversary finds a block at S1, the adversary increases c.

With this strategy, the selfish miner gets revenue by the following condition :

$$R_{selfish} = \frac{\alpha(1-\alpha)^2(4\alpha + \gamma(1-2\alpha)) - \alpha^3}{1 - \alpha(1-(2-\alpha)\alpha)} \tag{2.1}$$

$$\frac{1-\gamma}{3-2\gamma} < \alpha < \frac{1}{2} \tag{2.2}$$

B. Multiple selfish miners case

More general selfish mining model is shown in figure 2. Here we assume that each selfish miner uses the same strategy. The difference between figure 1 and figure 2 is in the state transition of S1. In the case of single selfish miner, only one block is published at a time. However, in case of multiple selfish miners, many blocks can be suddenly published at a time.

Assume that a selfish miner SM_i leads by c_i. If another selfish miner SM_j, whose $c_j > c_i$, publishes his hidden blocks at a time, then SM_i should give up all of his hidden blocks. Figure 3 illustrates this situation. Here, $MAX(\cdots)$ function returns the largest value from the lead values provided.

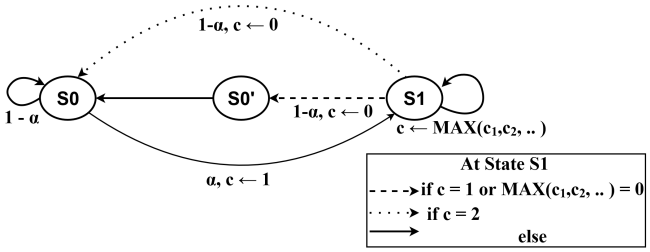

Figure 2: State machine of the general selfish mining

For example, let Bob and Cathy be selfish miners. Bob and Cathy have three and four unpublished private blocks respectively. Suppose one of the other honest miners finds and publish a new block. The figure 3 illustrates this situation. Then, Bob has only one lead so that he should publish his three blocks. After that, Cathy has one lead so that she should publish her four blocks. Consequently, one block caused the publication of three blocks at once. Bob does not get any revenue from his three hidden blocks. Among them, only Cathy get revenue from her four blocks.

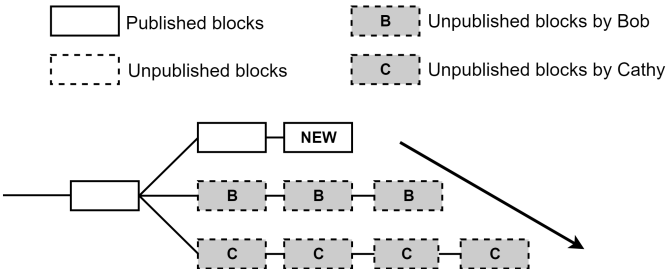

Figure 3: Serial disclosure of multiple selfish miners

Furthermore, there can be a fork situation. For example, Let's assume that there are three selfish miners. All of these selfish miners have one lead (i.e., c = 1 and all of their states are S1.). If a block is published in the public chain, other selfish miners will not be able to reveal their hidden blocks, consequently resulting in a fork situation with four different chains. Now the honest miners should choose a chain to mine. In the single selfish mining case, we use the parameter γ to define how many miners work for the selfish miner's chain. However, in the multiple selfish mining case, we need more parameters for each fork. In this paper, to simplify the proof,

we will not define all the parameters. Even so, there is no significant changes in the final result. In figure 2, we omit detailed description on the fork resolution transition (S0' to S0). Instead, for its simplicity, we assume that honest miners choose where to mine fully randomly in fork situations.

Leelavimolsilp et al. [8] and Bai et al. [1] investigated the situation with multiple selfish miners and their revenue. Their works commonly show that multiple selfish miners compete to each other and diminish revenue than single selfish mining case. Similar results will be shown in section V.

III. Analysis of Mining Pool Structure and Its Leaked Information

In this section, we analyze the structure of the mining pool, how it operates, and what information is inevitably leaked from the pool manager to pool members.

Eyal proposed the selfish mining strategy and other researchers have studied the optimized version of it. However, their researches overlooked three aspects of the real modern mining environment. First of all, most of miners work in mining pools. The figure 4 shows the Bitcoin hashrate distribution. As can be seen in the figure, most of the mining (at least 78.7%) is done on a pool basis and not on an individual basis. Furthermore, some mining pools have quite big hashrate, and this situation will accelerate as Bitcoin's mining difficulty level increases. Second, most of mining pools are public, thus miners can freely join and leave the pools. (The article [17] says only a few pools are closed.) Third, miners in the same mining pool are forced to share specific information with the pool manager due to the structural nature of the pool.

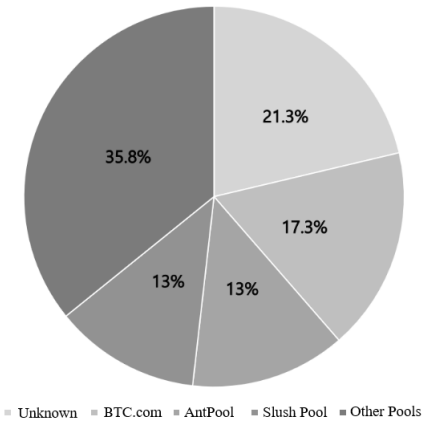

Figure 4: Hashrate distribution by Blockchain.com

In the following subsections, we will sketch the block structure of Bitcoin, and then explain the communication protocol between a mining pool manager and miners. Parameters used in this section are described in table I.

