Spy Based Analysis of Selfish Mining Attack on Multi-Stage Blockchain

Donghoon Chang , Munawar Hasan , and Pranav Jain

Dept. of Computer Science and Engineering, Indraprastha Institute of Information Technology Delhi, India

{donghoon, munawar1440, pranav16255}@iiitd.ac.in

Abstract

In this paper, we present selfish mining attack on the multi-stage blockchain proposed by Palash Sarkar. We provide detailed analysis of computational wastage of honest miners and biased rewards achieved by the selfish pool. In our analysis, we introduce a spy inside an honest pool which is a trivial task. Our spy is responsible for leaking the information of the stage mining from the honest pool to the selfish pool. In our analysis, we consider all the possible configurations of mining namely sequential, parallel and pipelining. In all of these configurations, we show through our mathematical equations as to how a selfish miner can succeed in wasting the computation power of the honest miner and how he can influence the reward of mining. For completeness, we provide an algorithm for performing a selfish mining attack on all the scenarios on multi-stage blockchain.

To thwart selfish mining on multi-stage blockchain we redesign the original verification algorithm by introducing a new parameter called the crypto-stamp. We present a new algorithm that uses crypto-stamp during the verification process of the mined stages or blocks and is able to detect with high probability whether the stages or blocks were kept private or not.

1 Introduction

Bitcoin [1] is a cryptocurrency which has gained a lot of popularity over the last few years. The driving force behind the bitcoin is the blockchain which provides immutability over the stored data. Bitcoin network is based on *trustless*, *permissionless* and *decentralized* environment. Mining is the process of approving any transaction in the bitcoin network. Miners are required to solve a crypto puzzle which is considered computationally hard to solve. Further, if the majority of the miners are honest, it is practically impossible for the dishonest miners

to solve the crypto puzzle before the honest miners can mine; often referred to as *51%* rule. Such mining approaches are commonly called *proof of work*. The miners gets reward upon successfully solving the crypto puzzle. To solve a crypto puzzle, it requires a lot of computational power and hence is not feasible for any individual to mine in solitary, instead joining a mining pool is useful to increase the chances of the reward.

Eyal et. al [2] showed that majority rule is not enough to reward only honest miners. He showed that collusion is possible in Bitcoin's decentralized network leading to loss of reward and wastage of computational power for honest miners. The idea is to keep mined block in the private pool rather than releasing it to the public network. This way the honest miners will continue to mine for the block that are public while the miners on the private pool will keep mining on their private block as the head. When this private chain is strategically released to public network, it will lead to forks in the public chain and power wastage for the honest miner. Further, depending on the strategy of the release of the private chain, its miners can also have unfair advantage in rewards. This phenomenon is referred as *selfish mining*. There by different researchers have shown results related to selfish mining in several scenarios and several platforms [4, 5, 6, 7].

Recently *Palash Sarkar* in [3] presented a completely new blockchain architecture by splitting a single blockchain into several stages. This provided several degree of parallelism inside a single block of the blockchain. *Palash Sarkar* also claim that this strategy also removes the possibility of the *selfish mining*. In this paper, we will show that *selfish mining* is still possible using a malicious user called spy.

Mining pools use a system that is based on shares which are blocks with proof of works for a different difficulty than the network difficulty. Mining pools will set the difficulty for a given miner to be something that is reasonable for it to achieve. So when that miner mines, it is looking for a block hash that meets the pool difficulty. Once it does, it submits the block. That submitted block, although likely not valid to the Bitcoin network, meets the pool difficulty and is known as a share.
So mining pools will count the shares and give weight to each share based upon the difficulty that it was mined at. In this way, the mining pool can determine how much work each miner has done on average and pay them accordingly once a block is actually found.
Blocks are found when a share meets both the pool difficulty and network difficulty.

Our contribution is based on the Multi-Stage Blockchain architecture proposed by *Palash Sarkar* [3]. We first provide an in depth analysis of the selfish mining attack on Multi-Stage Blockchain using a spy in section 5. A spy is a miner which joins a mining pool and leaks some information to other mining

pool. We then introduce a new parameter called the *crypto-stamp*, using which we design the multi-stage blockchain in a way that will prevent *selfish mining*.

2 Preliminaries

In this section, we provide the notions and definitions needed to describe and analyze the multi-stage blockchain in light of spy.

The honest miners are the miners that work in a fixed pattern and follow the blockchain's protocol. The selfish miners work in a way which leads to wastage of computational power of honest miners. Let the pool of honest miners be called as the honest pool, and the pool of selfish miners be called as the selfish pool. A spy is a malicious user inside the honest pool which is responsible for leaking information of stage mining to the selfish pool. In this paper, we consider only one honest and one selfish pool to be working in the blockchain network.

We say that a block/stage is known to the network if the proof of work of the block/stage is known and can be verified by anyone in the network. A block/stage is said to be mined if its proof of work is verified and valid. A block/stage is said to be kept private if it is mined but not known to the network. Revealing a stage/block means making known a previously private stage/block. We will explain the meaning of blocks, stage and proof of work in detail in section 3.

3 Multi-Stage Blockchain

A blockchain is a distributed ledger which stores immutable records [1] in form of blocks. Each block consists of data over which immutability is required. *Palash Sarkar* introduced a new design rationale for blockchain called the multi-stage blockchain [3] where he introduced the concept of stages inside the block such that a block is divided into k stages where $k \geq 1$. If $k = 1$, there is no distinction between a stage and a block. Bitcoin [1], a decentralized crypto-currency is based on a blockchain with $k = 1$. From here on we will concentrate on multi-stage blockchain. Moreover single-stage blockchain is a special case of multi-stage blockchain and would inherit the idea provided in this paper.

In multi-stage design approach, each of the k stages inside the block has to perform proof of work rather than just a single proof of work per block as in the case of the blockchain design of the bitcoin. Similarly, the verification process in multi-stage design is also split into k parts where each part is verified separately.

3.1 Architecture

In this section, we describe the overall architecture of the multi-stage blockchain. We will be using similar notation as described in [3].

3.1.1 Blockchain

Denoting the i^{th} block of the multi-stage blockchain by B_i, the multi-stage blockchain is of the form:

$$B_0 \leftarrow B_1 \leftarrow ... \leftarrow B_{k-1} \leftarrow B_k \leftarrow B_{k+1} \leftarrow ...$$

where $B_i \leftarrow B_{i+1}$ means that B_{i+1} is dependent on B_i and B_{i+1} cannot be mined if B_i is not already mined.
Blocks B_0, B_1,...,B_{k-1} are the genesis blocks while the blocks B_k, B_{k+1},... are general blocks. The blockchain starts growing only after the genesis blocks are present.

3.1.2 Block

We will denote B_n as the n^{th} block of the blockchain. Thus B_n is a general block if $n \geq k$ and a genesis block if $n < k$.
In the subsequent subsections, we will explain the general and genesis blocks in detail.

3.1.2.1 General Block

A general block $B_n (n \geq k)$ consists of the following information:

B_n

n
$bdigest_n$
$\mathcal{L}_n$
$t_{n,0}$, $\eta_{n,0}$, $\tau_{n,0}$, $a_{n,0}$, $c_{n,0}$
$t_{n,1}$, $\eta_{n,1}$, $\tau_{n,1}$, $a_{n,1}$, $c_{n,1}$
.
.
.
$t_{n,k-1}$, $\eta_{n,k-1}$, $\tau_{n,k-1}$, $a_{n,k-1}$, $c_{n,k-1}$

where,

- n is the block number.
- $bdigest_n$ is the n^{th} block's digest.
- $\mathcal{L}_n$ is the list of transactions in the n^{th} block.
- for the n^{th} block and $0 \leq j \leq k-1$,

- $t_{n,j}$ is the target for stage j.
- $\eta_{n,j}$ is the nonce corresponding to the proof-of-work for stage j.
- $\tau_{n,j}$ is the timestamp for the completion (when the block is revealed to the network) of stage j.
- $a_{n,j}$ is the address to which the reward for completing stage j is to be assigned.
- $c_{n,j}$ is the reward given to $a_{n,j}$ for completing stage j.

The j^{th} stage of a general block B_n is defined as the tuple
($t_{n,j}$, $\eta_{n,j}$, $\tau_{n,j}$, $a_{n,j}$, $c_{n,j}$).
Let s_j^n be the j^{th} stage ($0 \leq j < k$) of n^{th} block B_n ($n \geq k$). Then, a general block B_n can be interpreted as a block consisting of $s_0^n, s_1^n, ..., s_{k-1}^n$.

