FORTIS: Selfish Mining Mitigation by (FOR)geable (TI)me(S)tamps

OSMAN BİÇER and ALPTEKİN KÜPÇÜ, Koç University, Turkey

The selfish mining (SM) attack of Eyal and Sirer allows a rational mining pool with a hash power (α) much less than 50% of the whole Bitcoin network to steal from the fair shares of honest miners. This attack has been studied extensively in various settings in order for its optimization and mitigation. In this context, Heilman proposes a defense "Freshness Preferred", based on timestamps, which are issued routinely by a timestamp authority. In contrast, we consider the case where timestamps are generated by no authority; instead every miner includes the current time into a block freely. However, due to two attacks that we discover, this turns out to be a non-trivial task. These attacks are *Oracle mining*, which works by cleverly setting the timestamp to future, and *Bold mining*, which works by generating an alternative chain starting from a previous block. Unfortunately, these attacks are hard to analyze and optimize, and to our knowledge, the available tools fail to help us for this task. To ease this, we come up with generalized formulas for revenue and profitability of SM attacks. Our analyses show that the use of timestamps could be promising for selfish mining mitigation. Nevertheless, Freshness Preferred in its current form is quite vulnerable, as any rational miner with $\alpha > 0$ can directly benefit from our attacks. To cope with this problem, we propose a novel SM mitigation algorithm `Fortis` without an authority, which protects the honest miners' shares against any attacker with $\alpha < 27.0\%$ against all the known SM-type attacks. By building upon the blockchain simulator BlockSim, we simulate our Oracle and Bold mining attacks against Freshness Preferred and `Fortis`. Simulation results also demonstrate the effectiveness of these attacks against the former and their ineffectiveness against the latter.

CCS Concepts: • **Security and privacy → Security protocols**; **Economics of security and privacy**; • **Theory of computation → Distributed algorithms;**

Additional Key Words and Phrases: Selfish mining, cloud mining, bitcoin, proof-of-work, blockchain, timestamp

ACM Reference format:
Osman Biçer and Alptekin Küpçü. 2023. FORTIS: Selfish Mining Mitigation by (FOR)geable (TI)me(S)tamps. *Distrib. Ledger Technol.* 2, 4, Article 28 (December 2023), 26 pages.
https://doi.org/10.1145/3616397

1 INTRODUCTION

First proposed by [48] for Bitcoin, **proof-of-work** (**PoW**) blockchain plays a crucial role as the underlying technology behind modern cryptocurrencies (e.g., Bitcoin, Ethereum, and Litecoin) and many distributed and cloud-based applications (e.g., certificate transparency [28], smart contract [14], e-government [17], e-voting application [35], online donations systems [11], smart appliances [59], and healthcare applications [57]). PoW blockchain depends on an incentive-based mechanism, named *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 becomes appended to the blockchain. *Miners* are investors of computational resources that are specialized for mining procedures. This investment may be in the form of buying or hiring some computational and

We acknowledge support from TÜBİTAK, the Scientific and Technological Research Council of Turkey, under project number 119E088.
Authors' address: O. Biçer and A. Küpçü, Koç University, Sarıyer, İstanbul, Turkey; e-mails: obicer17@ku.edu.tr, akupcu@ku.edu.tr.

2769-6472/2023/12-ART28
https://doi.org/10.1145/3616397

network resources, or the miners can outsource the mining work to cloud services. When a miner mines a block, she receives a wealthy block reward, which builds up the incentive for her investment.

Selfish mining attack. Nakamoto claimed that as long as the majority (>50%) of hashing power in the system belongs to the miners that follow the correct mining algorithm, the reward of each miner would be proportional to her expenditures, securing proper operation of the blockchain. Yet, [22] proposed the ***selfish mining (SM) attack*** that yields a miner more than her fair share by deviating from honest mining (i.e., following the publicly accepted honest algorithm), 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 due to the fact that the honest miners always choose the chain with more blocks (the longer chain) and that the difficulty adjusts over time. In fact, none of these causes seems avoidable, as the former is a countermeasure against network partitions and propagation delays, and the latter is an initial design choice to ensure constant the expected inter-block time of the main chain. The selfish mining attack is extensively studied in the recent blockchain literature, including the research for optimizing it [27, 50, 56, 63], combining it with other attacks (e.g., with eclipse attack [32, 50] and block withholding attack [1, 8, 20, 21]), and defending against it [10, 22, 31, 68]. Although so far no known selfish mining attack occurred in practice on Bitcoin, [42] detected selfish mining anomalies in Monacoin, Bitcoin Cash, and Ethereum. The practice of the attack would be harmful not only by reducing the fair shares of miners, but also by reducing trust of honest users on the system and negatively affecting its perception and wide-spread use. Selfish mining remains as a major focus in blockchain research [33].