A. Bitcoin's Block Structure

The Bitcoin block consists of two parts, header and transaction as shown in Figure 5 [3].

The header part contains 6 elements. *Version* indicates the current version of the Bitcoin protocol. *PrevBlockHash* is the hash value of the previous Bitcoin block in order. By containing the hash value of the previous block header, the next block keeps the immutability of the context. *MerkleRootHash* is the hash value of the transaction part. *Timestamp* is the time when the block created. *Nonce* is the random value as a solution to the PoW puzzle of Bitcoin. *bits* is the difficulty of the current Bitcoin network.

Table I: Parameter explanation

Parameters	Description
v	the version of Bitcoin
r	prevBlockHash
m	a list of transactions
st	timestamp
T	the current difficulty target of Bitcoin network
$H(m)$	MerkleRootHash
ctr	nonce
q	the size of the nonce value
D	the difficulty value for share in a mining pool

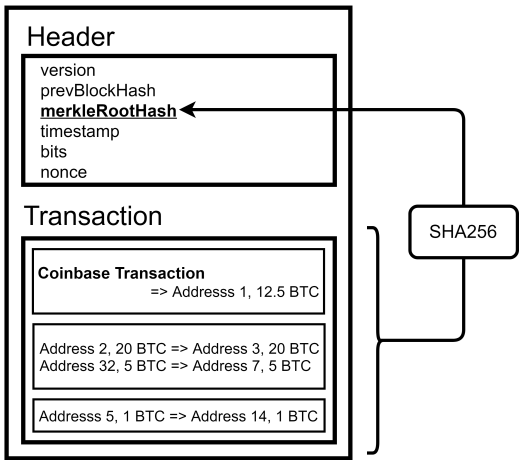

Figure 5: Bitcoin block structure

The transaction part contains Bitcoin transactions with the sender, the receiver and the amount of Bitcoin. All of the transactions in the block are hashed. Its result value is the Merkle root, *MerkleRootHash*. Among transactions, the first transaction is a special transaction called *Coinbase Transaction*. This transaction is for giving the Bitcoin reward to the miner who first publishes a valid block in the Bitcoin network. Since the coinbase transaction is a minting process, it does not contain the sender information of that transaction. Here and now, 12.5 Bitcoins are given to the publisher as a reward.

To publish a Bitcoin block, miners should find a proper *nonce* value which satisfies the difficulty condition. According to Garay's work [5], we can describe this as the following equation :

$$(H(v, r, H(m), st, T, ctr) < T) \wedge (ctr \leq q) \tag{3.1}$$

Here, $H(\cdot)$ is a cryptographically secure hash function such as SHA256. The hash output of the values in a block should be less than a target difficulty. To meet this condition, mining should involve a ton of brute force computing with the hash function.

B. Mining Pool's Communication Protocol

In order to explain the communication mechanism of mining pools, we will use the Stratum mining protocol [16] which is a text based communication protocol for mining pools. Figure 6 illustrates miners in mining pools. As we mentioned above, miners can rarely find a valid block, even though they work hard in a mining pool. Thus we need a new *indicator* in order to check the contribution of each miner. Let D be this indicator. This numerical value is bigger than *difficulty* (T in the inequality 3.1) of the network. In other words, D is easier to meet the condition than T. By using this, miners can solve the easier PoW puzzles and the pool manager can check each miner's contribution. Sometimes the solution submitted by the miner will satisfy *difficulty* T. Then the pool manager publish it to the Bitcoin network as valid block. We call a solution which meets the difficulty T as a full Proof-of-Work (fPoW), and a solution meeting the indicator D of the pool as a partial Proof-of-Work (pPoW) respectively.

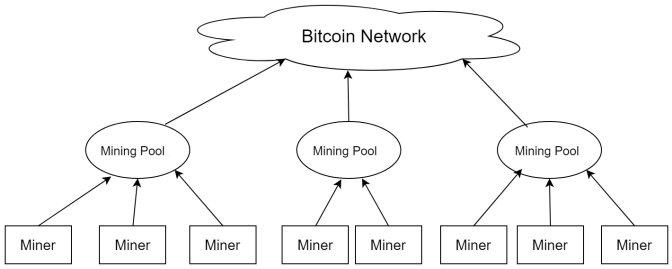

Figure 6: Bitcoin mining pools

For miners, their PoW task w consists with v, r, m, st, q except ctr which is a nonce to be found. Their mining pool manager distributes PoW task as 3.2. Then each miner in that pool should find a proper nonce which meets the inequality 3.3.

$$w = (v, r, m, st, q) \tag{3.2}$$

$$(H(v, r, H(m), st, T, ctr) < D) \wedge (ctr \leq q) \tag{3.3}$$

C. Leaked Information in Mining Pool

According to [17], most of mining pools are public, so miners can freely join and leave the pools. Also, miners in the same mining pool are forced to share specific information with the pool manager due to the structural nature of the pool.

There are many ways in which a third party can obtain this shared (in other words, leaked) information. One of the simple methods is *infiltration* [7]. In figure 7, the mining pool *F* try to infiltrate into the mining pool *S*. The manager of the mining pool *F* behaves as a proxy for the mining task of pool *S*, and its miners work normally for the given task of *S*. Here we should note that the manager of the infiltration mining pool *F* can see all the information in the task given by the mining pool *S*. The task includes all the information specified in the expression 3.2.