3.1.2.2 Genesis Block

For the blockchain network to start, the initial k blocks need to be defined.
A genesis block $B_n (n < k)$ consists of the following information:

B_n

n
$bdigest_n$
t_n, η_n, τ_n, a_n, c_n

The block digests are defined as follows

$$bdigest_0 = H_0(0, t_0, a_0, c_0, \tau_0, \eta_0)$$

$$bdigest_i = H_i(bdigest_{n-1}, n, t_n, a_n, c_n, \tau_n, \eta_n)$$
$$\forall \quad 0 < n < k$$

The verification condition is

$$bdigest_n < t_n$$

3.1.3 Proof of Work

Let $\{H_0, H_1, ..., H_{k-1}\}$ denote a set of predefined hash functions for each of the k stages.
Let $RH(\mathcal{L}_n)$ be a root hash tree of the list of transactions.

The proof of work of various stages are defined as follows,

$$g_{n,0} = H_0(bdigest_{n-k}, n, RH(\mathcal{L}_n), t_{n,0}, a_{n,0}, c_{n,0}, \tau_{n,0}, \eta_{n,0}$$

$$g_{n,j} = H_j(bdigest_{n-k+j}, g_{n,j-1}, t_{n,j}, a_{n,j}, c_{n,j}, \tau_{n,j}, \eta_{n,j}$$
$$\forall \quad 0 < j < k$$

Finally, $bdigest_n$ is set equal to $g_{n,k-1}$.

Verification conditions for the proof of work of the different stages of general block $B_n (n \geq k)$ are:

$$g_{n,j} < t_{n,j}$$
$$\forall \quad 0 \leq j \leq k-1$$

4 Block Mining in Multi-Stage Blockchain

We assume that the first k blocks (genesis blocks) B_0,B_1,...,B_{k-1} have been mined and added to the network. The general blocks $B_n (n \geq k)$ are mined and added after the k genesis blocks.
Every block has k stages. In order to mine a block, the miners need to complete proof of work of all the k stages. As every j^{th} stage depends on $(j-1)^{th}$ stage $(0 < j < k)$, the block is mined as soon as proof of work of last stage is completed.

A stage s is said to be dependent on another stage/block if the proof of work of s cannot be completed without knowing the hash output/digest of the stage/block on which s is dependent.
Let s_j^n be the j^{th} stage $(0 < j < k)$ of n^{th} block B_n $(n \geq k)$. Then, according to the structure of the general blocks, s_j^n $(0 < j < k)$ depends on:

- $j - 1^{th}$ stage of block B_n (s_{j-1}^n)
- $n - k^{th}$ block B_{n-k}

while s_0^n (first stage of n^{th} block) depends only on:

- $n - k^{th}$ block B_{n-k}

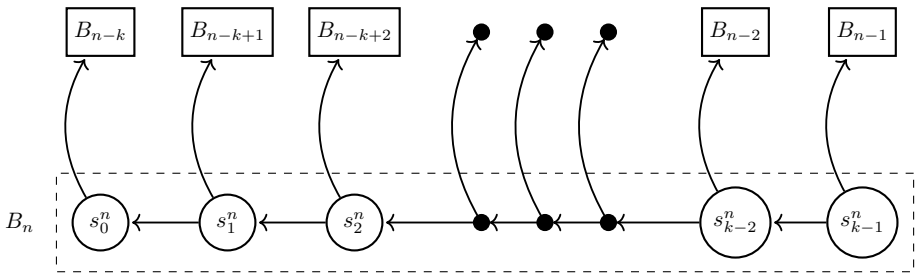


Figure 1: The figure shows the dependency of every stage of block B_n.
$M \leftarrow N$ depicts that N is dependent on M.

In the next section, we are going to explain different possible ways in which miners could work in the multi-stage blockchain.

4.1 Ways of Mining

Multi-stage blockchain supports multi-block mining which means that a miner can start working on the stages of a block even if the previous block is not completely mined (proof of work of all stages of previous block are not completed) as the first stage of a block B_n depends on the digest of the block B_{n-k} and not the previous block.

The following are the different ways of mining that the honest pool can follow.

4.1.1 Honest Pool Withholding Stages

An honest miner M can try to mine on the stages privately and reveal the block only when it has mined all the stages.
As mentioned in [3], if an honest miner keeps the mined stages privately, it runs the risk that the other miners can together mine the block earlier. Then, the entire work done by M in obtaining proofs of works of various stages will be wasted and will result in no return.
Therefore, no miner would try to mine the stages privately. It will immediately reveal the stage of which it has completed the proof of work.

4.1.2 Sequential Mining

In this way of mining, all the miners of the honest pool work on a single stage of a single block at a time and do not move to the next block without completing the previous block. Suppose that the miners start working on block B_n. Then, all the miners will mine on a single stage and move to the next stage only when the previous stage is mined. The miners will move to block B_{n+1} once the last stage of block B_n is mined. Therefore, the honest pool needs to complete proof of work of all stages of the block B_n in order to move to the next block B_{n+1}. Figure 2 illustrates an example of how the honest miners will mine.

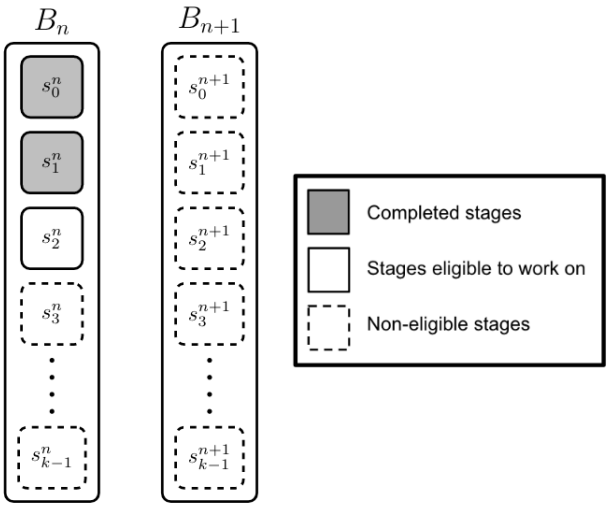


Figure 2: An example to show the eligible stages for the miners to work on in case of sequential mining.

As every block has k stages, the average time required to mine a block is equal to kT_{seq}, where T_{seq} is the average time to mine a stage in case of sequential mining.

4.1.3 Parallel Mining

In parallel mining, the honest pool divides itself into groups to work on different blocks simultaneously. Every different group starts working on a different block and follows sequential mining in their own respective block.
Figure 3 illustrates an example of how the honest pool will mine.

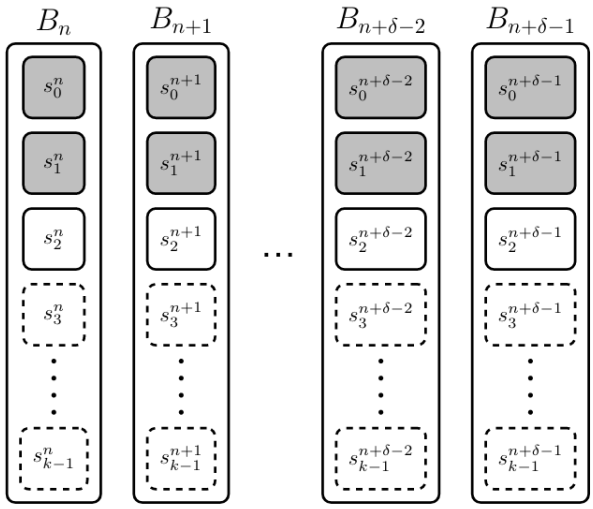


Figure 3: The figure shows the blocks on which miners mine simultaneously by dividing them into groups.

Suppose that the honest pool divides themselves into δ groups. Then, in time kT_{par}, δ blocks are mined where T_{par} is the average time taken by each group of honest pool to mine a stage.

4.1.4 Pipelined Mining

In case of pipelined mining, the miners work on multiple stages simultaneously. Suppose that the miners complete the proof of work of j^{th} stage of general block B_n $(0 \leq j \leq k-2)$. Then, the miners divides themselves to mine on the following:

- $(j+1)^{th}$ stage of block B_n
- j^{th} stage of block B_{n+1}

That is, some miners start working on the next stage (s_{j+1}^n) of the same block (B_n) while some miners start working on the same stage (s_j^{n+1}) of the next block (B_{n+1}).
Figure 4 illustrates an example of how the honest miners will mine.

