FORTIS: FORgeable TImeStamps Thwart Selfish Mining

Osman Biçer
Koç University
obicer17@ku.edu.tr

Alptekin Küpçü
Koç University
akupcu@ku.edu.tr

October 16, 2020

Abstract

Selfish mining (SM) attack (Eyal and Sirer, CACM '13) endangered Proof-of-Work blockchains by allowing a rational mining pool with a hash power (α) much less than 50% of the whole network to deviate from the honest mining algorithm and to steal from the fair shares of honest miners. Since then, the attack has been studied extensively in various settings, for understanding its interesting dynamics, optimizing it, and mitigating it.In this context, first, we propose generalized formulas for the calculation of revenue and profitability from SM-type attacks. Second, we propose two different SM-type attacks on the state-of-the-art mitigation algorithm "Freshness Preferred" (Heilman, FC '14). Our *Oracle mining* attack works on the setting with forgeable timestamps (i.e., if timestamps are not generated by an authority) and our *Bold mining* attack works on the setting with both forgeable or unforgeable timestamps (i.e., even if an authority issues timestamps). Although the use of timestamps would be promising for selfish mining mitigation, the analyses of our attacks show that Freshness Preferred is quite vulnerable in the presence of rational miners, as any rational miner with $\alpha > 0$ can directly benefit from our attacks. Third, we propose an SM mitigation algorithm Fortis with forgeable timestamps, which protects the honest miners' shares against any attacker with $\alpha < 27.0\%$ against all the known SM-type attacks.

Keywords: blockchain, selfish mining, bitcoin, proof-of-work.

1 Introduction

PoW Blockchain. First proposed by Nakamoto for Bitcoin [1], proof-of-work (PoW) blockchain plays a crucial role as the underlying technology behind modern cryptocurrencies and many applications (e.g., certificate transparency, smart contracts, e-government [2], smart appliances [3], and healthcare [4]). PoW blockchain depends on an incentive based mechanism called *mining* rather than a central authority for its proper operation. Mining is an unremitting competition for finding and propagating the hash value of the next block that appends the blockchain. *Miners* are investors on computational and bandwidth resources that are specialized for mining procedures. When a miner achieves this and so a block is mined, she receives a wealthy block reward, which builds up the incentive for her investment.

Selfish Mining (SM) Attack. Nakamoto claimed that as long as the majority of hashing power in the system belongs to the miners that follow honest mining algorithm, the reward of each miner would be proportional to her expenditures, securing the proper operation of the blockchain. Yet, Eyal and Sirer [5] proposed the *selfish mining (SM) attack* that yields a miner

more than her fair share by deviating from honest mining, even if the honest majority assumption holds. The main idea of the attack is keeping the mined blocks private for further extending them individually, and releasing them at a later time for elimination of others blocks from the main chain. The attack works because the honest miners always choose the longest chain (i.e., the one with more blocks) and the difficulty (the acceptable range set) of block hashes adjusts over time. In fact, none of these causes seem avoidable, as the former is a countermeasure against network partitions and propagation delays, and the latter is an initial design choice to ensure the expected inter-block time of the main chain would be constant. The selfish mining attack is extensively studied in the recent blockchain literature, including the research for optimizing it [6, 7, 8, 9] and defending against it [5, 10, 11, 12]. Although so far no known selfish mining attack occurred on Bitcoin, its practice would be harmful not only by reducing the fair shares of miners, but also by resulting in inconsistent views of blockchain among the users and by allowing double-spend exploits.

SM Defences. The initial defence mechanism proposed by [5] was that honest miners should choose one of the chains at random when there are multiple chains of the same length, while in the current Bitcoin implementation, miners choose the one that they received first. Regardless of the attacker's location or bandwidth, this defence prevented him from obtaining high rewards for computational powers below 25% hashing power (of the whole system).

A later proposal [10] by Heilman utilized timestamps issued by an authority to improve this resistance up to the computational power of 33.3%. The idea was that honest miners would pick the longest chain which has been mined more recently (i.e., with fresher timestamps) in case of a tie in chain lengths. As this solution was contradicting to the decentralized philosophy of blockchain, Heilman also claimed that removal of the timestamp authority would not reduce the protection against selfish mining, if a global synchronous clock functionality is available for use of all miners. However, this solution may be susceptible to an attack by setting the timestamp to the future.[1]

Our Attacks and Defence Proposal In this work, we propose two specialized rational attacks on timestamp based tie-breaking schemes of [10] (one is applicable on the unforgeable timestamp scheme and the other is applicable to both). We name these attacks as *Oracle mining* (i.e., mining on the last mined block in the main chain by setting the timestamp of the block being mined to a future point and then extending it privately) applicable on the one with forgeable timestamps and *Bold mining* (i.e., mining on a block in the main chain previous to the last mined block by setting the timestamp of the block being mined to the current time and then extending it privately) applicable on both of them. We also analyse our attacks via our generic formulas. Our attacks show that application of [10] even with the aid of a timestamp authority would allow any miner, regardless of his hashing power, to benefit from dishonest mining strategies. To solve these issues, we then propose Fortis, an honest mining algorithm based on forgeable timestamps (i.e., without a timestamp authority) that is not vulnerable to our proposed attacks or the state-of-the-art Optimal selfish mining attack [6] from an attacker with a computational power ratio up to 27.0%, as long as a global synchronous clock is available for all miners. Thereby, we achieve the highest security against a single rational miner capable of doing the selfish attacks known to date.

Our Contributions.

[1] [10] also identified an attack by setting the timestamp to the future from the current time, and named it as "slothful mining". To the best of our knowledge, this attack has neither been formally defined nor analysed so far.

1. In Section 4, we propose generic formulas that can be utilized for revenue calculations in selfish mining type of attacks. Our formulas makes complicated analysis of known attacks easier and more dependable by providing a systematical methodology.
2. In Section 5, we propose two separate attacks exploiting timestamp based solutions of [10], and show that there exists optimized choices of parameters in both attacks that yield a miner more than his fair share, even if his resources are very low. This means that any miner can conduct our attacks and damage the proper operation of the blockchain, if the schemes of [10] are deployed.
3. In Section 6, we propose our selfish mining mitigation algorithm FORTIS that defends against previous selfish mining attacks and our proposed ones, as long as the attacker's computational power ratio is less than 27.0% of the whole system, providing the best-known solution.

2 Preliminaries

Proof-Of-Work Blockchain. We briefly review proof-of-work (PoW) blockchain for our discussion and refer to [1, 13] for a comprehensive study. A blockchain is essentially an ever-growing chain of data blocks. Each block consists of a set of recent transactions, the height of the block (i.e, its index from the first block), the hash of its parent (e.g., SHA256 of the previous block), and a nonce. The hash of each block is required to be lower than a publicly determined value called "difficulty" (the current difficulty in Bitcoin is roughly 16×10^{12}). The blockchain miners keep competing to become the first one to mine[2] the next block. Whenever a miner mines a new block, she publishes it to the network. If the block is included in the main chain, she receives some block reward (the current block fixed reward in Bitcoin is 12.5 bitcoins). An honest miner always mines on (i.e., including the hash of) the head (the last block) of the longest branch (i.e., the chain that has the most blocks that is privy to her). If there exists a tie among the longest branches, various tie-breaking mechanisms can be involved (the current Bitcoin application suggests that each miner mines on the chain that she received the first). The blocks that do not end up in the main chain are not rewarded and are called as "orphan blocks". We highlight that each miner invests some computational and communication resources for mining coins. The aim for fairness in PoW blockchain is that the revenue of each miner is proportional to her investment [14]. If this is achieved, miners are better off with honest mining strategy rather than harmful ones to the system and the proper operation of the blockchain is secured.