In another example, a miner can split his mining power into two parts. One is for his own mining and the other is for infiltrating into other mining pool which has big enough mining power. Then the infiltrating miners can fetch information of the target pool.

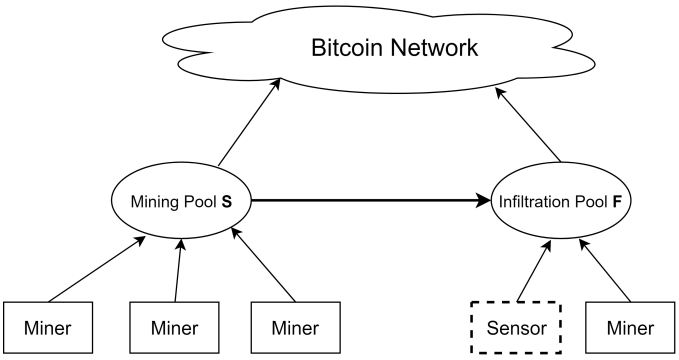

Figure 7: Leakage example: Infiltration between mining pools

IV. Our proposal: Detective Mining

A. Basic idea

Generally, the selfish miner tries to keep its private chain longer than the public chain. Since the honest miners do not know the existence of the longer private chain, the selfish miner can earn the maximized revenue than the honest miners.

However, if there is a way to know that there is a selfish miner hiding the block, the situation changes. Moreover, if the selfish miner does not know that we know it, the situation becomes even more dramatic. We call this as "*Detective Mining*" strategy. The term, detective mining (a.k.a. ghost mining), originally means working old mines and finding gold previous miners have left behind. Our strategy is very similar with this.

As said before, nowadays, most of mining is done by pool basis [17], and the miners in same mining pool have no choice but to share some information such as *Coinbase Transaction* or *MerkleRootHash* with the pool manager due to the structural nature of the pool. By using this we can easily detect the pooled selfish mining. If a mining pool is honest, then it does mining on the last block of the longest public chain. But, if a mining pool is selfish, it sometimes does mining with an unknown previous block on its private chain. Therefore, if we can distinguish these two cases, we are able to detect the selfish mining.

After finding the selfish mining pool, to maximize our revenue, we (called as "*Detective Miners*") can work on the selfish mining pool's hidden blocks stealthily. In section III, we showed that the shared task among miners in the same pool included the hash of the previous block, *prevBlockHash*. This shared (in other words, leaked) information of the selfish mining pool can be used for our detective mining. This mining process is described in algorithm 1.

If the detective miners find a new valid block on the private chain, they publish it, and then the selfish mining pool can not keep its chain secret any more. If a block after the private chain is published, the selfish mining pool should admit that it is not in lead state anymore. In this situation, for getting revenue, the smartest way is to open up all selfish mining

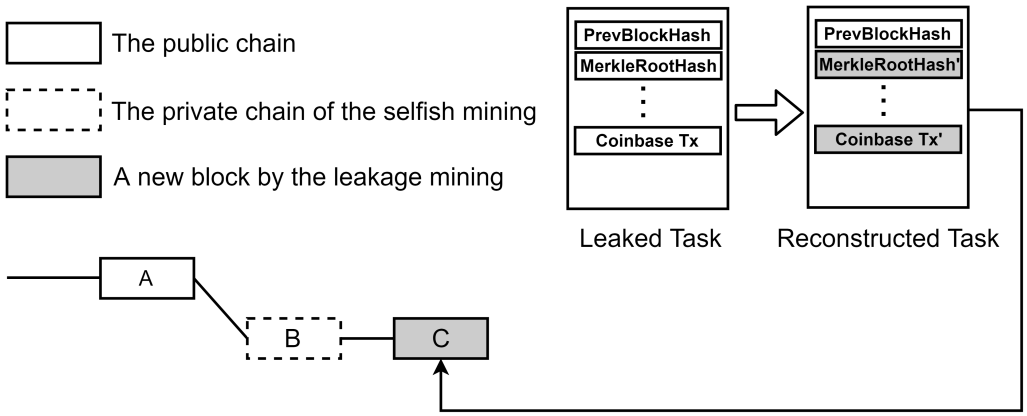

Figure 8: detective mining

Algorithm 1 Create a PoW task based on shared information

```
w: the shared PoW task
r: the shared prevBlockHash
m': the new transaction list obtained by the coinbase of the detective miner
w': the counter PoW task

procedure CREATETASKFROMSHAREDINFO
    if w is unknown then
        (v, r, m, st, T, q) ← w
        w' ← CreateTask(v, r, m', st, T, q)
        return w'
```

pool's private chain and pray for honest miners to choose its chain.

Figure 8 illustrates the whole process of our detective mining. The block A and the prior blocks of A consist the public chain. Let's assume that block B is on the private chain of selfish mining pool. In general, honest miners work at the block A which is the latest block of the public chain, and the selfish mining pool work at its private block B. On the other hand, the detective miners can work at block B by reconstructing a new task from the leaked information (i.e., the shared hash value of the private block B) from the selfish mining pool. If the detective miners succeed to find and publish a new block C which follows the block B, the selfish mining pool must inevitably release its block B. Then the block B and C become the public chain.

B. Simple model of detective mining

At first, we make a simple detective mining model against one selfish miner. The model is shown in the following figure 9.