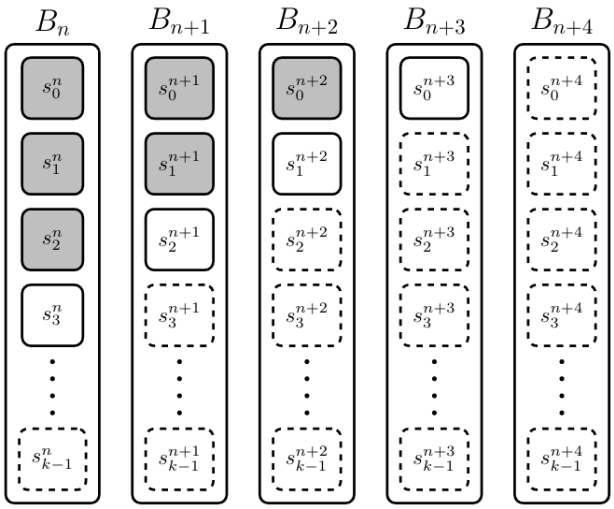


Figure 4: An example to show the eligible stages for the miners to work on in case of pipelined mining.

Suppose proof of work of each stage in pipelined mining requires about T_{pipe} seconds to be completed. Therefore, the time required to completely mine the first general block (B_k) is kT_{pipe}. But once a stage is completed, the honest pool divides into two groups and one group of miners work on the next stage, while the other group of miners shift to the same stage of the next block. Thus, if block B_n ($n \geq k$) is mined in kT_{pipe} seconds, block B_{i+1} will be mined in $(k+1)T_{pipe}$ seconds. Consequently, every subsequent block is mined in T_{pipe} seconds which makes the network very efficient.

4.1.5 Random Mining

The parameters of a stage are defined as the inputs that are required to compute the hash of the stage to complete the proof of work. For example the parameters of stage s_j^n is the tuple ($t_{n,j}$, $\eta_{n,j}$, $\tau_{n,j}$, $a_{n,j}$, $c_{n,j}$)
In case of random mining, the honest pool can start working on any stage whose parameters are known. There is no fixed way of mining.
Figure 5 illustrates one of the many ways of how the honest pool can mine.

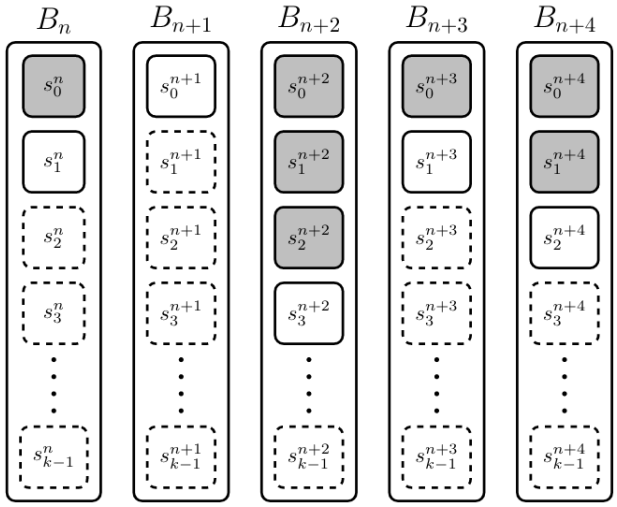


Figure 5: An example to show the eligible stages for the miners to work on in case of random mining.

The time required to mine a block is not constant in this case. It depends on the stages the honest pool chooses to work on.

5 Analysis of Multi-Stage Blockchain

In this section, we provide an in depth analysis of multi-stage blockchain in light of selfish strategy. We explain the attack scenario that can be applied on each and every way of mining discussed in section 3. We also do a mathematical analysis to prove that the selfish strategy will in fact result in a profit for the selfish pool.

5.1 Notations

Let α be the percentage of total hash power controlled by the selfish pool and β be the percentage of total hash power controlled by the honest pool.
Considering only one honest and one selfish pool,

$$\beta = 1 - \alpha$$

Let λ be defined such that $\beta = \lambda \cdot \alpha$. Hence, λ is the ratio of hash power controlled by honest pool and selfish pool.
In our analysis, we assume that the reward for mining a stage is 1.
Let the total revenue obtained by the honest pool be denoted by r_{honest}, while

the total revenue obtained by the selfish pool be denoted by $r_{selfish}$.
The relative revenue of the selfish pool is defined as the ratio of the revenue that selfish pool obtains and the total revenue obtained by all the pools working on the blockchain network. Considering only one selfish and one honest pool, the relative revenue of selfish pool is

$$R_{selfish} = \frac{r_{selfish}}{r_{selfish} + r_{honest}}$$

5.2 Possible Selfish Mining Attacks

In the Bitcoin protocol, the honest pool mines on the longest branch in the blockchain system and publishes a single block at a time. The selfish miners in a pool emphasize on generating more revenue than its ratio of mining power through creation of fork in the blockchain. The main focus of the selfish pool is to force the honest pool to waste its computational power on the rotten blocks in the public branch, where the rotten blocks are the blocks that are known to the network but are not a part of the longest branch.
In the selfish mining strategy adopted by Eyal & Sirer [2], the selfish pool generates the blocks and keeps them private, thus creating a private branch. The selfish pool publishes its blocks parallel to the publishing of blocks by the honest pool, so that it can result in the creation of a fork leading to the switching of honest pool onto the selfish mined block. At the point of creation a fork, there can be two possibilities; the first possibility being the addition of new block in front of honest pool's block invalidating the selfish pools lead and the second being mining of the second block in front of selfish pool's block which makes the selfish pool's block part of the longer branch.

5.2.1 Infiltration of Honest Pool

The first step of our proposed selfish mining attack is to place a selfish miner in the honest pool. As in general, there is no restriction on joining a public pool, this is feasible. We call the selfish miner present in the honest pool as a spy. As all mining pools always reveal the whole block and not the intermediate stages, the spy's job is to leak out information regarding the mining of the individual stages. It lets the selfish pool know which stage of which block is the honest pool currently working on.

5.2.2 Attack on Sequential Mining

Let the time taken by the honest pool to mine a stage be T. Then, the selfish pool mines a stage in time λT. Therefore, for every k stages mined by honest pool, the selfish pool mines $\left\lfloor \frac{k}{\lambda} \right\rfloor$ stages.

We now describe our strategy; Algorithm 1. The selfish pool follows Algorithm 1 in order to maximize their profit in case of sequential mining.

Algorithm 1: Selfish Strategy for Sequential Mining (Cases 1 and 2)

on Init
    public chain $\longleftarrow$ publicly known blocks
    private chain $\longleftarrow$ publicly known blocks
    $n \longleftarrow$ block number of the block whose stage 0 is currently getting mined by honest pool
    $\lambda \longleftarrow$ the ratio of hash power controlled by honest pool and selfish pool.
    $k \longleftarrow$ number of stages in a block

if $\lambda < k$ **then** // Case 1
    $\lambda' = \lambda$
    on honest pool reveals stage $s_{k-2}^{n+t\lambda}$ for some t inside the honest pool // Known by the spy
        reveal block $B_{n+t\lambda}$ in the public chain
        remove stage $s_i^{n+t\lambda}$ from private chain $\forall\ 0 \leq i \leq k-1$
    while *True* **do**
        $k' = 0$
        while $k' \leq (k-1)$ **do**
            if $k' \neq 0$ **then**
                $s_{k'}^{n+\lambda'} = mine(s_{k'-1}^{n+\lambda'}, B_{n+\lambda'-k+k'})$// Mine using previous stage and k^{th} previous block
            else
                $s_0^{n+\lambda'} = mine(B_{n+\lambda'-k})$// Mine using k^{th} previous block
            append $s_{k'}^{n+\lambda'}$ to private chain
            $k' = k' + 1$
        $\lambda' = \lambda' + \lambda$
else // Case 2
    $\xi = \lfloor k(k-1)/\lambda \rfloor$
    on honest pool reveals stage $s_{\xi-2}^{n+t(k-1)}$ for some t in honest pool // Known by the spy
        reveal ξ stages of block $B_{n+t(k-1)}$ // with the help of spy
        remove $s_i^{n+t(k-1)}$ from private chain $\forall\ 0 \leq i \leq \xi - 1$
    while *True* **do**
        $k' = 0$
        while $k' \leq (\xi - 1)$ **do**
            if $k' \neq 0$ **then**
                $s_{k'}^{n+k-1} = mine(s_{k'-1}^{n+k-1}, B_{n+k'-1})$// Mine using previous stage and k^{th} previous block
            else
                $s_0^{n+k-1} = mine(B_{n-1})$// Mine using k^{th} previous block
            append $s_{k'}^{n+k-1}$ to private chain
            $k' = k' + 1$
        $n \longleftarrow$ block number of the block currently getting mined by honest pool

Public chain consists of stages that are mined and known to the blockchain network while private chain consists of stages that are mined but are kept private by the selfish pool. In Algorithm 1, initially, the private and the public chain are the same.
We will divide our analysis into two cases, where the selfish pool's strategy will be different in both the cases.