Selfish Mining Type of Attacks. A major strike for blockchain security was invention of selfish mining (SM) attack by Eyal and Sirer [5]. The algorithm of the attacker briefly flows as follows. The attacker keeps his mined blocks as a private chain for further mining on them, and later releases his private chain, when the public chain approaches it in terms of length. This way, the attacker can eliminate the honest blocks from the main chain. The proposal shows that even if the honest majority assumption holds, a miner can obtain more revenue than honest mining via this mining strategy, since the difficulty adjusts automatically through time. Eventually what matters is the ratio of blocks being found by the attacker (the relative revenue) in the main chain, although block rewards and his resource investment remain the same.

[2]To mine a block means to find the nonce included in the block such that the hash of the block maps below the difficulty. The difficulty is frequently adjusted by the miners so that the expected inter-block time of the main chain would be constant (10 minutes in Bitcoin).

We call any type of attack involving a miner that keeps his mined blocks private to obtain high relative revenue as an SM-type attack. Some recent studies [6, 7, 8, 9] of selfish mining optimizes it further by allowing the attacker to mine on the private chain even when the public chain is longer. Other focuses of attention related to the SM attack are combining it with different attacks (e.g., with Eclipse attack [15, 8] and block witholding attack [16, 17]), conducting the attack on different currencies (e.g., on Ethereum [18]), and exploring the game theoretical implications of the attack [19, 20, 21, 22, 23].

We turn our focus on the particular subject of defending against the SM-type attacks [5, 10, 11]. We are interested in defences against the mining strategies generated by the state-of-the-art "Optimal selfish mining" (OSM) algorithm [6], as this also cover various choices of the selfish miner at each step. We do not go over the details of the algorithm or its relative revenue calculations here, and refer the reader to the original paper.

Selfish mining is harmful to the blockchain system, not only since it is stealing from the fair shares of honest miners, but also since it results in inconsistent views of blockchain among the users. A recent study [24] showed that other currencies than Bitcoin are far more susceptible to selfish mining, and to the best of our knowledge, differentiating this attack from a network partition is not fully possible yet. So far selfish mining seems to be an unavoidable burden for the blockchain community for being robust against network partitions [11].

Standard Model of Revenue Analysis for SM-type Attacks. The model that we utilize throughout the paper has been provided by the original SM paper [5]. Although analyses in different settings (e.g., without fixed block rewards [25, 26], in asynchronous setting [7, 27], in the presence of multiple attackers [20, 21, 22]) also exist, the previous attacks and defences are mostly analysed in this model. It mainly includes two players, Adam (the attacker) and Helen (the collection of the rest of the miners, assumed to be honest). Adam has a relative hashing power $\alpha < 0.5$ (i.e., the ratio of the number of hash operations by him over that all hash operations in a given time interval) and a relative network power γ (i.e., the expected ratio of the honest miners that will work on the selfish miner's chain in case of a tie). We note that in the current Bitcoin, γ depends on the bandwidth and location of the attacker's competitor blocks origin, as miners choose the branch that they receive first in case of a tie. We omit propagation delays and network partitions, therefore honest miners know whatever is public immediately. Honest miners may mine on different blocks due to forks. We assume a fully anonymized network, where detecting a malicious miner personally is impossible. We assume that computation times other than hashes and costs due to mining are 0. The finder of any block in the main chain is rewarded 1, unless it is stated otherwise. We omit the transaction fees from the reward for simplicity. Difficulty adjustment is quick enough that the revenue of attacker is equivalent to his relative revenue (i.e., the ratio of the blocks found by him over those found by all miners). We say that Adam benefits from an attack if his relative revenue from the attack is more than his benefit from honest mining. Otherwise we say that he does not benefit from an attack.

Notation for Mining Algorithms. In a miner's view, there exists 2 separate chains, PubCh (known by all miners) and MyCh (the chain chosen by the miner for mining on). PubCh gets updated automatically and might have forks of equal length, in which case a miner needs to choose which block to mine on.

- Append(Chain,b) denotes that "append the block b to the head of the Chain and other blocks between b and the head of the Chain.

- PUBLISH(b) denotes that "publish the block b".,
- PUBLISH(MYCH,z/head) denotes that "publish all the blocks until and including either z-th block of MYCH indexed from start of the fork from PUBCH (the first forked block is indexed as 1) or the head of PUBCH".
- TRUN(CHAIN,z) returns the chain obtained by truncating CHAIN by z blocks backwards from the head. LAST denotes the index of the last block found by the miner in PUBCH backwards from the head.
- $a \leftarrow b$ denotes that a is set as whatever b is at that moment.
- A $\rightarrow b$ denotes that "A finds the next block b". I denotes "The attacker's pool" and O denotes "Any pool other than attacker's one".
- MINE(CHAIN,FORK,z,TS) denotes that "mine on starting from z-th block backward indexed from the head (the index of the head is 1) of the FORK of the CHAIN, and set the timestamp of the currently mined block as TS". We highlight that an attacker always mines on the fork that includes more blocks found by him, therefore we omit the FORK input in attack algorithms. Also, if the fork resolving policy of the honest miners does not depend on timestamps, the input TS can be omitted.

3 Evaluation of SM Defence Algorithms

In this section, we describe the evaluation measures and the standard model for comparing SM defences, and then provide a comparison of the existing defences.

3.1 Evaluation Measures

Zhang and Preneel [11] propose some notions (decentralization, incentive compatibility, and backward compatibility) to be considered for an ideal defence against SM-type attacks. Here we modify them as measures of the quality of defences with minor changes, and provide further measures, i.e., robustness, inflation balance, and integrity. We note that decentralization, incentive compatibility, robustness and inflation-balance was already among the design goals in the invention of Bitcoin. To the best of our knowledge, this is the first work that take into account integrity as a quality measure of SM defence algorithms. We describe these measures as follows:

- **Decentralization.** Ideally, the system should be kept away from single points of failures as much as possible. In this work, we limit this to how many trusted authorities required and which opearations they are involved in.
- **Incentive compatibility.** Ideally, the expected revenue of each miner should be maximized by the honest mining strategy, and this revenue should be proportional to her computational power. As a measure of incentive compatibility, our work is mainly concerned with the highest bounds of computational and communication powers of a single attacker for which the honest mining strategy dominates an SM-type attack strategy.
- **Robustness.** Ideally, in case of forks due to recovery from a network partition or block propagation delays, the miners should reach a consensus immediately. This is measured

Algorithm 1 Freshness Preferred Honest Mining Algorithm [10]

```
while true do
    if PubCh has 1 branch then
        Mine(PubCh,1,τ)
    else
        Mine(PubCh,fresher branch,1,τ)
    end if
    Propagate PubCh
end while
```

as the expected number of blocks to be mined before the consensus, in case of a tie among separate forks.

- **Inflation-Balance.** As each block mined is also creation of some new coins, allowing this more than a constant rate (even in cases of partitions and attacks) is considered problematic. Another design goal of Bitcoin was to obtain a currency whose inflation rate is constant.

- **Backward Compatibility.** Ideally, a solution should be as compatible as possible with the existing clients who cannot upgrade for wide use. The realm of compatibility involves block validity rules (i.e., a block should be either valid for both old and new clients or neither of them) and reward distribution policy (only the blocks in the main chain will be rewarded as before). This measure is a simplified version of "backward compatibility" of [11] by removal of "eventual consensus".

- **Integrity.** Ideally, the solution should allow new miners to join and old miners to leave without the system security being further affected, as long as the honest miner ratio is kept above a fixed value. This is measured as the maximum hashing power ratio of honest miners that can be securely replaced with the new ones at a time best for the attacker. This measure also covers an existing miner's loss of memory due to an accident.