If the detective miners find a block after the private chain, every positive lead state can transits to the state S0. We define this possibility as δ, which means the hashrate of the detective miners. For an intuitive understanding of the process, we add a memory element c. The memory value c indicates how far ahead the selfish miner's chain. It is updated in several transitions. The memory element reduces infinitely many states into one state (S1). S0 means there is no advance on the private chain. In S0, every miners work on the public chain. S0' means the height of the public chain and the selfish mining pool's chain is the same. State S1 means the private chain of the selfish mining pool is longer than the public chain. For example, if the selfish mining pool finds a block in advance at S0, the state transits to S1 with setting c to 1. As another example, in the state S1, miners find a new block in the public chain with the probability of $1 - \alpha - \delta$. In this case, the next transition depends on the memory value c. It transits to S0 if c is 1 and sets c to 0. If c is 2, it transits to S0' and sets c to 0. Otherwise, it just decreases c. Due to the detective mining, the state machine model contains a transition to zero state (S0) whenever the selfish mining pool has a longer chain (S1). This transition by δ possibility indicates the success of the detective mining. That is, detective mining disturbs the selfish mining pool to maintain its long private chain. At the same time, the detective miners' revenue increase as their block become the main chain. We will check their revenue variation at the next section. At the competition state 0', the detective miners can randomly choose which chain to be mined. So the transitions from the state 0' is same with the previous figure.

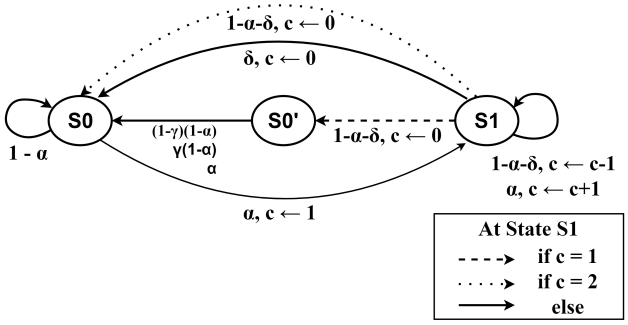

Figure 9: Simple model of the detective mining

where

α is the hashrate of the selfish mining pool
δ is the hashrate of the detective miners
γ is a proportion of the miners on the selfish pool's chain in a fork situation.

C. General model of detective mining

In a general model, we apply the detective mining strategy to the multiple selfish mining model at the section II. For simplicity, we assume that the detective miner mines at the longest chain among all the chains of selfish mining pools. Also we assume that if the public chain and selfish mining pools' chains compete in fork situation, the detective miners mine for the public chain. If only selfish mining pools' chains compete with each other in a fork situation, detective miners randomly choose which chain to be mined for. There can be an informal fork situation that plural selfish mining pools have the same height of private chains. In this case, detective miners also chooses a chain randomly.

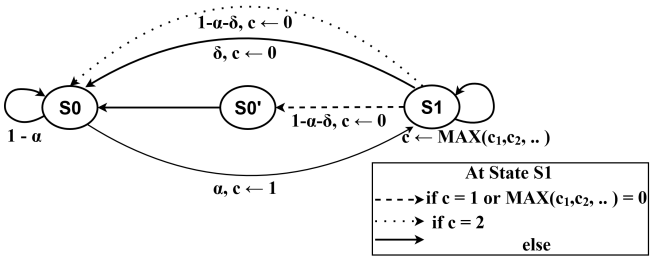

Figure 10: General model of the detective mining

where

α is the total hashrate of the selfish mining pools
δ is the hashrate of the detective miners

V. Simulation and Results

To validate our strategy, we simulate the revenue variation of the selfish mining in various settings. In the following subsections, simulation results of one selfish mining pool model and multiple selfish mining pools model are shown respectively. All simulations are evaluated with 50,000 blocks, which are the number of Bitcoin blocks mined during 2 years. We will calculate the percent ratio of selfish mining pools among entire blocks.

A. Simple model

The first simulation is based on the simple model with only one selfish mining pool. To show detective mining suppresses a selfish pool's revenue, we check the ultimate (figure 11) and the relative (figure 12) variation of revenue.

In figure 11, three plots are simulations under γ is 0, 0.5 and 1 respectively. In each plots, three lines indicates revenue of a selfish mining pool under the different proportions of detective miners. To explain with the simple model, three lines indicate revenue of a selfish mining pool with δ value 0, $\frac{0.5}{1-\alpha}$, and $\frac{1}{1-\alpha}$ respectively. For simplicity, let θ is the percent ratio of detective miners except the selfish miners. Using θ, the three lines indicate revenue of a selfish mining pool with $\theta = 0\%$, 50%, and 100% respectively. In figure 11, we use the below symbols.

- H_S is the hashrate of the selfish mining pool
- R_H is the revenue of the honest mining
- R_S is the revenue of the selfish mining pool when θ is 0%
- R_{S50} is the revenue of the selfish mining pool when θ is 50%
- R_{S100} is the revenue of the selfish mining pool when θ is 100%

In the plot, the X-axis shows percent of the selfish mining pool's computing power. The Y-axis shows percent of the selfish mining pool's revenue among all miners. The black solid line, R_H, is the revenue of the honest mining. So it determines whether the selfish mining strategy is extra profitable or not. In the three lines except the black line R_H, we can see that the revenue decreases significantly as the detective mining increases. As the ratio of detective miners decreases, our strategy is more critical to the selfish mining pool. When 50% of the rest miners join to the detective mining (the double dotted line, R_{S50}), the selfish mining is not profitable than the honest mining under 32% of computing power. Furthermore, the threshold point that the selfish mining is extra profitable is higher than there is no detective miners. For example, in the line R_{S50}, at $\gamma = 0$, 42%, at $\gamma = 0.5$, 37%, and at $\gamma = 1$, 32% are thresholds respectively.