Timestamp based SM defense. The initial defense mechanism proposed by [22] was that honest miners should choose one of the chains at random, when there are multiple longest chains of the same *length* (i.e., the number of blocks), while in the recent 22.0 version of Bitcoin client implementation Bitcoin Core[1] as of April 2022, miners choose the one that they received first. Regardless of the attacker's location or bandwidth, this defense prevented him from obtaining high rewards for computational powers below 25% hashing power of the whole system.

A later proposal [31] utilizes timestamps issued by an authority, claiming to improve this resistance up to the computational power of 33.3%. Although by the state-of-the-art selfish mining attack [56], its security lowers to 30.1%, [31] still remains as the most resistant work against selfish mining. The defense idea is that honest miners would pick the chain with fresher timestamps in case of a tie in chain lengths. As this solution was contradicting to the decentralized philosophy of blockchain by requiring a timestamp authority, Heilman also proposed another solution without the authority. However, we show that this solution without a timestamp authority (i.e., with forgeable timestamps) is susceptible to two attacks: one by cleverly setting the timestamp to the future and another by mining on the block previous to the last mined one.[2]

Our approach. Ideally, one would like to achieve better security against optimized SM attack ([56]) without needing a central authority. Yet, when the miners put the current time into their mined blocks themselves, two issues arise: it becomes possible for a defective miner to set the timestamp of the block being mined to a future point (*Oracle mining*), and it gets easier to mine an alternate chain by starting from an old block (*Bold mining*). We show that if the attacker fine-tunes the attack parameters, unfortunately, choosing the fresher chain works in favor of the maliciously mined chain in both cases. These attacks effectively zero-out the security of the Heilman proposals in [31].

Although these attacks are simple and natural, their analyses and optimization for the attacker led us to develop a new mathematical model, which can be standardized for using in selfish mining attack analyses beyond our work. To solve these issues, we then propose `Fortis`, a mining algorithm based on forgeable timestamps

[1] https://bitcoincore.org

[2] Without a timestamp authority, [31] also consider a possible attack by setting the timestamp to the future, and named it as "slothful mining". However, this attack has neither been formally defined nor analyzed so far.

that is optimized to be *not* vulnerable to these attacks or the state-of-the-art optimal selfish mining attack of [56] against a computational power ratio up to 27.0%. Thereby, we achieve the highest security against a single rational miner capable of performing all the selfish mining attacks known to date. More concretely, we list the contributions of this work as follows:

(1) We define two mining attacks, namely Bold mining and Oracle mining, exploiting the timestamp based solutions of [31].
(2) We propose generic formulas that can be utilized for revenue calculations in selfish mining type of attacks. Our formulas makes complicated analysis of attacks easier and more dependable by providing a systematical methodology. We use this to show that there exists optimized choices of parameters in both Bold and Oracle mining attacks that yield a miner more than his fair share, even if his resources are very low.
(3) 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 to date. Our solution is decentralized and only assumes that miners have synchronized clocks (which is a common assumption in other blockchain proposals [6, 37]).
(4) We provide the simulation results for Oracle and Bold mining attacks against the solution of [31] and against our `Fortis` algorithm. Our simulation is based on implementation of these attacks and defenses on top of the core blockchain simulation Blocksim [3]. It confirms our theoretical results by showing that these attacks are indeed effective against [31] and ineffective against our `Fortis`. We also simulate small (but realistic) clock drifts and show that the attacks and our defense are still viable.

2 RELATED WORK

Blockchain. A blockchain is essentially an ever-growing chain of data blocks [48, 49]. Each block consists of a set of recent transactions, its index from the first block, the hash of the previous block (its parent), and a nonce. The hash of each block is required to be lower than a publicly determined value called *difficulty*. The blockchain miners keep competing to become the first one to mine the next block.[3] 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. An honest miner always mines on 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.[4] 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. [25] consider the aim for fairness in a PoW blockchain as the revenue of each miner being proportional to her investment. 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. A further review of blockchain security issues can be found in [43].

Selfish Mining (SM) attacks. A major strike for blockchain security was the invention of the SM attack by [22]. Briefly, 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 his private chain in terms of length. This way, the attacker can eliminate the honest blocks from the main chain. They 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.