We note that further security metrics (i.e., "chain quality", "subversion gain", and "censorship susceptibility") for evaluation of selfish mining defences are provided in [28]. Here, we do not include chain quality measured by the ratio of the attacker's found blocks in the main chain, as it is directly correlated with incentive compatibility in fixed reward systems. Moreover, we do not comprise subversion gain and censorship susceptibility, as these are related to different attacks than SM-type ones, and the schemes considered here perform similar against those attacks.

3.2 Review of Existing Defenses

Here we briefly review the existing selfish mining defences in the literature, and compare them based on the the evaluation criteria that we have provided in Section 3.1.

Uniform Tie-Breaking (UT). The initial SM defence proposed by Eyal and Sirer was that an honest miner picks a branch uniformly at random among the longest chains in case of ties [5]. We call this defence as Uniform Tie-breaking (UT). Against the SM attack the defence achieves incentive compatibility up to $\alpha = 0.250$. However, the later proposal of OSM [6] reduced this to $\alpha = 0.232$.

Freshness Preferred (FP) with Unforgeable/Forgeable Time Stamps. Heilman proposes use of time stamps for tie-braking to reduce the relative revenue of a selfish miner [10]. In case of a tie, the miners choose the branch that is fresher (i.e., the branch whose time stamps (TS) are closer to the current time τ), hence the scheme is called "Freshness Preferred" (FP).

		UT	FPwUTS	FPwFTS	PP	Our Fortis
Decentralization		Full	TS auth.	Full	Full	Full
Incentive Compatibility	against OSM [6]	23.2%	30.1%	30.1%	25.0%	27.0%
	against our OM	$*$	0	0	$*$	27.0%
	against our BM	$*$	$*$	0	$*$	27.0%
	overall	23.2%	0	0	25.0%	27.0%
Robustness		1	0	0	18.5	1
Integrity		1	1	1	0.333	1

Table (1) Comparison of Uniform Tie-breaking (UT) [5], Freshness Preferred with unforgeable and forgeable timestamps (FTwUTS and FTwFTS, respectively) [10], Publish or Perish (PP) [11], and our Fortis scheme based on the evaluation criteria in Section 3.1. Regarding incentive compatibility, we consider three separate attacks, i.e., Optimal selfish mining (OSM) [6], our Oracle mining attack, and our Bold mining attack. TS auth. denotes timestamp authority. $*$ denotes that an attack is inapplicable to a scheme.

As the primary proposal, the author recommended support of a TS authority that generates unforgeable timestamps that will be embedded in the blocks to eliminate any forgery risks. However, considering that this would be an obstacle for decentralization, the author also argued that the authority may not be necessary in the presence of a global synchronous clock, as the timestamps cannot be changed once the block is mined. In this case each miner can just embed the current time into the block being mined. We call the former and the latter schemes as Freshness Preferred with unforgeable timestamps (FPwUTS) and Freshness Preferred with forgeable timestamps (FPwFTS), respectively. The honest miner's algorithm in both schemes only differ in the generation of the timestamps and is formalized in Algorithm 1.

The main idea of this scheme is that the blocks of the private chain of a selfish miner have to be mined earlier than their siblings (the ones with equal length) in the public chain. Therefore, by choosing the fresher chain in case of ties, the honest miner enforces the network power for the selfish miner as $\gamma = 0$. If the timestamps are generated in an infinitesimal fashion, the scheme increases incentive compatibility against SM to $\alpha = 0.333$ and against OSM to $\alpha = 0.301$.

Publish or Perish (PP) [11]. "Publish or Perish" (PP) scheme [11] provides incentive compatibility against OSM up to $\alpha = 0.250$. The proposal defines a "fail-safe parameter" (i.e., the minimum height difference required for enforced adoption of the longest branch), and claims that fail-safe parameter might be optimum as 3. Therefore, we will also compare the scheme with others based on this choice. We skip the further protocol details and refer the reader to [11].

Some other works to mention are GHOST [29], Bobtail [12], and [30, 31, 32], which, unfortunately, fail to satisfy backward incompatibility. Also, there exist some works [33, 34] for detecting the behavior of the selfish miner and eliminate his blocks from the main chain. However, this detection has not been reliably achieved yet, as there seems to be no clear way for differentiating it from network partitioning. So far, selfish mining seems to be an unavoidable burden for the blockchain community for being robust against network partitions.

To conclude our review, to the best of our knowledge, the selfish mining defence algorithm that scores the best with respect to the above-mentioned evaluation criteria is FPwUTS, as currently it provides decentralization, incentive compatibility up to $\alpha = 0.301$, robustness 0, high inflation balance, backward compatibility, and full integrity. Table 1 provides comparison of the existing schemes and our Fortis scheme that we will describe in Section 6. Only the systems that provide backward compatibility and high inflation balance are compared. We calculate the integrity of PP scheme as 33.3%. If more than this limit of honest miners are replaced all at once, a selfish miner with hashing power 25.0% can deceive the new comers to

obtain the majority of the hashing power. Also, the robustness value 18.5 is given by [11] setting fail-safe parameter as 3 (recommended by the authors).

4 Generic Formulas For SM-Type Attacks

The generic idea of selfish mining attack strategies is that the attacker wants to eliminate some of the already-found blocks of the honest majority from the main chain. Often the attacker continues with the honest mining, but sometimes when the attacking conditions occur, he deviates from the honest mining strategy. He might even risk some of his own already-found blocks (yet, we will see that this is not necessarily the case). We identify mainly two different states of the attacker: one where he is mining on a block on the public chain (we call this S_{pub}) and one where he is mining on his private chain (we call this S_{priv}). At a given time, the attacker will be at either S_{pub} or S_{priv} state. One can calculate the expected revenue of the attacker as

$$R = \frac{\alpha - f \cdot \ell_A}{1 - f \cdot (\ell_A + \ell_H)} \tag{1}$$

where ℓ_A, ℓ_H, and f denote the expected block loss of the attacker in S_{priv}, the expected block loss of the honest parties in S_{priv}, and the expected frequency of returning to S_{priv}, respectively. Since being at states S_{priv} and S_{pub} keep alternating, f can be calculated as

$$f = \frac{1}{ES_{priv} + ES_{pub}} \tag{2}$$

where ES_x is the expected number of all blocks mined by both parties secretly or publicly while being at state S_x. The intuition for the equation is that if the attacker only executes honest mining all the time during an expected period $ES_{priv} + ES_{pub}$, the expected number of blocks that he finds would be $\alpha(ES_{priv} + ES_{pub})$ (where α is the attacker's hashing power) and those that are found by all miners would be $ES_{priv} + ES_{pub}$. We obtain Eq. 1 by subtracting the corresponding expected losses from both, then dividing them by the period, and then calculating the ratio of the former to the latter.

Whether the attacking strategy is expected to benefit the attacker depends on the ratio of $\frac{\ell_A}{\ell_H}$. More concretely, iff the attacking strategy benefits the attacker, then we have

$$\frac{\ell_A}{\ell_A + \ell_H} < \alpha \quad \text{or equivalently} \quad \frac{\ell_A}{\ell_H} < \frac{\alpha}{1 - \alpha} \tag{3}$$

We name the ratio $\frac{\ell_A}{\ell_H}$ as profitability ratio B. We emphasize that sole knowledge of ℓ_A and ℓ_H is not enough to determine the attacker's revenue, but rather a tool for his decision-making in choosing between applying an SM-type mining strategy and honest mining. We note that setting $R > \alpha$ in Eq. 1 leads to Eq. 3.

To show that our formulas indeed work, we apply them to the basic selfish mining algorithm of [5]. Algorithm 2 provides the same SM algorithm given by [5], but it shows S_{priv} and S_{pub} states as distinct procedures.