The figure 12 shows variation of relative revenues of the selfish pool, the detective miners, and the rest honest miners. The three graphs are under α values 0.35, 0.4 and 0.45 respectively. They are each red dot on the middle graph in figure 11 when θ is 50%. Not to be biased, we choose these three points on the graph when γ is 0.5. We use relative extra revenue (RER) to evaluate the impact of the detective mining. It indicates the proportion of the extra revenue to the honest mining. If RER is negative, a miner earns less revenue than the honest mining. Thus, only when RER is positive, our new strategy is effective. It is given in formula 5.1.

$$RER = \frac{R_n - R_h}{R_h} \tag{5.1}$$

where

R_n is the revenue of miners with our new strategy
R_h is the revenue of miners with the honest mining

In the figure 12, we use below symbols:

- RER_S is RER of the selfish mining pool
- RER_N is RER of the detective miners
- RER_O is RER of the other miners except the selfish mining pool and the detective miners

The simulation evaluated RER of the selfish mining pool, the detective miners and the other miners. The X-axis of the graph means variation of the ratio of the detective mining to the miners except the selfish miners. It is performed with the fixed value $\gamma = 0.5$. In total range, RER of the selfish mining pool (the black solid line, RER_S) decreases as the number of detective miners increases. The detective miners (the double dotted line, RER_L) get high RER value. And the other honest miners (the dotted line, RER_O) who implement neither selfish