Case 1: $\lambda < k$

Whenever, the honest pool works on block $B_n (n \geq k)$, the selfish pool will work on block $B_{n+\lceil\lambda\rceil}$. As $\lambda < k$, the selfish pool can work on block $B_{n+\lceil\lambda\rceil}$ because the digest of block $B_{n+\lceil\lambda\rceil-k}$ is available.
When honest pool completely mines λ blocks, the selfish pool will be able to mine the whole $B_{n+\lceil\lambda\rceil}$ block and the selfish pool keeps the block private. When the honest pool starts mining on block $B_{n+\lceil\lambda\rceil}$, the selfish pool starts mining on block $B_{n+2\lceil\lambda\rceil}$.
When honest pool completes $k-1^{th}$ stage (second last stage) of block $B_{n+\lceil\lambda\rceil}$, the spy reveals all the stages of block $B_{n+\lceil\lambda\rceil}$, thus wasting honest pool's computation on block $B_{n+\lceil\lambda\rceil}$.

Case 2: $\lambda \geq k$

In this strategy, the selfish pool knows that it cannot outrun honest pool since $\lambda > k$. Therefore, selfish pool plans to mine a fixed number of stages of a block rather than mining whole block.
When honest pool start working on block B_n, the selfish pool will start mining the first stage of block B_{n+k-1}. The selfish pool will keep a threshold ξ on number of stages of block B_{n+k-1} it plans to mine. Once ξ stages are mined, the selfish pool will move to block B_{n+2k-2} and again start mining ξ stages. On an average selfish pool can waste computations as well as consume reward of ξ stages of the honest pool on the chosen block.

For every one stage mined by the selfish pool, the honest pool would have mined λ stages. Therefore, for every ξ stages mined by the selfish pool, honest pool would have mined $\xi \cdot \lambda$ stages. The selfish pool needs to make sure that the honest pool do not outrun the selfish pool. Therefore,

$$\frac{\xi\lambda}{k} < k-1$$

$$\implies \xi < \frac{k(k-1)}{\lambda}$$

In order to maximize the profit, ξ is chosen to be,

$$\xi = \left\lfloor \frac{k(k-1)}{\lambda} \right\rfloor$$

5.2.2.1 Revenue

Theorem 1. *Let α be the hash power of the selfish pool and let k be the number of stages in a block.*
If $\lambda < k$, then algorithm 1 is profitable for selfish pool in case of sequential mining if

$$\frac{1}{2} > \alpha > \frac{1}{k+1}$$

If $\lambda \geq k$, then algorithm 1 is profitable for selfish pool in case of sequential mining if

$$\frac{1}{2} > \alpha > \frac{1}{k(k-1)+1}$$

Proof.
Case 1: $\lambda < k$
For every λ blocks mined by the honest pool, the selfish pool mines a single block and keeps it private. Once the honest pool mines $k-1$ stages of the block held private by the selfish pool, the spy reveals all the stages of the block wasting the honest pool's computation on the $k-1$ stages.
Let the reward of mining every stage is equal to 1. As the honest pool mines λ blocks, each consisting of k stages,

$$r_{honest} = k\lambda \cdot t \text{ for some } t \in \mathbb{N}$$

The selfish pool reveals a single block in the meantime. By the time the honest pool works on $k-1$ stages of the block held private by the selfish pool, selfish pool would have mined $\dfrac{k-1}{\lambda}$ stages of other block. Thus,

$$r_{selfish} = (k + \frac{k-1}{\lambda}) \cdot t \text{ for some } t \in \mathbb{N}$$

The relative revenue of selfish pool is:

$$\begin{aligned} R_{selfish} = \frac{r_{selfish}}{r_{selfish} + r_{honest}} &= \frac{k + \dfrac{k-1}{\lambda}}{k + \dfrac{k-1}{\lambda} + k\lambda} \\ &= \frac{k(\lambda+1) - 1}{k\lambda^2 + k(\lambda+1) - 1} \end{aligned}$$

When the selfish pool's relative revenue is greater than its size; α, then the attack is successful. Therefore,

$$\frac{k(\lambda+1) - 1}{k\lambda^2 + k(\lambda+1) - 1} > \alpha$$

Putting $\lambda = \dfrac{1-\alpha}{\alpha}$ and solving, we get,

$$k > 1$$

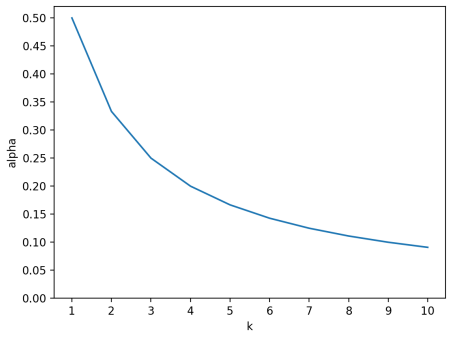
As $\lambda < k$,

$$\frac{1-\alpha}{\alpha} < k$$

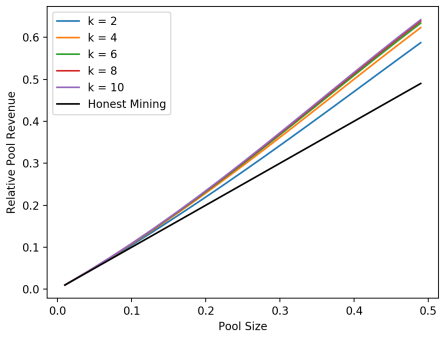
$$\implies \alpha > \frac{1}{k+1}$$

As the selfish pool's size cannot be greater than 50% of the network's size,

$$\frac{1}{2} > \alpha > \frac{1}{k+1}$$



(a) The figure shows how α is dependent on the k, the number of stages in a block.



(b) Pool revenue using the attack on sequential mining for different k

Figure 6: Plots showing the results of analysis on attack on sequential mining (Case 1)

Figure 6(a) shows the region in which α should lie so that Algorithm 1 ($\lambda < k$) is profitable. If α is above the curve, then the attack is profitable. Figure 6(b) shows that algorithm 1 for case where $\lambda < k$ is profitable for selfish pool and the profit increases with an increase in the value of k.

Case 2: $\lambda \geq k$

If honest pool is mining on block B_n, the selfish pool mines on block B_{n+k-1}. When the honest pool completely mines block B_{n+k-2}, the selfish pool would have mined $\xi(\xi < k)$ stages of block B_{n+k-1} and kept them private. The spy present in the honest pool reveals the stages as soon the honest pool mines $(\xi - 1)^{th}$ stage of block B_{n+k-1}.
As honest pool mines $k-1$ blocks each consisting of k stages,

$$r_{honest} = k(k-1) \cdot t \text{ for some } t \in \mathbb{N}$$

The selfish pool reveals ξ stages and by the time the honest pool works on $\xi - 1$ stages of the block, selfish pool would have mined $\frac{\xi - 1}{\lambda}$ stages of other block. Thus,

$$r_{selfish} = (\xi + \frac{\xi - 1}{\lambda}) \cdot t \text{ for some } t \in \mathbb{N}$$

The relative revenue of selfish pool is:

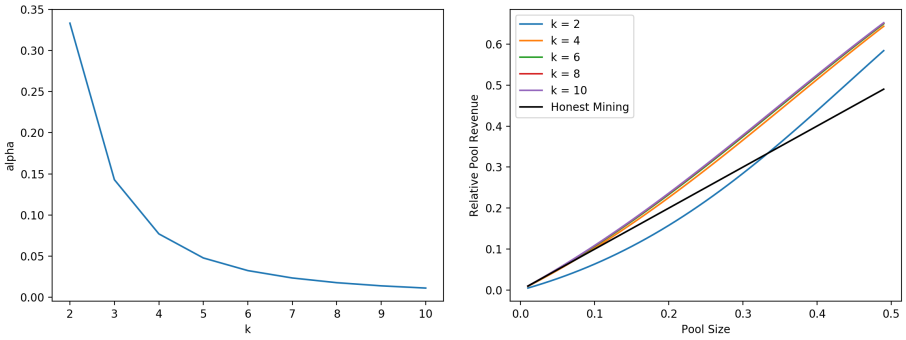
$$\begin{aligned} R_{selfish} = \frac{r_{selfish}}{r_{selfish} + r_{honest}} &= \frac{\xi + \frac{\xi - 1}{\lambda}}{k(k-1) + \xi + \frac{\xi - 1}{\lambda}} \\ &= \frac{\xi(\lambda + 1) - 1}{k\lambda(k-1) + \xi(\lambda + 1) - 1} \end{aligned}$$

When the selfish pool's relative revenue is greater than α, then the attack is successful. Therefore,

$$\frac{\xi(\lambda + 1) - 1}{k\lambda(k-1) + \xi(\lambda + 1) - 1} > \alpha$$

Putting $\lambda = \frac{1 - \alpha}{\alpha}$, $\xi = \left\lfloor \frac{k(k-1)}{\lambda} \right\rfloor$ and solving, we get,

$$\frac{1}{2} > \alpha > \frac{1}{k(k-1) + 1}$$



(a) The figure shows how α is dependent on the k

(b) Pool revenue using the attack on sequential mining for different k

Figure 7: Plots showing the results of analysis on attack on sequential mining (Case 2)

Figure 7(a) shows the region in which α should lie so that Algorithm 1 ($\lambda < k$) is profitable. If α is above the curve, then the attack is profitable.