An example flow of SM is shown in Figure 1. In S_{pub}, the adversary tries to mine a block. Whenever he mines it, instead of publishing, he keeps it secret and goes to S_{priv}. His block loss can only occur if an honest miner finds a block first in S_{priv}, and then $\gamma(1-\alpha)$ fraction of honest

Algorithm 2 Selfish Mining Attack Algorithm of [5]

```
procedure S_pub
    Set Δ ← 0
    while true do
        MyCh ← PubCh
        Mine(MyCh,1)
        if I → b then
            Append(MyCh,b)
            Set Δ ← 1
            Go to procedure S_priv
        end if
    end while
end procedure
```

```
procedure S_priv
    while 0 < Δ < k + 1 do
        Mine(MyCh,1)
        if I → b and Δ > 0 then
            Append(MyCh,b)
            Set Δ ← Δ + 1
        else if O → b and Δ ≤ 2 then
            Publish(MyCh,head)
            Go to procedure S_pub
        else if O → b and Δ > 2 then
            Publish(MyCh,1)
            Set Δ ← Δ − 1
        end if
    end while
end procedure
```

miners that work on Helen's block finds the next block. In this case, Adam loses 1 block. As the probability that this case occurs is $(1-\alpha)^2(1-\gamma)$, we obtain

$$\ell_{A,SM} = (1-\alpha)^2(1-\gamma)$$

Helen may lose blocks in S_{priv} mainly in two different ways: (1) Helen finds the first block, Adam publishes the head of his private chain, Adam's block wins the block race with probability $(1-\alpha)\big(\gamma(1-\alpha)+\alpha\big)$, Helen loses 1 block. (2) Adam finds the first block with probability α, eventually Helen catches up with Adam, Adam publishes his chain and Helen loses all the blocks she has found within S_{priv}. For the second outcome, we consider this as a "monkey at the cliff" problem [35], starting when Adam's chain is 2 blocks ahead of Helen's chain, going away from the cliff with probability α and towards it with probability $1-\alpha$. Therefore, the expected number of steps for this walk to the end is $\frac{1}{(1-\alpha)-\alpha} = \frac{1}{1-2\alpha}$ [35]. Since the expected steps towards the cliff should be 1 more than those away from it for the state to end, we calculate Helen's expected block loss in this case as $\frac{1}{2}\big(\frac{1}{1-2\alpha}+1\big) = \frac{1-\alpha}{1-2\alpha}$. At the end, we obtain

$$\ell_{H,SM} = (1-\alpha)\big(\gamma(1-\alpha)+\alpha\big) + \alpha \cdot \frac{1-\alpha}{1-2\alpha}$$

Regarding f_{SM}, we need to obtain ES_{priv} and ES_{pub}. Upon going to state S_{priv}, the probability that Helen finds the next block is $(1-\alpha)$, ending the state with 1 block. Upon going to state S_{priv}, the probability that Adam finds the next block is α, ending the state in expected $1+\frac{1}{1-2\alpha}$ steps. We calculate $ES_{priv} = (1-\alpha) + \alpha\big(1+\frac{1}{1-2\alpha}\big) = 1+\frac{\alpha}{1-2\alpha}$. At the beginning of S_{pub}, there may be a block race where either Adam's branch or Helen's one (the forks differ by only 1 block) will be the winner if Helen found the first block in the last S_{priv}. Thus, this case occurs with probability $1-\alpha$. After this, S_{pub} finishes when the attacker mines the next block requiring the expected number of $\frac{1}{\alpha}$ blocks since it is a geometric random variable. Hence, we calculate $ES_{pub} = 1-\alpha+\frac{1}{\alpha}$. From Eq. 2, we obtain that

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

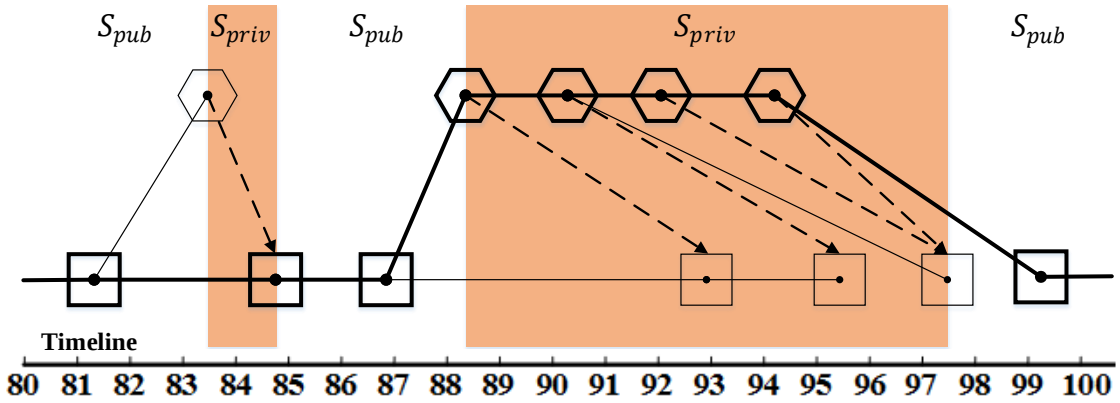

Figure (1) An example flow of Selfish mining attack. The squares are honest miners' blocks, while the hexagons are those of the attacker. Straight lines shows parent-child relation between two blocks. The thick lined blocks are the ones that remains in the main chain and gets rewarded, while the thin lined ones are orphan blocks. The dashed arrows point to the time when the attacker publishes the block. S_{pub} and S_{priv} states are shown by the areas colored as white and coral, respectively.

If we plug $\ell_{A,SM}$, $\ell_{H,SM}$, and f_{SM} into Eq. 1, we obtain

$$R_{SM} = \frac{\alpha - \left(\frac{(1-2\alpha)(\alpha)}{1-4\alpha^2+2\alpha^3}\right)\left((1-\alpha)^2(1-\gamma)\right)}{1 - \left(\frac{(1-2\alpha)(\alpha)}{1-4\alpha^2+2\alpha^3}\right)\left((1-\alpha)^2(1-\gamma) + (1-\alpha)\big(\gamma(1-\alpha)+\alpha\big) + \alpha \cdot \frac{1-\alpha}{1-2\alpha}\right)}$$

We omit the intermediary steps, but this equation is equivalent to the revenue equation of [5] as

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

Figure 2 shows the revenues of the attacker with respect to his hashing power for network powers $\gamma = 0$, $\gamma = 0.5$, and $\gamma = 1$ for both equations.

5 Attacks on Timestamp Based Defenses

Use of unforgeable timestamps with "freshness preferred" (FP) tie-breaking for SM mitigation was first proposed by Heilman [10]. It promised a good result for fighting with SM by setting $\gamma = 0$; thus a rational adversary with up to 33.3% of overall hashing power is better of with honest mining, which later reduced to 30.1% by [6]. [10] also considered the case of using forgeable timestamps (i.e., without an authority, but in the presence of a global clock functionality), but realized that this might result in "slothful mining" (i.e., setting the timestamp in a future time). The author argued that this would not cause any harm, as a timestamp will be committed, i.e., it cannot be changed after a block is found. In Section 5.2, we show that this claim is not true, as there exists optimized strategies that benefits a miner even with a very small hashing power (essentially for any $\alpha > 0$). Further, in Section 5.2, we show that there exist attacks on the scheme even if an authority issues the timestamps that benefits an attacker for any $\alpha > 0$.