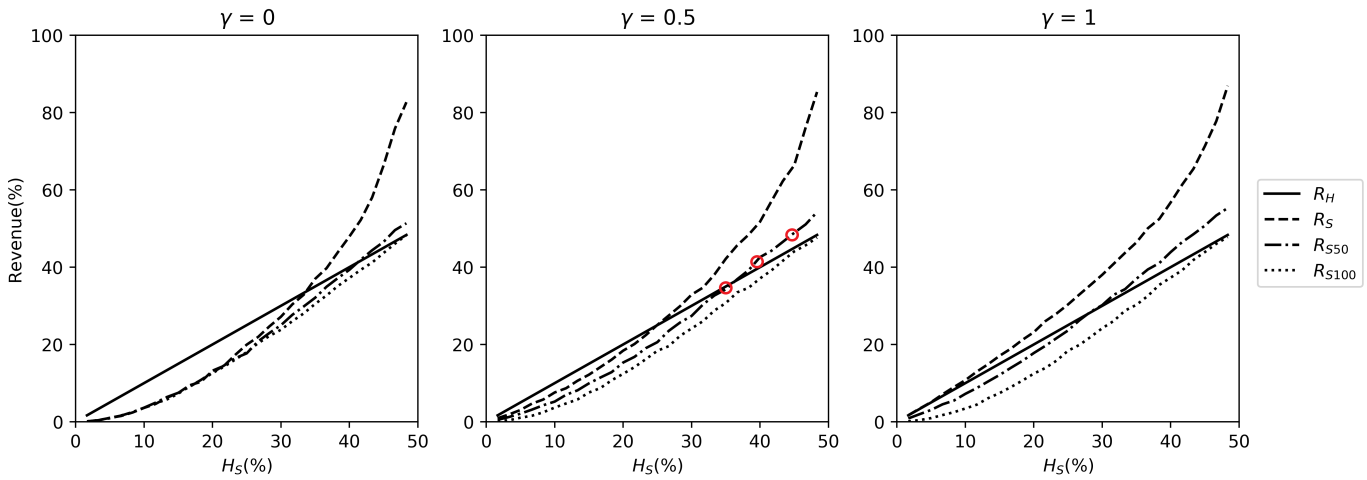

Figure 11: Simple model simulation and revenue

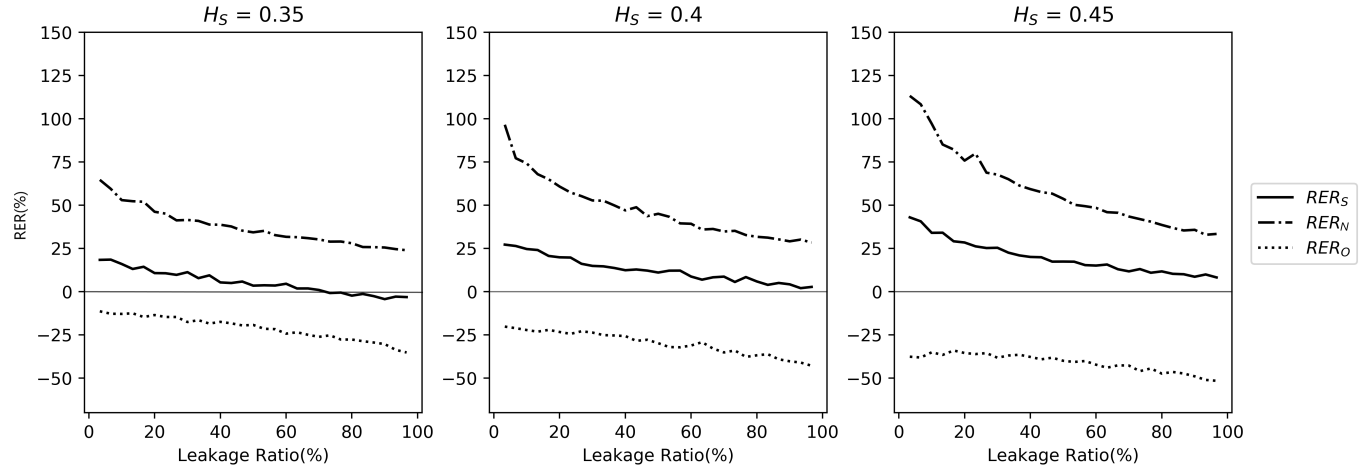

Figure 12: Simple model simulation and RER variance

mining or detective mining get negative RER value. Hence, it is always advantageous to use the detective mining in the presence of a selfish mining pool. The selfish mining pool gets much smaller RER value than the detective miners, and this gives enough motivation for miners to use our method.

To summarize, our detective mining is very effective against the selfish mining pool. Furthermore, it gives much higher revenue to the miners than the honest strategy. Provided that there are enough detective miners, the selfish mining pool gets less revenue than it uses the honest strategy.

B. General Model

In the above simple model simulation, we found the decrease of the selfish pool's revenue and big RER of the detective miners. To analyze our method in general cases, we simulated our proposal against multiple selfish mining pools. Here we assume that multiple selfish mining pools follow the general model at the previous sections.

We simulated two, three and four selfish pools respectively with several join rate of detective mining. The first figure 13 is result under no detective miners ($\theta = 0\%$). The second figure 14 is a result under half of the rest miners joining in the detective mining ($\theta = 50\%$). The last figure 15 is a result under all the miners joining in the detective mining except the selfish mining pools ($\theta = 100\%$). The value N above graphs means the number of selfish pools. In this simulation, we assume that multiple selfish mining pools have the same computing power. Take note that the range of X-axis are different. It is trivial if there is any selfish miner with bigger hashpower than honest miners, the selfish miner's hidden chain won't be published as its chain is always longer than the public chain. The figure 16 illustrates this situation. Let Bob and Cathy be selfish miners with bigger hashpower than the rest of the miners. As each hashpower of them are bigger than others, their private chain exceeds many blocks than the public chain at some time. Furthermore, they don't publish their private chain forever as the public chain cannot catch the length of their chain. In this case, we cannot count their revenue in normal ways.