Figure 7(b) shows that algorithm 1 for case where $\lambda > k$ is profitable for selfish pool of size greater than 0.33 for all values of k. □

5.2.3 Attack on Pipelined Mining

Let the time taken by the honest pool to mine a stage be T. Then, the selfish pool mines a stage in time λT. Therefore, for every k stages mined by honest pool, the selfish pool mines $\left\lfloor \frac{k}{\lambda} \right\rfloor$ stages.

Case 1: $\lambda < k$
The selfish strategy for pipelined mining ($\lambda < k$) is described in algorithm 2.

Algorithm 2: Selfish Strategy for Pipelined Mining ($\lambda < k$)

```
on Init
    public chain ⟵ publicly known blocks
    private chain ⟵ publicly known blocks
    n ⟵ block number of the block whose last stage is currently getting mined by honest pool
    λ ⟵ the ratio of hash power controlled by honest pool and selfish pool.
    k ⟵ number of stages in a block

if λ < k then
    on honest pool reveal stage s_{k-2}^{n+t(k-1)} for some t in honest pool
     // Known by the spy
        reveal block B_{n+t(k-1)} in the public chain
        remove stage s_i^{n+t(k-1)} from private chain ∀ 0 ≤ i ≤ k − 1
    while True do
        k' = 0
        while k' ≤ (k − 1) do
            if k' ≠ 0 then
                s_{k'}^{n+k-1} = mine(s_{k'-1}^{n+k-1}, B_{n+k'-1})// Mine using previous stage and k^th previous block
            else
                s_0^{n+k-1} = mine(B_{n-1})// Mine using k^th previous block
            append s_{k'}^{n+k-1} to private chain
            k' = k' + 1
        n ⟵ block number of the block currently getting mined by honest pool
```

We consider an assumption in order to ease our analysis.
Assumption: The honest pool distributes equally on all possible stages where the miners could work at a particular point of time.

Consider block B_n ($n \geq k$). All the honest pool works on first stage of block B_n (s_0^n). Once the first stage is completed, the honest pool divide themselves equally into 2 groups to work on second stage of block B_n (s_1^n) and first stage of block B_{n+1} (s_0^{n+1}).
As, each group consists equal number of miners, both the stages will be mined in the same amount of time.
After (s_1^n) and (s_0^{n+1}) are mined, the honest pool divides into 3 equal groups to work on (s_2^n), (s_1^{n+1}), (s_0^{n+2}).

Similarly, when honest pool finish its work on the $n+k-1^{th}$ block, the mining power of honest pool working on k^{th} stage of block B_{n+k} is $\frac{\beta}{k}$.

Once the block B_n is mined, the honest pool has $\frac{\beta}{k}$ power unused and $\frac{\beta}{k+1}$ of the honest pool are working on stage k of block B_{n+1}. Also, the miners are divided to work on $k+1$ stages of different blocks. Therefore, the extra power of honest pool will be divided into $k+1$ stages.
The mining power of honest pool working on the k^{th} stage of block B_{n+1} is

$$\frac{\beta}{k+1} + \frac{\beta}{k(k+1)} = \frac{\beta}{k}$$

Therefore, always $\frac{\beta}{k}$ of the honest pool are working on the last stage of a block at any point of time.
The selfish pool is able to mine a stage ahead of honest pool with high probability if

$$\alpha > \frac{\beta}{k}$$
$$\implies \lambda < k$$

Case 2: $\lambda > k$
The selfish strategy for pipelined mining ($\lambda > k$) is described in algorithm 3.

Algorithm 3: Selfish Strategy for Pipelined Mining ($\lambda > k$)

```
1  on Init
2      public chain ⟵ publicly known blocks
3      private chain ⟵ publicly known blocks
4      n ⟵ block number of the block whose first stage is currently getting
       mined by honest pool
5      λ ⟵ the ratio of hash power controlled by honest pool and selfish
       pool.
6      k ⟵ number of stages in a block
7      privateBranchLen ⟵ 0
8      StageNumber ⟵ 0
9
10 on honest pool mines a stage // Known by the spy
11     StageNumber ← StageNumber + 1
12     φ ← length(private chain) - length(public chain)
13     append new stage to private chain
14     privateBranchLen ← privateBranchLen + 1
15     if φ = 0 and privateBranchLen = 2 then
16         publish all of private chain
17         privateBranchLen ← 0
18
19 on selfish pool mines a stage
20     StageNumber ← StageNumber + 1
21     φ ← length(private chain) - length(public chain)
22     append new stage to private chain
23     if φ = 0 then
24         private chain ← public chain
25         privateBranchLen ← 0
26     else if φ = 1 then
27         publish last block of the private chain
28     else if φ = 2 then
29         publish all of the private chain
30         privateBranchLen ← 0
31     else
32         publish first unpublished block in private block
33 StageNumber ← 0
```

In this case ($\lambda > k$), the selfish pool cannot mine ahead of honest pool as by the time selfish pool could mine a stage of a block, the honest pool would have already mined it.
Therefore, in this case, the selfish pool tries to apply a strategy similar to *Eyal and Sirer's Selfish-Mine* strategy [2] but on stages of a single block.
In the selfish mining strategy, the selfish pool mines on stages and keeps them

private, thus creating a private branch. The spy present in the honest pool publishes its stages parallel to the stages mined by the honest pool, so that it can result in the creation of a fork leading to the switching of honest pool onto the selfish mined stage. At the point of creation a fork, there can be two possibilities; the first possibility being the creation of new stage by the honest pool invalidating the spys lead and the second being mining of the second stage by the spy and carry on with its lead against the honest pool.

5.2.3.1 Revenue

Theorem 2. *Let α be the hash power of the selfish pool and let k be the number of stages in a block.*
If $\lambda < k$,then Algorithm 2 is profitable for selfish pool in case of pipelined mining ($\lambda < k$) if

$$\frac{1}{2} > \alpha > \frac{1}{k+1}$$

Proof. Once a block is completely mined in case of pipelined mining, every subsequent block is mined in time T, where T is the average time to mine a stage. In time T, honest pool works on k stages simultaneously of k different blocks. As $\frac{\beta}{k}$ of the honest pool works on the last stage of a block at any point of time, by the time the honest pool completes mining one stage, selfish pool will mine $\frac{k}{\lambda}$ stages. Therefore, if honest pool mines λ stages, the selfish pool will mine k stages and completely mine a block. Also, as $\frac{\beta}{k}$ of the honest pool work on the last stage of a block, the honest pool mining λ stages is equivalent to say that the honest pool mines λ blocks.

If honest pool is mining on last stage of block B_n, then the selfish pool starts mining on block B_{n+k-1}. When the honest pool mines λ stages/blocks, the selfish pool completely mines the block B_{n-k+1} and keeps it private. Then, the selfish pool will move to mine first stage of block $B_{n+\lfloor\lambda\rfloor+k-1}$.
By the time the honest pool reaches block B_{n+k-1} (mines $k-\lambda$ blocks), the selfish pool would have mined $\frac{k(k-\lambda)}{\lambda}$. Once honest pool reaches last stage of block B_{n+k-1}, the spy reveals the block wasting the computation of honest pool.