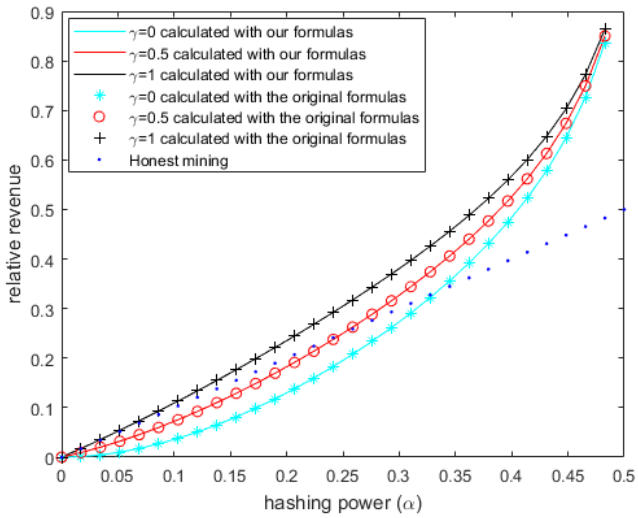

Figure (2) Revenue from selfish mining [5] with respect to hashing power α for network powers $\gamma = 0$, $\gamma = 0.5$, and $\gamma = 1$ calculated using our Eq. 1 and the original equation of [5] given at Eq. 4.

5.1 Oracle Mining Attack on FPwFTS of [10]

Algorithm. We now describe a rational algorithm based on setting the timestamp of the currently mined block to the future. At S_{pub} the attacker sets the timestamp $TS = \tau + t$ of the block he is mining on, where τ is the current time and $t > 0$ (we will later show how to set t for maximizing the revenue depending on α). If he succeeds, he will have a block that cannot be published immediately, but can be kept private for mining on until about the time $\tau + t$ (i.e., he switches to S_{priv}). He will increment the timestamps of the next blocks he finds at S_{priv} with the unit increment ϵ. If the honest miners outperform him by mining two blocks more than him at S_{priv}, the attacker's found blocks will be lost. Otherwise, he will kick all the blocks of honest miners within S_{priv} off the main chain. Although more sophisticated attacks where the attacker might swing between the honest and private chains are possible, and likely to benefit him more, we restrict his attacking strategy as keeping mining on his private chain until t, and publishing all the chain at the time t. We name this mining strategy as *Oracle Mining* (OM). Algorithm 3 provides full description of our OM attack, where ϵ, T_c, and t_m denote the incremental unit of timestamps, the current expected time interval between block found by any miner (differs from the public expected inter-block time by including the blocks that do not end up in the main chain), and the value t that maximizes the attacker's revenue, respectively. Also, an example flow of OM is shown in Figure 3, including example S_{pub} and S_{priv} states.

Analysis. For simplicity we let $\epsilon = 0$. We model the mining in a given time range as a Poisson random variable as in [6]. The success probability is given as $P[X = x] = \frac{e^{-\mu}\mu^x}{x!}$ where $X = \{0, 1, 2, 3 \ldots\}$ is a set of possible number of successes, e is the Euler's number, and μ is the mean number of successes (success rate) in a given time interval. We let $\mu_A = \frac{\alpha t}{T_c}$ and $\mu_H = \frac{(1-\alpha)t}{T_c}$ denote success rate of Adam and Helen, respectively.

We can state

$$\ell_{A,OM} = \sum_{h=2}^{\infty} \sum_{a=0}^{h-2} \big(P[A = a] \cdot P[H = h] \cdot (a + 1)\big)$$

$$\ell_{H,OM} = \sum_{a=0}^{\infty} \sum_{h=1}^{a+1} \big(P[A = a] \cdot P[H = h] \cdot h\big)$$

Algorithm 3 Our Oracle Mining Attack Algorithm

```
procedure S_pub
    Set Δ ← 0
    while true do
        Find t_m for T_c and α
        MyCh ← PubCh
        Mine(MyCh,τ + t_m)
        if I → b then
            Append(MyCh,b)
            Set t_a ← τ + t_m
            Set Δ ← 1
            Go to procedure S_priv
        end if
    end while
end procedure
```

```
procedure S_priv
    while τ < t_a + (Δ − 1) · ε do
        Mine(MyCh,1,t_a + Δ · ε)
        if I → b and Δ > 0 then
            Append(MyCh,b)
            Set Δ ← Δ + 1
        end if
    end while
    Publish(MyCh,head)
    Go to procedure S_pub
end procedure
```

Here $P[A = a]$ and $P[H = h]$ denote probabilities that Adam finds a blocks and Helen finds h blocks within S_{priv}, respectively. Since these are Poisson random variables, $P[A = a] = \frac{e^{-\mu_A}\mu_A^a}{a!}$ and $P[H = h] = \frac{e^{-\mu_H}\mu_H^h}{h!}$. As the time spent in S_{priv} is t, we calculate $ES_{priv} = \frac{t}{T_c}$. Also, $ES_{pub} = \frac{1}{\alpha}$, as S_{pub} finishes when Adam finds a block. Hence, from Eq. 2, we obtain

$$f_{OM} = \frac{\alpha T_c}{\alpha t + T_c}$$

Plugging $\ell_{A,OM}$, $\ell_{H,OM}$, and f_{OM} into Eq. 1, we obtain Adam's revenue as

$$R_{OM} = \frac{\alpha - \frac{\alpha T_c}{\alpha t + T_c} \cdot \sum_{h=2}^{\infty} \sum_{a=0}^{h-2} \left(\frac{e^{-\mu_A}\mu_A^a}{a!} \cdot \frac{e^{-\mu_H}\mu_H^h}{h!} \cdot (a+1) \right)}{1 - \frac{\alpha T_c}{\alpha t + T_c} \cdot \left(\sum_{h=2}^{\infty} \sum_{a=0}^{h-2} \left(\frac{e^{-\mu_A}\mu_A^a}{a!} \cdot \frac{e^{-\mu_H}\mu_H^h}{h!} \cdot (a+1) \right) + \sum_{a=0}^{\infty} \sum_{h=1}^{a+1} \left(\frac{e^{-\mu_A}\mu_A^a}{a!} \cdot \frac{e^{-\mu_H}\mu_H^h}{h!} \cdot h \right) \right)}$$

By setting t properly, the attacker can maximize his revenue from this attack and thus beat the honest mining strategy for any α

How to Set the optimum t_m. Intuitively, setting t as a large value does not make much sense for an attacker with $\alpha < 0.5$, since it gives the honest majority a larger time interval (and hence a higher chance) to beat him. From the analysis, we deduce that increasing the value of t increases both $\ell_{A,OM}$ and $\ell_{H,OM}$, and decreases f_{OM}. Figure 4 shows the revenue of the attacker with respect to α for optimum t_m/T_c. The attacker can keep setting the t_m by preparing a look up table for t_m/T_c and α from this figure or computing them by himself. We note that both α and T_c may vary over time, but they can be estimated by using the statistics of the all mined blocks (as the attacker knows the number of both public and private ones) and the difficulty.

Impact of the Attack. The most important implication of this attack is that for any α value, there exists a value $t > 0$ that provides more revenue than honest mining does. Even if the direct revenue gain from this attack does not seem as great as the previously known SM-type attacks, it is more likely to occur in practice in an FPwFTS application, since it may be attractive for any rational miner, while the latter usually requires large α or γ values. Its effects are similar to selfish mining: It exhausts honest miners, which can reduce their numbers, leading to a more vulnerable blockchain system to other SM-type attacks. It also results in inconsistency as a ledger of transactions.