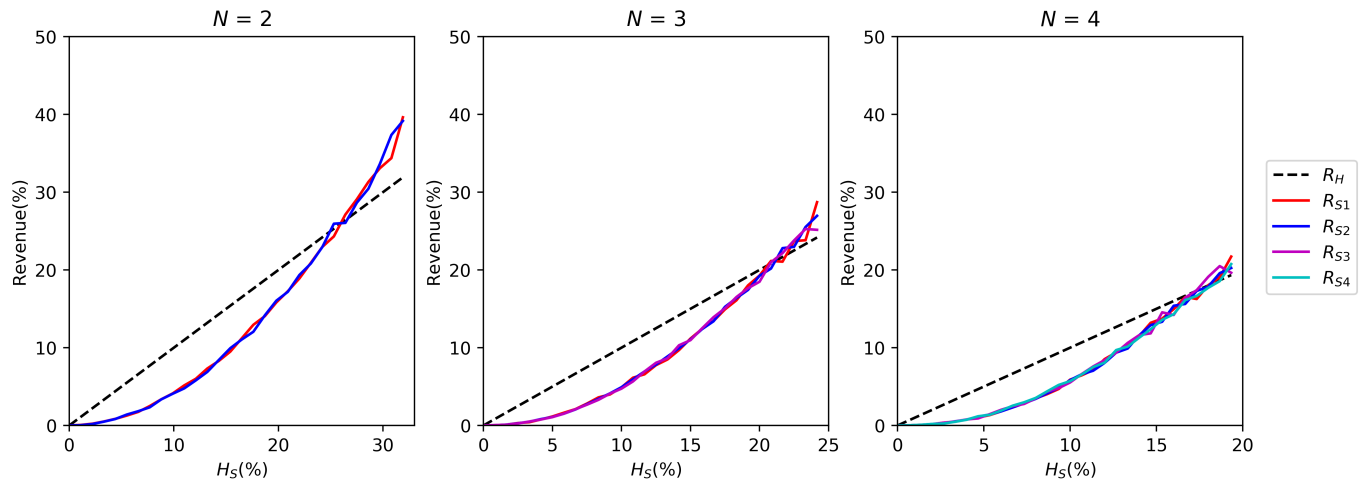

Figure 13: General model simulation when θ is 0%

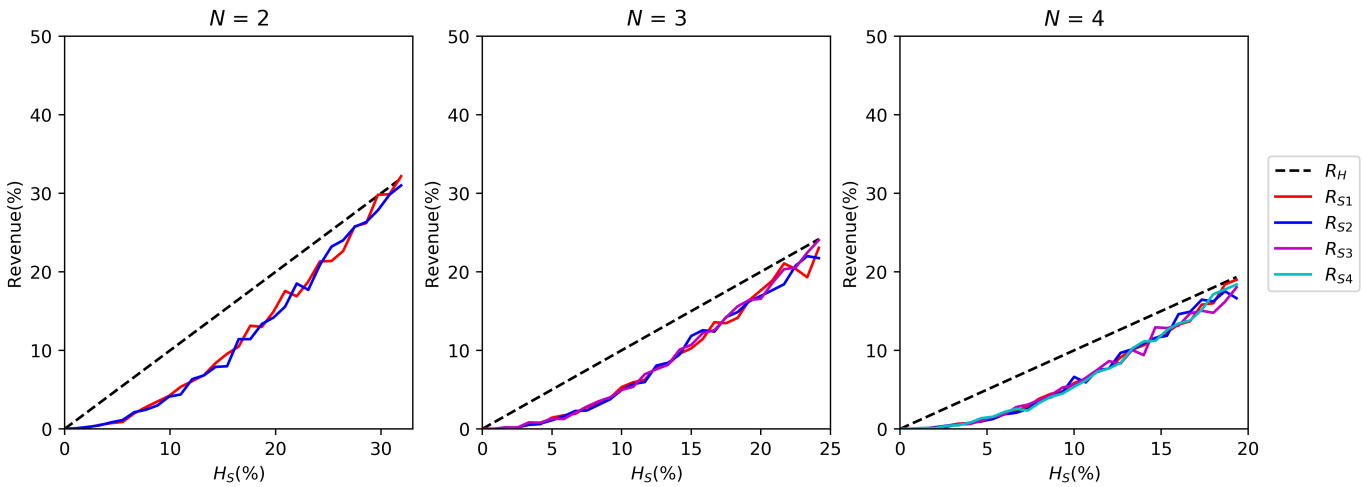

Figure 14: General model simulation when θ is 50%

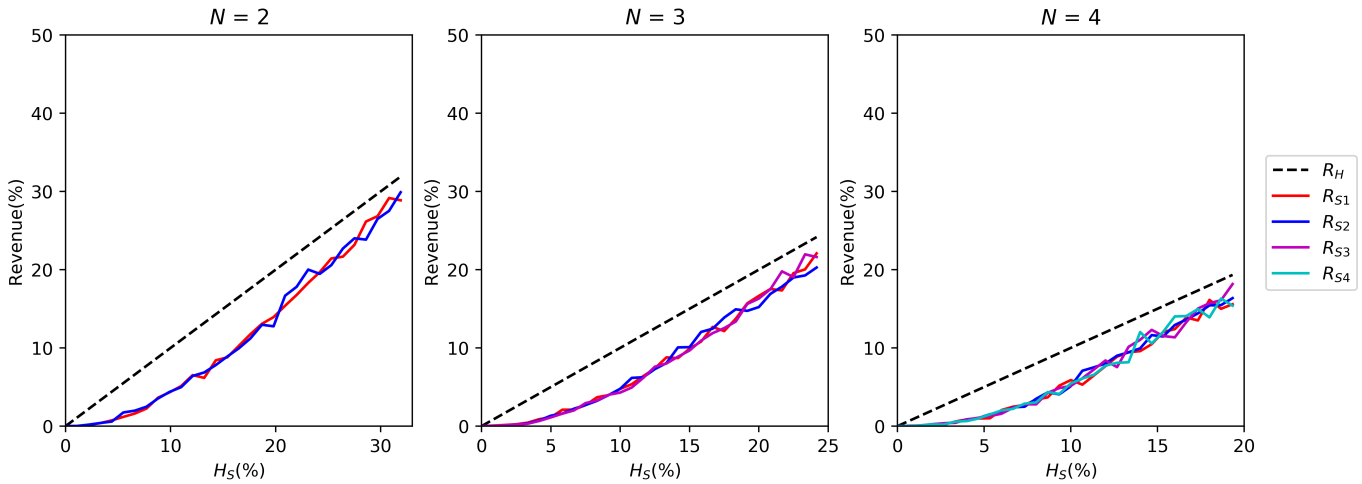

Figure 15: General model simulation when θ is 100%

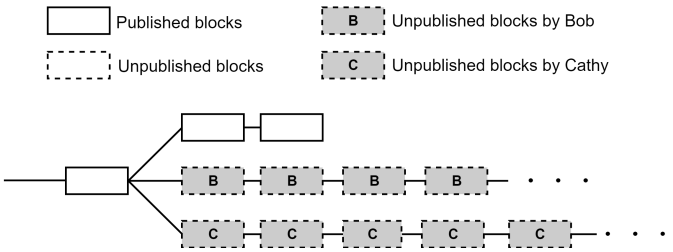

Figure 16: Endless private chains by two selfish miners

Therefore, we limit selfish miners' hashpower not to exceed honest miners' hashpower. For example, when three selfish miners have 25% hashpower respectively, the rest hashpower is 25%. If selfish miners have more hashpower, the rest hashpower is less than the selfish miner. To generalize, if there are N selfish pools with the same hashpowers, their limit is $1/(N+1)$. Therefore, 33%, 25% and 20% are limits for $N = 2, 3, 4$ respectively. In every graph, a block line shows expected revenue by honest mining and it becomes a baseline to evaluate revenue of selfish pools.