As honest pool gets reward for $k-1$ blocks,

$$r_{honest} = k(k-1) \cdot t \text{ for some } t \in \mathbb{N}$$

As in the meantime, the selfish pool gets reward for the block B_{n+k-1} and $\frac{k(k-\lambda)}{\lambda}$ stages of block $B_{n+\lfloor\lambda\rfloor+k-1}$

$$r_{selfish} = (k + \frac{k(k-\lambda)}{\lambda}) \cdot t \text{ for some } t \in \mathbb{N}$$

The relative revenue of the selfish pool is:

$$R_{selfish} = \frac{r_{selfish}}{r_{selfish} + r_{honest}} = \frac{k + \frac{k(k-\lambda)}{\lambda}}{k + \frac{k(k-\lambda)}{\lambda} + k(k-1)}$$
$$= \frac{k}{k + \lambda(k-1)}$$

Putting $\lambda = \frac{1-\alpha}{\alpha}$, we get

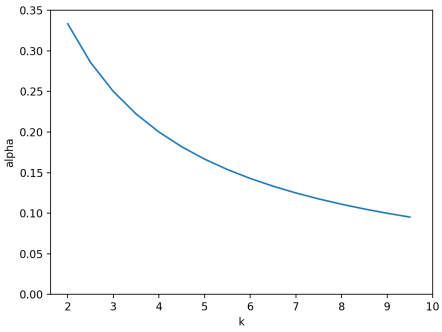
$$R_{selfish} = \frac{\alpha k}{k + \alpha - 1}$$

When the selfish pools relative revenue is greater than, then the attack is successful. Therefore,

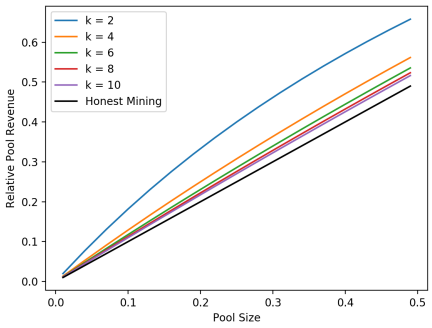
$$R_{selfish} = \frac{\alpha k}{k + \alpha - 1} > \alpha$$
$$\implies \alpha < 1$$

which is always true. Therefore, the attack is profitable if

$$\alpha > \frac{\beta}{k}$$
$$\implies \alpha > \frac{1}{k+1}$$



(a) The figure shows how α is dependent on the k, the number of stages in a block



(b) Pool revenue using the attack on pipelined mining for different k

Figure 8: Plots showing the results of analysis on attack on pipelined mining (Case 1)

Figure 8(a) shows the region in which α should lie so that Algorithm 2 is profitable. If α is above the curve, then the attack is profitable.

Figure 8(b) shows that algorithm 2 is profitable for the selfish pool and the profit decreases with an increase in the value of k. □

Theorem 3. *Let α be the hash power of the selfish pool and let γ be the percentage of number of miners of honest pool mining in front of the selfish pool's block in case of a fork. Then, Algorithm 3 is profitable for selfish pool in case of pipelined mining ($\lambda > k$) if*

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

Proof. Let us assume, H to be the honest pool and S to be a selfish pool; γ be the percentage of number of miners of H mining on selfish pool's block in case of a fork. We assume that value of γ remains same for every fork that takes place in the Blockchain network.
Figure 2 represents the progression of the system as a state machine, where the lead of the selfish pool is illustrated by the states of the system. Each number on the state diagrammatic representation shows the difference between the length of private and public branch.
In this state machine diagram, $0, 0'$ represents no forked branch and two-forked branch respectively.
When S mines a stage with the frequency α and state s = 0, 1, 2,..., the lead gets increased to (s+1); when the H mines a stage with frequency β and state s = 3, 4,..., the lead gets decreased by one to (s-1). If S has a lead of two and H mines a stage, S publishes its private branch and the system falls down from state 2 to state 0. The possible transitional frequencies have been shown in between the different states in the state machine diagram.

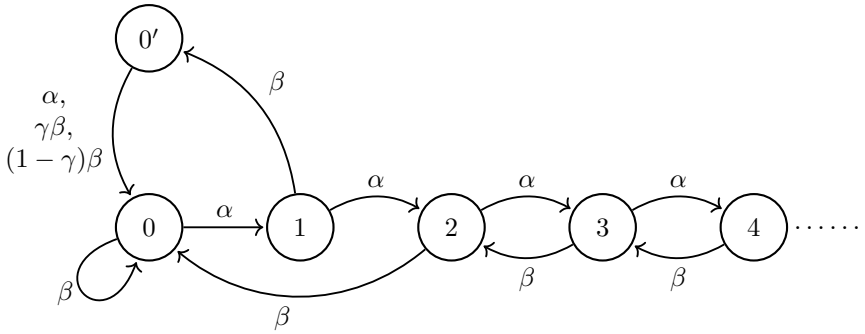


Figure 2: The diagrammatic representation of a state machine as a progression system

We can find out the state probabilities from the state machine diagram using

$$P[state = x] = P_x = \sum_k (P_k \times \omega)$$

where transition from state k with frequency ω goes to state x. The following equations are obtained by calculating the probability distribution from the state machine.

$$\begin{cases} P_0 = [\alpha + \beta\gamma + (1-\gamma)\beta]P_{0'} + \beta P_0 + \beta P_2 \\ P_1 = \alpha P_0 \\ P_{0'} = \beta P_1 \\ \forall k \geq j \geq 2, P_j = \alpha P_{j-1} + \beta P_{j+1} \\ \sum_{j=0}^{k}(P_j) + P_{0'} = 1 \end{cases} \tag{1}$$

After solving equation (1) , we obtain

$$P_0 = \frac{(1-2\alpha)}{2\alpha^3 - 4\alpha^2 + 1} \tag{2}$$

$$P_1 = \frac{\alpha(1-2\alpha)}{2\alpha^3 - 4\alpha^2 + 1} \tag{3}$$

$$P_{0'} = \frac{\beta\alpha(1-2\alpha)}{2\alpha^3 - 4\alpha^2 + 1} \tag{4}$$

$$P_j[j \geq 2] = (\frac{\alpha}{1-\alpha})^{j-1} \cdot \frac{\alpha(1-2\alpha)}{2\alpha^3 - 4\alpha^2 + 1} \tag{5}$$

We calculate the revenue of both selfish and honest pool using state probabilities and transitional frequencies,

$$r_{selfish} = 2 \cdot \alpha P_{0'} + 1 \cdot \gamma\beta P_{0'} + 2 \cdot \beta P_2 + 1 \cdot \beta P[i > 2] \tag{6}$$

$$r_{honest} = 1 \cdot \gamma\beta P_{0'} + 1 \cdot \beta P_0 + 2 \cdot (1-\gamma)\beta P_{0'} \tag{7}$$

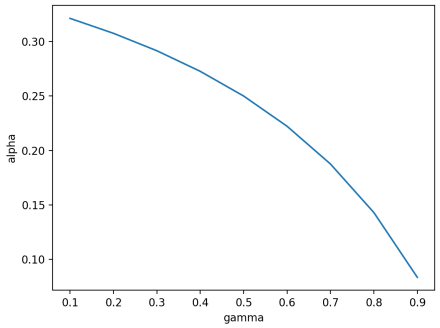
In order to calculate the selfish pools revenue, we substitute the probabilities from equation (2) to (5) in the revenue equation of (6) and (7).

$$\begin{aligned} R_{selfish} &= \frac{r_{selfish}}{r_{selfish} + r_{honest}} \\ &= \frac{\alpha(1-\alpha)(1-2\alpha)(2\alpha + \gamma(1-\alpha)) + 2\alpha^2(1-2\alpha) + \alpha^3}{\alpha^3 - 2\alpha^2 - \alpha + 1} \end{aligned} \tag{8}$$

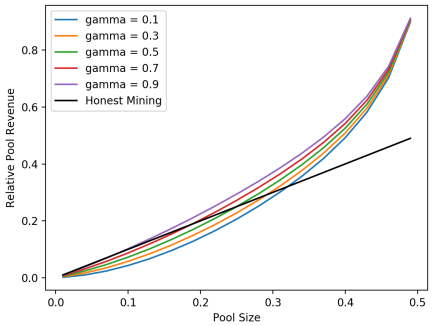
When the selfish pools relative revenue is greater than, then the attack is successful. Therefore,

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

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



(a) The figure shows how α is dependent on the k, the number of stages in a block



(b) Pool revenue using the attack on pipelined mining for different γ

Figure 9: Plots showing the results of analysis on attack on pipelined mining (Case 2)

Figure 9(a) shows the region in which α should lie so that Algorithm 3 is profitable. If α is above the curve, then the attack is profitable.
We observe that $R_{selfish}$ is independent of k. Figure 9 shows that algorithm 3 increases the profit of selfish pool with an increase in the value of γ. Also, higher value of γ helps a small sized selfish pool to enjoy the profit. □

The result we obtain for the selfish revenue is similar to the result obtained in [2] as the selfish attack proceeds similarly as [2] but on stages rather than blocks.

5.2.4 Attack on Parallel Mining

The selfish strategy for parallel mining is described in algorithm 4.