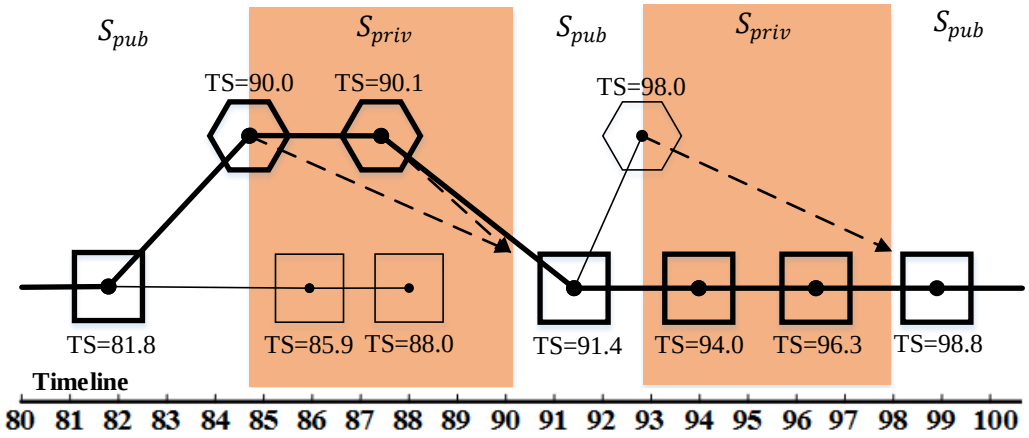

Figure (3) An example flow of Oracle mining attack. The squares are honest miners' blocks, while the hexagons are those of the attacker. Straight lines shows parent-child relation between two blocks. The thick lined blocks are the ones that remains in the main chain and gets rewarded, while the thin lined ones are orphan blocks. The dashed arrows point to the time when the attacker publishes the block. S_{pub} and S_{priv} states are shown by the areas colored as white and coral, respectively.

5.2 Bold Mining Attack on FPwUTS of [10]

Algorithm. The attacker's strategy is essentially to mine a sibling to the k-th past block from the current head at S_{pub} (if he does not own any block between the current head and the k-th past block, both inclusive), and then to keep it private (going to S_{priv}), and next to try to catch up with honest miners by mining on his found block. If the honest chain surpasses his chain (with $k+1$ blocks), he accepts defeat and goes back to S_{pub}. If his chain succeeds by having an equal number of blocks to the public chain, he publishes his chain and goes back to S_{pub} as the winner. We especially require the honest chain to be $k+1$ blocks ahead for the attacker's failure, since if the honest chain is k blocks ahead, the attacker is better off by following his private chain, since his private chain has more blocks belonging to him than the other one. Upon going to S_{pub} with success, to be flexible with the number k, we allow the attacker to choose a value K, such that K is the trail bound of the honest miners' blocks for k. An example flow of the BM attack is shown in Figure 5, including example S_{pub} and S_{priv} states. In the full analysis of this attack (available in the full version), we deduce that both $\forall \alpha \in (0, 0.432)\ \forall k > 1\ B_k \geq \frac{\alpha}{1-\alpha}$ and $\forall \alpha \in (0, 0.432)\ B_1 < \frac{\alpha}{1-\alpha}$. Therefore, we rationally set $k = 1$, and the attack simplifies to the attack in Algorithm 4,

Revenue Analysis For Optimized Bound $K = 1$. Clearly $\ell_{A,BM} = 0$, as there are no blocks lost by Adam. Similarly, $\ell_{H,BM} = 1$, since Helen will lose 1 block whenever attack condition occurs. Also $ES_{priv} = 0$, as Adam immediately releases his found block. To calculate ES_{pub}, upon going out of S_{priv}, first, Helen needs to find a block, whose average length is $\frac{1}{1-\alpha}$. Then, Adam needs to find a block, whose average length is $\frac{1}{\alpha}$. From Eq. 2, we obtain $f_{BM} = \frac{1}{\frac{1}{1-\alpha}+\frac{1}{\alpha}} = \alpha(1-\alpha)$. Applying Eq. 1, we deduce

$$R_{BM} = \frac{\alpha - \alpha(1-\alpha).0}{1-\alpha(1-\alpha)(0+1)} = \frac{\alpha}{1-\alpha(1-\alpha)}$$

Figure 6 shows the expected revenue outcome of this attack for various α values, clearly

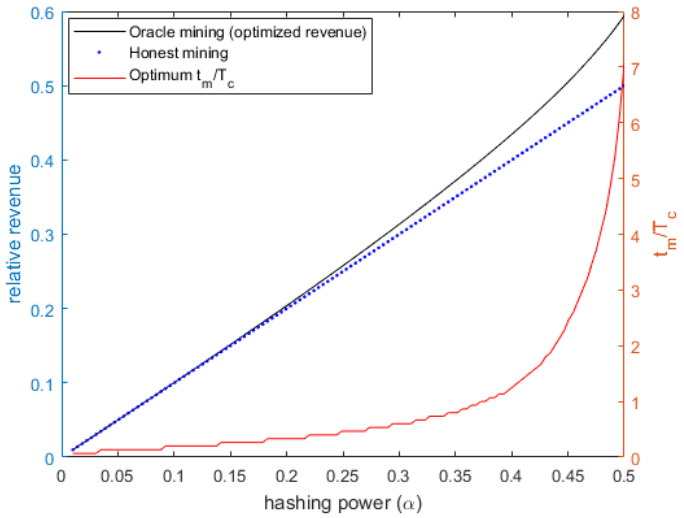

Figure (4) Revenue from Oracle mining w.r.t. the hashing power α when the timestamps set optimally. Also the optimum t_m/T_c values for all α of interest are provided.

Algorithm 4 Our Bold Mining Attack Algorithm as $k = 1$

procedure S_{pub}
  while true **do**
    Set $\Delta \leftarrow \min(\text{LAST}, 1) - 1$
    MYCH $\leftarrow$ TRUN(PUBCH,Δ)
    MINE(MYCH,1,τ)
    if I $\rightarrow b$ **and** $\Delta = 0$ **then**
      PUBLISH(MYCH,head)
    else if I $\rightarrow b$ **and** $\Delta = 1$ **then**
      APPEND(MYCH,b)
      Set $\Delta \leftarrow \Delta - 1$
      Go to procedure S_{priv}
    else if O $\rightarrow b$ **and** $\Delta = 0$ **then**
      Set $\Delta \leftarrow \Delta + 1$
    end if
  end while
end procedure

procedure S_{priv}
  PUBLISH(MYCH,head)
  Go to procedure S_{pub}
end procedure

demonstrating that it outperforms the honest mining strategy for any $\alpha < 0.5$.

Impacts of the Attack. The impact of this attack is similar to that of Oracle mining, but with increased severity. For any α value, the attacker's revenue surpass the one receivable from honest mining, therefore it is more likely to occur in practice in an FTwUTS or FTwFTS application than previously known SM-type attacks. It exhausts honest miners, which can reduce their numbers, leading to a more vulnerable blockchain system to other SM-type attacks. It also results in inconsistency as a ledger of transactions.

6 Our SM Mitigation Algorithm

We provide a tie-breaking strategy that if applied, an attacker with $\alpha < 27.0\%$ can benefit from none of the selfish mining type attacks oracle mining (OM), bold mining (BM), and optimized selfish mining (OSM) [6]. Our solution (we name it as FORTIS) is a relaxation of Freshness Preferred with forgeable timestamps (FPwFTS) [10], biasing some forks with respect to timestamps unlike Uniform Tie-breaking [5]. For the sake of proper operation of FORTIS, we assume that a global synchronous clock functionality is available to every party as in [10].