We call any type of attack that involves a miner who keeps his mined blocks private so as to obtain high relative revenue as an SM-type attack. Some recent studies of selfish mining optimize it further by allowing the

[3]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.
[4]The Bitcoin application in April 2022 suggests that each miner mines on the chain that she received the first.

attacker to mine on the private chain even when the public chain is longer ([27, 50, 56, 63]). Other focuses of attention related to the SM attack are combining it with different attacks (e.g., with Eclipse attack [32, 50] and block witholding attack [1, 8, 20, 21]), conducting the attack on different currencies (e.g., on Ethereum [54]), and exploring the game theoretical implications of the attack by [36, 38, 41, 44, 47]. We note that the state-of-the-art optimized selfish mining attack is "**optimal selfish mining**" (**OSM**) algorithm [56]. We do not go over the details of the algorithm or its relative revenue calculations here, and refer the reader to the original article.

Selfish mining is harmful to the blockchain system, since it not only is stealing from the fair shares of the honest miners, but also results in inconsistent views of blockchain among the users. A recent study [18] 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 [68]. The attack is further harmful, if miners are open to bribery [23, 65]. Low resource expenditure systems are even further vulnerable to the attack [1].

Existing selfish mining defenses. We briefly review the existing selfish mining defenses (honest mining algorithms) in the literature.

Uniform tie-breaking. The initial SM defense proposed by [22] was that an honest miner picks a branch uniformly at random among the longest chains in case of ties. We call this defense as **Uniform Tie-breaking** (**UT**). Against the SM attack, this defense achieves security up to hashing power 25.0%. However, the later optimized attack proposal of [56] reduced this to 23.2%.

Freshness preferred. Heilman proposes use of timestamps for tie-braking to reduce the relative revenue of a selfish miner [31]. In case of a tie, the miners choose the branch that is fresher (i.e., the branch whose **timestamps** (**TS**) are closer to the current time τ); hence the scheme is called "**Freshness Preferred**" (**FP**). 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. If the timestamps are generated in an infinitesimal fashion, the scheme claims to improve security against SM to the hashing power 33.3% and against OSM to the hashing power 30.1% (computed by using the implementation of [67]). However, this scheme suffers from centralization and novel attacks that we propose.

Publish or Perish (PP). [68] provides incentive compatibility against OSM up to a hashing power 25.0%. As we do not utilize this algorithm and its related defense ideas in our work, we skip the protocol details and refer the reader to [68]. For comparison, `Fortis` provides security to all known attacks (including ones analyzed in this work) up to a hashing power 27.0%. Also, [68] defines a notion "fail-safe parameter" as the minimum length difference required for enforced adoption of the longest branch, and suggests 3 as the optimum fail-safe parameter for PP. The article shows that in this case, there exists an attack that results in an expected 18.5 blocks to pass for a consensus in case of a tie. We stress that this attack is not possible in our case and even in case of a tie, the consensus among the honest miners occurs immediately. Although, in a setting with complete absence of synchronized clocks, PP remains as the most secure solution known to date, our experiments in Section 8 shows that even with a loose synchronization, `Fortis` defends against attackers with hashing power of 27.0%.

To clarify the significance of our 2% improvement against attacks, in Bitcoin, it corresponds to roughly 19 Billion US Dollars additional investment in hardware for an attacker to benefit from the attack, as of April 2022.[5] We note that this does not include the additional cost of electricity expenditures.

Other works. GHOST [61], Bobtail [10], and the other works [9, 53, 58, 60, 69], to the best of our knowledge, fail to satisfy backward compatibility [68]. That is, old blocks cannot be verified with them. Also, there exist some works for detecting the behavior of the selfish miner and eliminate his blocks from the main chain [15, 34, 55].

[5]The cost is calculated from the fact that in April 2022, the total hash rate is roughly 200 million tera hash per second (TH/s) according to https://ycharts.com/indicators/bitcoin_network_hash_rate and the mining hardware Bitmain Antminer S19 Pro 110TH with the hash rate 110 TH/s can be found for 10,399 US Dollars in Amazon.

Table 1. Comparison of Uniform Tie-breaking (UT) [22], Freshness Preferred with unforgeable and forgeable timestamps (FTwUTS and FTwFTS, respectively) [31], Publish or Perish (PP) [68], and our Fortis Schemes

		UT	FPwUTS	FPwFTS	PP	Fortis
Decentralization		Full	Timestamp authority	Full	Full	Full
Incentive Compatibility	against [56]	23.2%	30.1%	30.1%	25.0%	27.0%
	against Oracle mining	N/A	0	0	N/A	27.0%
	against Bold mining	N/A	N/A	0	N/A	27.0%
	overall	**23.2%**	**0**	**0**	**25.0%**	**27.0%**

Regarding the attack [56], the values are computed by using the implementation [67]. N/A denotes that the attack is not applicable to the given defense. Overall incentive compatibility of each defense is obtained by the minimum incentive compatibility of that defense against applicable attacks.