Algorithm 4: Parallel Approach

on Init
    public chain ⟵ publicly known blocks
    private chain ⟵ publicly known blocks
    n ⟵ block number of the block whose stage 0 is currently getting mined by honest pool
    δ ⟵ Number of blocks that the honest pool work on simultaneously
    k ⟵ number of stages in a block

if $\delta < k - 1$ **then**
    $\delta' = \delta$
    on honest pool reveals stage $s_{k-2}^{n+t\delta}$ for some t in honest pool `// Known by the spy`
        reveal block $B_{n+t\delta}$ in the public chain
        remove stage $s_i^{n+t\delta}$ from private chain $\forall\ 0 \leq i \leq k-1$
    while *True* **do**
        $k' = 0$
        while $k' \leq (k-1)$ **do**
            if $k' \neq 0$ **then**
                $s_{k'}^{n+\delta'} = mine(s_{k'-1}^{n+\delta'}, B_{n+\delta'-k+k'})$`// Mine using previous stage and` k^{th} `previous block`
            else
                $s_0^{n+\delta'} = mine(B_{n+\delta'-k})$`// Mine using` k^{th} `previous block`
            append $s_{k'}^{n+\delta'}$ to private chain
        $\delta' = \delta' + \delta$

Suppose, that the honest pool divides itself into δ groups to work on δ blocks at the same time.
Therefore, the amount of hash power of honest pool working on a stage of single block is $\frac{\beta}{\delta}$.

The selfish pool has an advantage in this case if

$$\alpha > \frac{\beta}{\delta}$$

$$\implies \lambda < \delta$$

$$\implies \alpha > \frac{1}{\delta + 1} \quad [\beta = 1 - \alpha]$$

If the honest pool works on blocks $B_n, B_{n+1}, ..., B_{n+\delta-1}$, the selfish pool will work on block $B_{n+\delta}$.
Therefore, for every δ blocks mined by the honest pool, the selfish pool mines 1

block and keeps it private. When the honest pool completes second last stage of block $B_{n+\delta}$, the selfish pool reveals the block wasting honest pool's computation.

If $\lambda > \delta$, then the selfish pool follows Algorithm 3 to generate a profit because it could never outrun the honest pool.

5.2.4.1 Revenue

Theorem 4. *Let α be the hash power of the selfish pool and let k be the number of stages in a block.*
If $\lambda < \delta$, then Algorithm 4 is profitable for the selfish pool in case of parallel mining if

$$\frac{2k}{(k+1)+\delta(2k-1)} > \alpha > \frac{1}{\delta+1}$$

Proof. Suppose $\alpha > \dfrac{1}{\delta+1}$. Then, the time in which honest pools mines $k-1$ stages of block $B_{n+\delta+1}$, the selfish pool might have completed another block. Therefore,

$$r_{honest} = (k\delta + (k-1)(\delta-1)) \cdot t \text{ for some } t \in \mathbb{N}$$

$$r_{selfish} = (k+k) \cdot t = (2k) \cdot t \text{ for some } t \in \mathbb{N}$$

. The relative revenue of selfish pool is:

$$\begin{aligned} R_{selfish} = \frac{r_{selfish}}{r_{selfish} + r_{honest}} &= \frac{2k}{2k + k\delta + (k-1)(\delta-1)} \\ &= \frac{2k}{(k+1)+\delta(2k-1)} \end{aligned}$$

When the selfish pool's relative revenue is greater than α, then the attack is profitable.

$$\frac{2k}{(k+1)+\delta(2k-1)} > \alpha > \frac{1}{\delta+1}$$

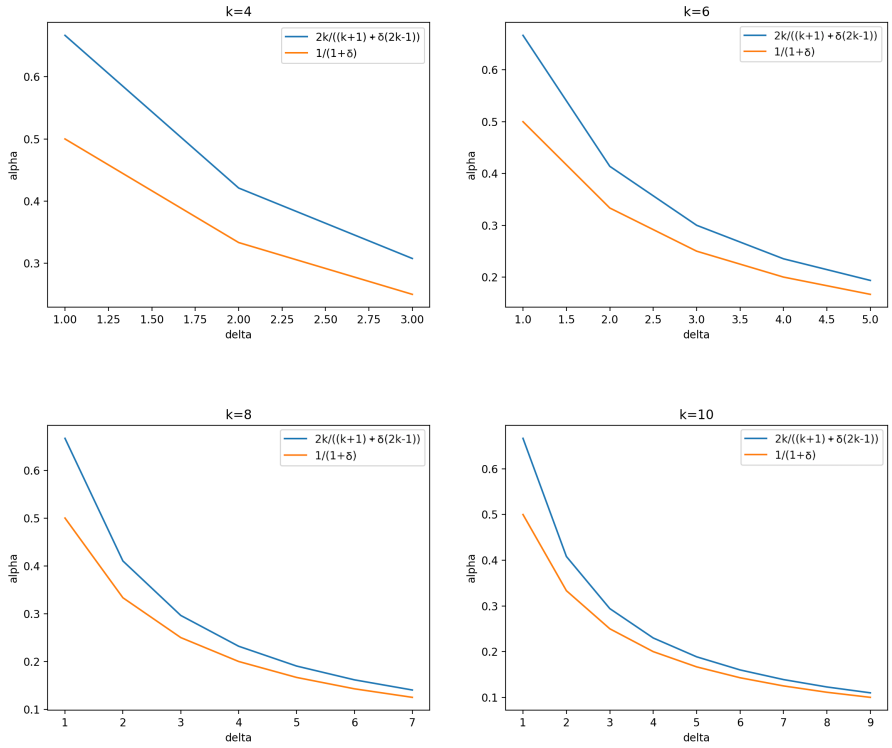


Figure 10: The figures shows the region of possible values of α for which the attack is profitable for different values of k.

□

From figure 10, we can observe that the feasible region for α becomes narrower as the value of k increases. Therefore, the selfish mining attack in case of parallel mining becomes less feasible as the number of stages of a block increases.

6 Crypto-Stamped Multi-Stage Blockchain

6.1 Architecture

From the above analysis, we can conclude that a selfish pool's strategy is to exploit the fact that there is no cross dependency on the timestamp of the block whose hash value is used to mine current stage of the current block. Hence, the selfish pool can store mined stages or blocks privately indefinitely. We introduce a tuple, crypto-stamp; which establishes a relationship between the timestamp of the current mined stage and the timestamp of the mined block whose hash

value is currently being used; thereby forcing each miner to release the mined stages as soon as possible else face the risk of getting their mined stages rejected by the network.
We have also updated the validation constraints so as to incorporate our new approach. In the subsequent sections we present the updated architecture of [3], such that no selfish mining is possible.

6.1.1 Blockchain

The blockchain is of the form:

$$B_0 \leftarrow B_1 \leftarrow ... \leftarrow B_{k-1} \leftarrow B_k \leftarrow B_{k+1} \leftarrow ...$$

where $B_i \leftarrow B_{i+1}$ means that B_{i+1} is dependent on B_i i.e B_{i+1} cannot be mined if B_i is not already mined.
Blocks B_0, B_1,...,B_{k-1} are genesis blocks while the blocks B_k, B_{k+1},... are general blocks.

6.1.2 General Block

We add an extra parameter to the general block data structure, called the crypto-stamp. The crypto-stamp is a tuple containing a hash value of block and the corresponding time stamp of the generation of that hash value. Hence, the updated block consists of the following information:

B_n

n
$bdigest_n$
$\mathcal{L}_n$
$t_{n,0}$, $\eta_{n,0}$, $\tau_{n,0}$, $a_{n,0}$, $c_{n,0}$, $(g'_{n,0}, \tau'_{n,0})$
$t_{n,1}$, $\eta_{n,1}$, $\tau_{n,1}$, $a_{n,1}$, $c_{n,1}$, $(g'_{n,1}, \tau'_{n,1})$
.
.
.
$t_{n,k-1}$, $\eta_{n,k-1}$, $\tau_{n,k-1}$, $a_{n,k-1}$, $c_{n,k-1}$, $(g'_{n,k-1}, \tau'_{n,k-1})$

where,

- n is the block number.
- $bdigest_n$ is the block digest.
- $\mathcal{L}_n$ is the list of transactions in the block.
- for n^{th} block and $0 \leq j \leq k-1$,
  - $t_{n,j}$ is the target for stage j.
  - $\eta_{n,j}$ is the nonce corresponding to the proof-of-work for stage j.

- $\tau_{n,j}$ is the timestamp for the completion (when the block is revealed to the network) of stage j.
- $a_{n,j}$ is the address to which the reward for completing stage j is to be assigned.
- $c_{n,j}$ is the reward given to $a_{n,j}$ for completing stage j.
- $g'_{n,j}$ is the hash value (random value) of the most recent stage mined. It is needed to mine stage j of block bn.
- $\tau'_{n,j}$ is the timestamp for the completion (when revealed to the network) of the most recent stage mined. It is needed to mine j^{th} stage of n^{th} block.