Intuition. Observe that both OM and BM are applicable to FP, since in case of ties the

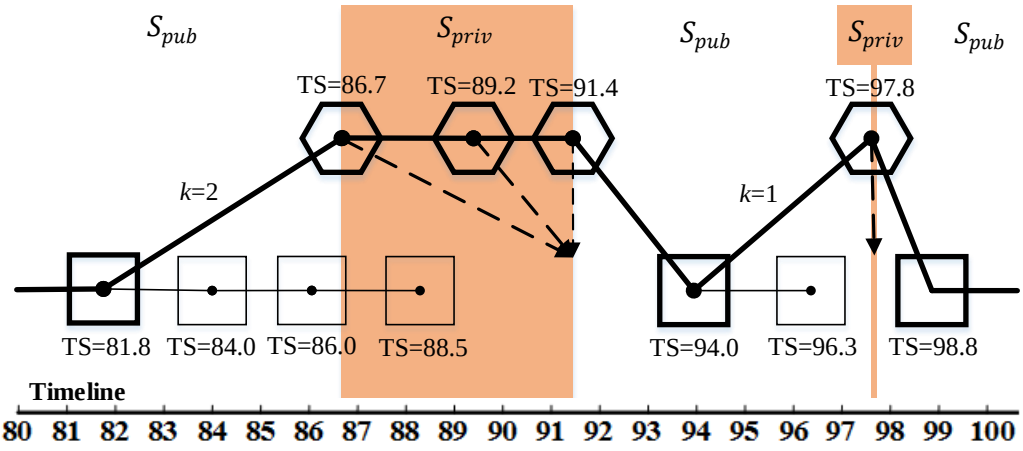

Figure (5) An example flow of Bold mining attack. The squares are honest miners' blocks, while the hexagons are those of the attacker. Straight lines shows parent-child relation between two blocks. The thick lined blocks are the ones that remains in the main chain and gets rewarded, while the thin lined ones are orphan blocks. The dashed arrows point to the time when the attacker publishes the block. S_{pub} and S_{priv} states are shown by the areas colored as white and coral, respectively. Note that in case $k = 1$, the attacker immediately releases the block in S_{priv}.

fresher block is the one that always wins. However, neither of them is possible against Uniform Tie-breaking (UT) where $\gamma = 0.5$ (yet it prevents OSM only for adversaries with hashing power less than 0.232): Regarding OM, the most likely outcome of S_{priv} is that only Helen would find one block within the time t. Thus, if $\gamma = 0.5$, Adam will risk his first found block in cases Helen also finds a block. Regarding BM, clearly, Adam is better off with honest mining, since the blocks he finds in BM will be counted only roughly half of the time, whereas Helen's block loss will be less in terms of ratio.

Let ψ denote the bias for the fresher chain (i.e., probability that an honest miner chooses the fresher chain in case there exist two branches with equal length). FP sets $\psi = 1$ for defending against SM. We can set this number to any value $0.5 \leq \psi < 1$, and still be better against OSM than UT that set it to $\psi = 0.5$. Decreasing ψ gives more chance to the blocks with lower timestamp, and we start to be susceptible to BM and OM. Therefore, we need an optimized value of ψ for a certain lowest bound of α that an attacker would benefit from any of OM, BM, and OSM. Let us reconsider the OM attack with the introduction of $\psi < 1$, i.e., including the case that upon going back to S_{pub} with a tie, the following occurs: Adam releases his blocks, and a block race takes place for the next block (Adam sets the timestamp of the block he is mining on as the current time). Then, the probabilities that Adam and Helen lose the mined blocks are $(1-\psi)(1-\alpha)$ and $\alpha + \psi(1-\alpha)$, respectively. Whoever finds the next block, Adam switches back to the honest chain, and continues with the attack algorithm the same way as in OM. We state the $\ell^{\psi}_{A,OM}$, $\ell^{\psi}_{H,OM}$, and f^{ψ}_{OM} (ES_{pub} of OM is elongated with 1 block) are as follows:

$$\ell^{\psi}_{A,OM} = \sum_{h=2}^{\infty}\sum_{a=0}^{h-2}\big(P[A=a]\cdot P[H=h]\cdot(a+1)\big) + (1-\psi)(1-\alpha)\sum_{h=1}^{\infty}\big(P[A=h-1]\cdot P[H=h]\cdot h\big)$$

$$\ell^{\psi}_{H,OM} = \sum_{a=0}^{\infty}\sum_{h=1}^{a}\big(P[A=a]\cdot P[H=h]\cdot h\big) + \big(\alpha+\psi(1-\alpha)\big)\sum_{h=1}^{\infty}\big(P[A=h-1]\cdot P[H=h]\cdot h\big)$$

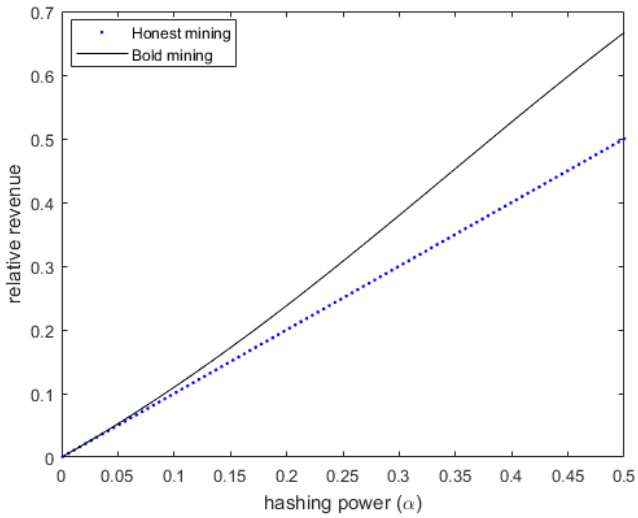

Figure (6) Revenue from Bold mining vs. revenue from honest mining with respect to α.

Algorithm 5 Our FORTIS Honest Mining Algorithm

```
while true do
    if PUBCH has 1 branch then
        MINE(PUBCH,1,τ)
    else if PUBCH has 2 branches then
        MINE(PUBCH,fresher branch,1,τ) with probability ϒ
    else if PUBCH has n branches s.t. n > 2 then
        MINE(PUBCH,i-th branch,1,τ) with probability 1/n
    end if
    Propagate PUBCH
end while
```

$$f_{OM}^{\psi} = \frac{\alpha T_c}{\alpha t + T_c + \alpha T_c}$$

Also, in BM, in case of a tie upon going back to S_{pub}, the above-mentioned additional procedure change again occurs. We state that

$$\ell_{A,BM}^{\psi} = (1-\psi)(1-\alpha), \quad \ell_{H,BM}^{\psi} = \alpha + \psi(1-\alpha), \quad f_{BM}^{\psi} = \frac{\alpha(1-\alpha)}{1+\alpha(1-\alpha)}$$

We have an optimization problem which can be stated as

$$\begin{aligned} \textit{Maximize } \alpha \textit{ subject to}: \; & R_{OM}^{\psi} < \alpha, \; R_{BM}^{\psi} < \alpha, \; R_{OSM}^{\psi} < \alpha, \\ & \alpha \in (0, 0.5), \; \psi \in (0.5, 1) \end{aligned}$$

For R_{OM}^{ψ} and R_{BM}^{ψ}, we utilize Eq. 1, and for R_{OSM}^{ψ} we utilize the implementation given in [36]. We obtain the solution for this problem is $\alpha \approx 0.270$ and occurs at $\psi \approx 0.630$. We denote the number 0.630 as Υ. We provide our FORTIS algorithm in Algorithm 5. We highlight that in case more than 2 longest branches exist, the honest miners pick the branch uniformly at random to mine on. Note that generating more than one longest branch is less beneficial for an attacker than extending only one chain; therefore, one of the longest branches can belong to the attacker[3]. Therefore, uniform picking results in less than $1-\Upsilon$ or Υ chance for the

[3]In case multiple attackers exist in the system, it is true that there may be multiple longest branches belonging