In figure 13, we can see selfish pools get less revenue than the single selfish mining. We think they exploit each other and it makes their blocks wasted. When there are four selfish pools, they nearly don't get extra revenue than honest mining. In the conventional selfish mining, the threshold to be extra profitable is near 25%. On the other hand, we found an interesting fact that their threshold to get extra revenue decreases as the number of selfish miners increases. For example, at $N = 2$, 27%, at $N = 3$, 2%, and at $N = 4$, 18% are thresholds respectively.

Outstandingly, with θ is 50% or 100%, the selfish pools are not extra profitable anymore. It means multiple selfish pools will not be a valid strategy with enough detective miners.

VI. Extensibility and Security Analysis

Variations of Selfish Mining Strategies Optimal selfish mining strategies were studied in Saprishtein [14] and Nayak [10]. In the middle of them, the optimal selfish mining [14] uses dynamic selfish mining strategies. They depend on miner's hashrate and network environment to make their revenue optimal. Basically, they share the basic structure of the selfish mining strategy. Their optimal strategies are based on not publishing blocks. Hence, we are sure that they are also significantly affected by our detective mining.

Optimization of the Detective Mining The model of the detective mining is quite complicated so that we did not make theoretical calculation. Nevertheless, we believe our strategy can be optimized and advanced more. We guess an optimized form of the detective mining will be similar with the dynamic selfish mining model of Saprishtein [14]. We leave it for the future work.

Security We can consider an adversary who wants to bypass the detective mining. This adversary needs to hide the core information used to reconstruct the PoW task, *PrevBlockHash*. This value is necessary for miners to solve PoW puzzles in the Bitcoin. It is, therefore, impossible to distribute PoW task without the hash of the previous block. For another bypass method, we can consider the adversary who changes the coinbase transaction in order to hinder detective miners from reconstructing a new PoW task. In Stratum protocol, a mining pool manager gives transaction information to miners [16]. Not like the Stratum protocol, mining pools can create a PoW task which does not include a transaction list. Miners need the hash value of the merkle root, not the transaction list in solving PoW directly. So, the detective miners should create a transaction list and obtain *MerkleRootHash* from it. If the selfish mining pools do not leak detailed transaction information and already generated private blocks, the context of a block generated by the detective miner can conflict with transactions already contained in private blocks. For example, let the transaction A be already included in a private block B_{200} of height 200. Without this knowledge, the detective miner can generate a block B_{201} of height 201. If the block B_{201} also includes the transaction A, the block B_{201} cannot be valid. In this case, the detective miner can avoid this integrity problem by generating an empty block only with coinbase transaction.

VII. Conclusion

In this paper, we proposed a new strategy, named *detective mining*, to counter the selfish mining pools by investigation of the selfish mining strategy and shared information in mining pools. We designed our method with information in mining pools can easily be shared to miners. In the result of simulations, we show that selfish mining pools get big damage by our strategy. Moreover, miners can get significant extra revenue by adopting our method. It motivates that miners to apply our method. Then selfish mining will not be feasible anymore in the real Bitcoin environment.

References

[1] Qianlan Bai, Xinyan Zhou, Xing Wang, Yuedong Xu, Xin Wang, and Qingsheng Kong. A deep dive into blockchain selfish mining. *arXiv preprint arXiv:1811.08263*, 2018.

[2] Vitalik Buterin. A next-generation smart contract and decentralized application platform.

[3] Bitcoin Wiki Contributors. Bitcoin wiki : Bitcoin protocol documentation, 2019. [Online; accessed 29-March-2019].

[4] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. *Communications of the ACM*, 61(7):95–102, 2018.

[5] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 281–310. Springer, 2015.

[6] Cyril Grunspan and Ricardo Pérez-Marco. On profitability of selfish mining. *arXiv preprint arXiv:1805.08281*, 2018.

[7] Suhyeon Lee and Seungjoo Kim. Countering block withholding attack efficiently. *IEEE INFOCOM 2019 Workshops - CryBlock 2019 (Workshop on Cryptocurrencies and Blockchains for Distributed Systems)*, April 2019. To be published.

[8] Tin Leelavimolsilp, Long Tran-Thanh, and Sebastian Stein. On the preliminary investigation of selfish mining strategy with multiple selfish miners. *arXiv preprint arXiv:1802.02218*, 2018.

[9] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[10] Kartik Nayak, Srijan Kumar, Andrew Miller, and Elaine Shi. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pages 305–320. IEEE, 2016.

[11] Jianyu Niu and Chen Feng. Selfish mining in ethereum. *arXiv preprint arXiv:1901.04620*, 2019.

[12] Rafael Pass and Elaine Shi. Fruitchains: A fair blockchain. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, pages 315–324. ACM, 2017.

[13] Fabian Ritz and Alf Zugenmaier. The impact of uncle rewards on selfish mining in ethereum. In *2018 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 50–57. IEEE, 2018.

[14] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. Optimal selfish mining strategies in bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 515–532. Springer, 2016.

[15] Siamak Solat and Maria Potop-Butucaru. Zeroblock: Timestamp-free prevention of block-withholding attack in bitcoin. *arXiv preprint arXiv:1605.02435*, 2016.

[16] Slush Pool Team. Stratum mining potocol, 2019. [Online; accessed 29-March-2019].

[17] Jordan Tuwiner. 10 best and biggest bitcoin pools, 2019. [Online; accessed 29-March-2019].

[18] Ren Zhang and Bart Preneel. Publish or perish: A backward-compatible defense against selfish mining in bitcoin. In *Cryptographers Track at the RSA Conference*, pages 277–292. Springer, 2017.