6.1.3 Genesis Block

There is no change in the structure of the genesis block. Therefore, for $0 \leq n \leq k-1$,

B_n

n
$bdigest_n$
t_n, η_n, τ_n, a_n, c_n

The block digests are defined as follows

$$bdigest_0 = H_0(0, t_0, a_0, c_0, \tau_0, \eta_0)$$

$$bdigest_n = H_i(bdigest_{n-1}, n, t_n, a_n, c_n, \tau_n, \eta_n)$$
$$\forall \quad 0 < n < k$$

The verification condition is

$$bdigest_n < t_n$$

6.1.4 Proof of Work

The proof of work of various stages are defined as follows,

$$g_{n,0} = H_0(bdigest_{n-k}, n, RH(\mathcal{L}_n), t_{n,0}, a_{n,0}, c_{n,0}, \tau_{n,0}, \eta_{n,0}, (g'_{n,0}, \tau'_{n,0}))$$

$$g_{n,j} = H_j(bdigest_{n-k+j}, g_{n,j-1}, t_{n,j}, a_{n,j}, c_{n,j}, \tau_{n,j}, \eta_{n,j}, (g'_{n,j}, \tau'_{n,j}))$$
$$\forall \quad 0 < j < k$$

Finally, $bdigest_n$ is set equal to $g_{n,k-1}$.

The verification conditions are:

$$\begin{cases} g_{n,j} < t_{n,j} \\ |\tau_{n,j} - \tau'_{n,j}| < T \\ (g'_{n,j}, \tau'_{n,j}) \text{ is a correct pair} \end{cases}$$

where $0 \leq j < k$ and T is the average time of completion of a stage.
A pair $(g'_{i+k,j}, \tau'_{i+k,j})$ is called correct if $g'_{i+k,j}$ is computed using $\tau'_{i+k,j}$ as the input.

6.1.5 Sub-Routine to Find Most Recent Stage

Before mining a stage, the miner needs to know the hash and the timestamp of the most recent stage mined. Let $\mathcal{T}$ be the time to find the most recent stage. As the most recent stage mined will be at the top of network, $\mathcal{T}$ is a very small value.
There are various block exploring services [22] which could be used to fetch the most recent stage.
Whenever a mining pool distributes work to the miners, it queries via the block exploring service and gives the information of the most recent stage along with the proof of work to the miners working in the pool. This way, the number of queries to the block exploring service will decrease drastically and therefore, there won't be any delay in obtaining the output from the service.

7 Defence Mechanism

In this section, we show how our proposed structure of multi-stage blockchain doesn't allow the selfish pool to proceed with selfish mining.

One of the verification condition of proof of work of a stage includes:

$$|\tau_j - \tau'_j| < T$$

where

- τ_j is the timestamp of the completion of proof of work of the stage.
- τ'_j is the timestamp of the completion of proof of work of the stage that is most recently revealed.
- T is the average time of mining a stage.

Hence, the selfish pool cannot withhold mined blocks and reveal them later. Because if he does, then the verification condition wouldn't hold true. Therefore, the proof of work will be discarded and the hash power of the selfish pool will be wasted.
Therefore, the selfish pool is forced to reveal the proof of work of a stage as soon as it mines it in order to get a reward for it. Thus, selfish mining is not possible.

8 Handling Forking

In the proposed Crpyto-Stamped Blockchain, to maintain the time of completion of one block to be ten minutes just like in Bitcoin, the time to complete each stage will be less than ten minutes. This increases the chance of fork within stages due to network latency. The following strategy could be adopted to resolve the forking and minimize network partitioning due to forking.

In case of a fork, we would have multiple digests (output of proof of work for each stage) which satisfy the verification conditions.
Suppose $g_1, g_2, ..., g_n$ be different hashes such that

$$g_i < t_{n,j} \text{ for all } i = 1...n$$

where $t_{n,j}$ is the target value of stage j of n^{th} block.
To resolve the fork, we choose $\min_{1 \leq i \leq n} g_i$ as the main branch and therefore the miners will choose $\min_{1 \leq i \leq n} g_i$ to mine the next stage. The remaining hashes are discarded.

The above strategy doesn't give an adversary any advantage as creating a fork wouldn't generally be in the adversary's favour. If the adversary tries to create a fork with a previously mined stage, then the proof of work is discarded due to failure of second verification condition. Therefore, an adversary cannot reveal an alternating long branch and expect it to be the main branch.

9 Conclusion

We have shown that how we can perform successful selfish mining attack on multi-stage blockchain by introducing a malicious user called the spy, inside an honest pool. The task of the spy is merely to leak the information regarding the stage completion in the honest pool. This leaked information is used by the selfish pool to waste computation power of honest pool and influence rewards. We proposed the selfish mining attack algorithms and analysed them on the three different types of mining strategies. Our results shows that selfish mining attack is still feasible on the Multi-Stage Blockchain [3] irrespective of the mining strategy the honest pool adopts. Hence, through our analysis we prove that multi-stage blockchain is not secure from selfish mining attack as claimed by the author.
We also proposed a modification in the Multi-Stage Blockchain by introducing an extra parameter to the general block data structure, called the crypto-stamp. The crypto-stamp prevents a pool/miner to keep blocks/stages private and this modification makes the selfish mining attack infeasible on Crpyto-Stamped Multi-Stage Blockchain.

References

[1] Satoshi Nakamoto. Bitcoin: A peer to peer electronic cash system `https://bitcoin.org/bitcoin.pdf`, 2009

[2] Ittay Eyal, Emin Gün Sirer. Majority is not Enough: Bitcoin Mining is Vulnerable, 2013

[3] Palash Sarkar. Multi-Stage Proof-of-work Blockchain, 2019

[4] Jianyu Niu, Chen Feng. Selfish Mining in Ethereum, 2019

[5] Fabian Ritz, Alf Zugenmaier. The Impact of Uncle Rewards on Selfish Mining in Etrhereum, 2018

[6] Qianlan Bai, Xinyan Zhou, Xing Wang, Yuedong Xu, Xin Wang, Qingsheng Kong. A Deep Dive into Blockchain Selfish Mining, 2018

[7] Muhammad Saad, Laurent Njilla, Charles Kamhoua, Aziz Mohaisen. Countering Selfish Mining in Blockchains, 2018

[8] Daniel Fullmer, A. S. Morse. Analysis of Difficulty Control in Bitcoin and Proof-of-Work Blockchains, 2018

[9] Alejandro Baldominos, Yago Saez. Coin.AI: A Proof-of-Useful-Work Scheme for Blockchain-based Distributed Deep Learning, 2019

[10] Vivek Bagaria, Sreeram Kannan, David Tse, Giulia Fanti, Pramod Viswanath. Deconstructing the Blockchain to Approach Physical Limits, 2018

[11] R. Bowden, H.P. Keeler, A.E. Krzesinski, P.G. Taylor. Block arrivals in the Bitcoin blockchain, 2018

[12] Ben Kaiser, Mireya Jurado, Alex Ledger. The Looming Threat of China: An Analysis of Chinese Influence on Bitcoin, 2018

[13] Nick Arnosti, S. Matthew Weinberg. Bitcoin: A Natural Oligopoly, 2018

[14] Nikolai Zaitsev. Empirical forward price distribution from Bitcoin option prices, 2019

[15] Istvn Andrs Seres, Lszl Gulys, Dniel A. Nagy, Pter Burcsi, Topological Analysis of Bitcoin's Lightning Network, 2019

[16] Andrew Chi-Chih Yao. An Incentive Analysis of some Bitcoin Fee Designs, 2018

[17] Evangelos Georgiadis, Doron Zeilberger. A Combinatorial-Probabilistic Analysis of Bitcoin Attacks, 2018

[18] Alexandre Bovet, Carlo Campajola, Jorge F. Lazo, Francesco Mottes, Iacopo Pozzana, Valerio Restocchi, Pietro Saggese, Nicol Vallarano, Tiziano Squartini, Claudio J. Tessone. Network-based indicators of Bitcoin bubbles, 2018

[19] Cuneyt Akcora, Matthew Dixon, Yulia Gel, Murat Kantarcioglu. Bitcoin Risk Modeling with Blockchain Graphs, 2018

[20] Dorit Ron, Adi Shamir. Quantitative Analysis of the Full Bitcoin Transaction Graph. 2013

[21] Simon Barber, Xavier Boyen, Elaine Shi, Ersin Uzun. Bitter to Better - How to Make Bitcoin a Better Currency, 2012

[22] Block Explorers
`https://www.blockchain.com/explorer`
`https://blockexplorer.com/`

[23] RISC Pipeline
`https://en.wikipedia.org/wiki/Classic_RISC_pipeline`