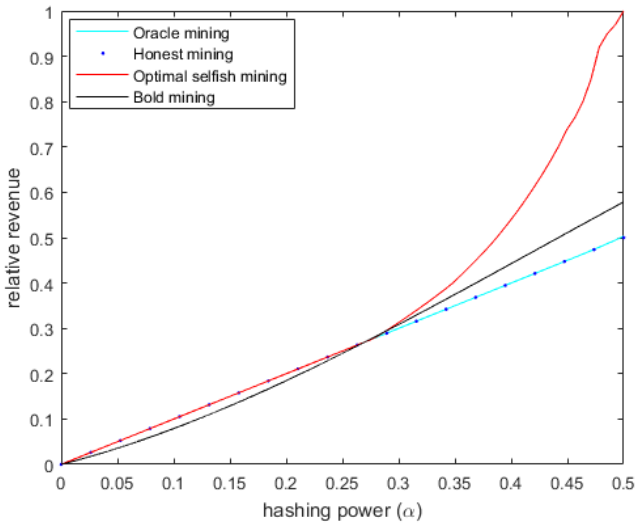

Figure (7) The relative revenues of the Optimized Selfish Mining of [6], our Bold Mining, and our Oracle Mining on our FORTIS algorithm. The attacker cannot benefit from these attacks, if his hashing power $\alpha < 27.0\%$. We note that for $\alpha < 27.0\%$, Optimized Selfish Mining and Oracle Mining generate the honest mining algorithm as the optimum strategy.

attacker's chain, i.e., for all n we have $1/n < 1 - \Upsilon$ and $1/n < \Upsilon$, if he applies OSM or BM (OM), respectively. This is good enough to ensure the incentive compatibility bounds that our scheme provides. We also note that the index of the siblings for determining the fresher chain does not make any difference against the currently known attacks, but for completeness it is decided based on the timestamps of the oldest forked siblings.

Figure 7 shows the revenues of an attacker from OM, BM, and OSM attacks applied on a PoW blockchain where honest miners practice our FORTIS algorithm.

References

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

[2] A. David, S. Maciej, V. Lorenzino, and P. Francesco, "Blockchain for digital government," tech. rep., Publications Office of the European Union, 2019.

[3] P. K. Singh, R. Singh, S. K. Nandi, and S. Nandi, "Managing smart home appliances with proof of authority and blockchain," in *I4CS '19*, 2019.

[4] T. Mashamba-Thompson and E. Crayton, "Blockchain and artificial intelligence technology for novel coronavirus disease 2019 self-testing," *Diagnostics*, vol. 10, no. 198, 2020.

[5] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," *Commun. ACM*, vol. 61, June 2018.

[6] A. Sapirshtein, Y. Sompolinsky, and A. Zohar, "Optimal selfish mining strategies in bitcoin," in *FC '16*, 2016.

[7] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains," *ACM CCS*, 2016.

to the attacker. However, this already means division of the hashing power, which has been shown to be less beneficial than collusion [20, 21, 22]

[8] K. Nayak, S. Kumar, A. Miller, and E. Shi, "Stubborn mining: Generalizing selfish mining and combining with an eclipse attack," in *IEEE EuroS&P '16*, 2016.

[9] T. Wang, S. C. Liew, and S. Zhang, "When blockchain meets ai: Optimal mining strategy achieved by machine learning," *arXiv preprint arXiv:1911.12942*, 2019.

[10] E. Heilman, "One weird trick to stop selfish miners: Fresh bitcoins, a solution for the honest miner (poster abstract)," in *FC '14*, 2014.

[11] R. Zhang and B. Preneel, "Publish or perish: A backward-compatible defense against selfish mining in bitcoin," in *CT-RSA '17*, 2017.

[12] G. Bissias and B. Levine, "Bobtail: Improved blockchain security with low-variance mining," in *NDSS*, 2020.

[13] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder, *Bitcoin and cryptocurrency technologies*, vol. 1 of *Fundamental Algorithms*. Princeton University Press, 3rd ed., 2016. (book).

[14] J. A. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and applications," in *EUROCRYPT '15*, 2015.

[15] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, "Eclipse attacks on bitcoin's peer-to-peer network," in *USENIX SEC '15*, 2015.

[16] S. Bag, S. Ruj, and K. Sakurai, "Bitcoin block withholding attack: Analysis and mitigation," *IEEE TIFS*, vol. 12, no. 8, 2017.

[17] S. A. Elliott, "Nash equilibrium of multiple, non-uniform bitcoin block withholding attackers," *ICDIS '19*, 2019.

[18] F. Ritz and A. Zugenmaier, "The impact of uncle rewards on selfish mining in ethereum," in *IEEE Euro S&P '18 Workshop*, 2018.

[19] A. Kiayias, E. Koutsoupias, M. Kyropoulou, and Y. Tselekounis, "Blockchain mining games," in *ACM EC '16*, 2016.

[20] H. Liu, N. Ruan, R. Du, and W. Jia, "On the strategy and behavior of bitcoin mining with n-attackers," in *ASIACCS '18*, 2018.

[21] F. J. Marmolejo-Cossío, E. Brigham, B. Sela, and J. Katz, "Competing (semi-)selfish miners in bitcoin," in *ACM AFT '19*, 2019.

[22] T. Leelavimolsilp, L. Tran-Thanh, S. Stein, and V. H. Nguyen, "Selfish mining in proof-of-work blockchain with multiple miners: An empirical evaluation," *arXiv preprint arXiv:1905.06853*, 2019.

[23] E. Koutsoupias, P. Lazos, F. Ogunlana, and P. Serafino, "Blockchain mining games with pay forward," in *WWW '19*, 2019.

[24] M. Davidson and T. Diamond, "On the profitability of selfish mining against multiple difficulty adjustment algorithms." Cryptology ePrint Archive, Report 2020/094, 2020.

[25] M. Carlsten, H. Kalodner, S. M. Weinberg, and A. Narayanan, "On the instability of bitcoin without the block reward," in *ACM CCS '16*, 2016.

[26] I. Tsabary and I. Eyal, "The gap game," in *ACM CCS '18*, 2018.

[27] R. Pass, L. Seeman, and A. Shelat, "Analysis of the blockchain protocol in asynchronous networks," in *EUROCRYPT '17*, 2017.

[28] R. Zhang and B. Preneel, "Lay down the common metrics: Evaluating proof-of-work consensus protocols' security," in *IEEE SP*, 2019.

[29] Y. Sompolinsky and A. Zohar, "Secure high-rate transaction processing in bitcoin," in *FC '15*, 2015.

[30] L. Bahack, "Theoretical bitcoin attacks with less than half of the computational power (draft)," *arXiv preprint arXiv:1312.7013*, 2013.

[31] B. Shultz, "Certification of witness: Mitigating blockchain fork attacks," 2015.

[32] S. Solat and M. Potop-Butucaru, "Zeroblock: Timestamp-free prevention of block-withholding attack in bitcoin," in *SSS '17*, 2017.

[33] M. Saad, L. Njilla, C. A. Kamhoua, and A. Mohaisen, "Countering selfish mining in blockchains," *ICNC '19*, 2019.

[34] V. Chicarino, C. Albuquerque, E. Jesus, and A. Rocha, "On the detection of selfish mining and stalker attacks in blockchain networks," *Ann. Telecommun.*, vol. 75, p. 143–152, 2020.

[35] S. E. Alm, *Simple random walk*. 2002. `http://www2.math.uu.se/~sea/kurser/stokprocmn1/slumpvandring_eng.pdf`.

[36] `https://github.com/nirenzang/Optimal-Selfish-Mining-Strategies-in-Bitcoin`, 2017. Accessed: 2020-04-30.