However, this detection has not been reliably achieved yet, as there seems to be no clear way for differentiating it from network partitioning. [40] elaborates on the structure of mining pools and shared information among miners within a pool, and propose a defence mechanism accordingly.

To the best of our knowledge, the selfish mining defense algorithm that scores the best with respect to the above-mentioned evaluation criteria is FPwUTS, as it provides full decentralization, and incentive compatibility up to $\alpha = 0.301$. However, we provide novel SM-type attacks, **oracle mining** (**OM**) and **bold mining** (**BM**), specialized on FPwFTS and FPwUTS schemes, respectively. Our attacks effectively reduce the incentive compatibility of both FPwFTS and FPwUTS to $\alpha = 0$ (i.e., an attacker with any hashing power benefits from our attacks). Table 1 provides comparison of the existing schemes and our Fortis scheme that we describe in Section 7. Only the systems that provide backward compatibility [68] are compared, for possible transition of existing systems such as Bitcoin. [68] defines the notion of backward compatibility briefly as acceptance of blocks (generated by the updated mining algorithm) by the previously deployed mining algorithm.

3 PRELIMINARIES

In this section, we introduce the notation that we employ for general SM attacks, and detail the Uniform Tie-breaking and Freshness Preferred approaches.

Notation for mining algorithms. In a miner's view, there exists two separate chains, the chain PubCh known by all miners and the chain MyCh 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) signals "append the block b to the head of the Chain".
- Publish(b) signals "publish the block b".
- Publish(MyCh,z) signals either
  - (in case z is an integer) "publish all the blocks including z-th block of MyCh indexed from start of the fork from PubCh" or
  - (in case z is "head") "publish the head of PubCh".
- Trun(Chain,z) returns the chain obtained by truncating Chain by z blocks backward from the head.
- Last denotes the index of the last block found by the miner in PubCh backward from the head.
- A $\twoheadrightarrow b$ denotes that "A finds the next block b". A denotes "The attacker's pool" and O denotes "Any pool other than attacker's one".
- Mine(Chain,Fork,z,TS) signals "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". For simplicity, if the fork resolving policy does not depend on timestamps, the input TS may be omitted. Also, the attacker always mines on the head of $MyCh$ therefore, the inputs Chain, Fork, and z are omitted.

Table 2. Table of Frequent Abbreviations and Notations used throughout this Article

Abbreviation/Notation	Explanation
TS	Timestamp
UT	Uniform tie-braking
FP	Freshness preferred
Authority FP	Freshness preferred scheme with unforgeable timestamps
Decentralized FP	Freshness preferred scheme with forgeable timestamps
PP	Publish or perish scheme of [68]
SM	Selfish mining attack of [22]
SM-type attacks	Attacks performed to receive high revenue via private mining
OSM	Optimized selfish mining attack of [56]
OM	Oracle mining (our novel attack on FP)
BM	Bold mining (our novel attack on FP)
α	Hashing power
γ	Network power
τ	The current timestamp (generated by either the local synchronized clock or timestamp authority)
ϵ	Increment of the timestamp
t_m	The difference of the timestamp from current time for maximizing the revenue of Oracle miner
t_C	The expected time interval between block found by any miner (meaningful in Oracle mining)
S_{pub}	The state of the SM-type attacker when mining on a publicly known block
S_{priv}	The state of the SM-type attacker when mining on a block private to himself
R	Revenue of a miner

Table of abbreviations. We provide Table 2 for the frequently used abbreviations and notations throughout the article.

Uniform tie-breaking algorithm. The honest miner's UT strategy of [22] is given in Algorithm 1.

ALGORITHM 1: Uniform Tie-breaking Mining Algorithm of [22]

```
while true do
    if PubCh has 1 branch then
        Mine(PubCh,PubCh,1)
    else if PubCh has n branches then
        Mine(PubCh,i-th branch,1) with probability 1/n
    end if
    Propagate PubCh
end while
```

Freshness preferred algorithm. The honest miner's strategy in [31] is given in Algorithm 2. In case of a tie, the miners choose the branch that is fresher (i.e., the branch whose TS are closer to the current time τ); hence the scheme is called "**Freshness Preferred**" (**FP**). 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 synchronized clocks, 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 (Authority FP) and Freshness Preferred with forgeable timestamps (Decentralized FP), respectively.

Model of revenue analysis.[6] The model that we utilize throughout the article 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

[6]This model is also the one provided by the original SM article [22]. Although analyses in different settings (e.g., without fixed block rewards [13, 62], in asynchronous setting [27, 52], in the presence of multiple attackers [41, 44, 47]) also exist, the previous attacks and defenses are mostly analyzed in this model.

ALGORITHM 2: Freshness Preferred Mining Algorithm of [31]

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

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). Throughout the article, we do not consider $\alpha \geq 0.5$ as in such a case, the attacker may simply mount a double-spending attack instead. We note that in Bitcoin, γ depends on the bandwidth and location of the attacker, as miners choose the branch that they receive first in case of a tie. We omit the effect of 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 and blacklisting a malicious miner is impossible. We also assume that computation times other than hashes and costs due to mining are negligible. The finder of any block in the main chain is rewarded 1 unit, 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.

It would be interesting to see the analyses of our attack and defense proposals in presence of multiple attackers. It has been shown that selfish miners perform better, when there are other selfish miners in the system [41, 44, 47]. Yet, these analyses become very complicated, even when there exist two attackers. Thus, in this work, we limit ourselves to one attacker. Also, the analysis of our defense in a more realistic "imperfect network" setting as in [64, 66] is left as an interesting future work. We note that other mentioned defenses (except for Uniform Tie-breaking) are analyzed only in the presence of one attacker. The only additional assumption that we make over these works is presence of synchronized clocks, but later in Section 8, we provide simulation results to show that even with a loose synchronicity we provide high security.

4 ORACLE AND BOLD MINING ATTACKS

In this section we present two novel SM-type attacks, the Oracle and Bold Mining attacks, on Decentralized FP. These attacks are direct exploits of enforcing the honest miners pick the freshest *looking* chain in case of a tie. We highlight that these attacks are not covered in the optimal selfish mining [56], as they do not consider the use of self-generated timestamps for selfish mining mitigation.

We represent all SM-type attack algorithms in this article with two distinct 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. This separation is sound based on the generic idea of selfish mining attack strategies 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 in S_{pub}, but sometimes when the attacking conditions occur, he deviates from the honest mining strategy to go to S_{priv}. The reason for separation of these two states is simplicity of revenue calculation, and will be clearer in Section 5).

Oracle mining algorithm. At S_{pub} the attacker sets the timestamp $TS = \tau + t_m$ of the block he is mining on, where τ is the current time (based on the local synchronized clock) and $t_m > 0$. We will later show how to

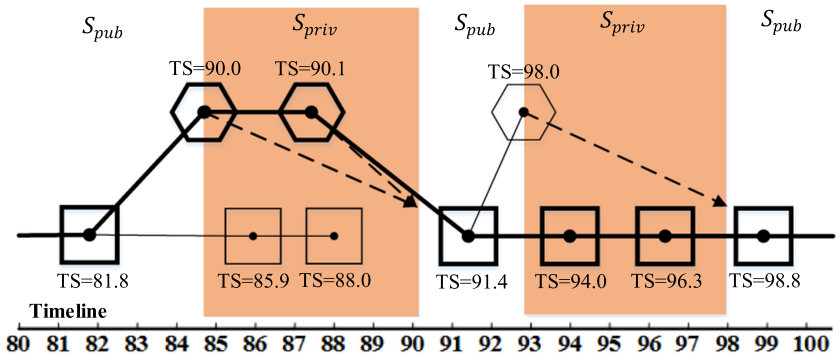


Fig. 1. An example flow of Oracle mining attack. The squares are honest miners' blocks, while the hexagons are those of the attacker. Straight lines show parent-child relation between two blocks. The thick-lined blocks are the ones that remain in the main chain and get rewarded, while the thin-lined 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 shaded as white and orange, respectively. Timestamps are incremented with $\epsilon = 0.1$

set t_m 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 $\tau + t_m$ (i.e., by switching 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 blocks will be orphaned. Otherwise, he will orphan all the blocks of honest miners found during S_{priv} off the main chain. Algorithm 3 provides full description of the OM attack. Also, an example flow of OM is shown in Figure 1. The attacker finds a block at the time 84.7 (but its timestamp is set as 90.0), and kept for private mining. At 90.1, the attacker releases his private chain and kicks out two blocks from the public chain as his chain has a fresher timestamp. He also finds another block at the time 92.8 (the timestamp is set as 98.8), which is released at the time 98.8, but this time the attacker's block is kicked out as the public chain is longer.

Bold mining algorithm. In this attack, the attacker's strategy is essentially to mine a sibling to the kth past block from the current head at S_{pub} (if he does not own any block between the current head and the kth past block, both inclusive), and then to keep it private (going to S_{priv}) and 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. Algorithm 4 provides the BM algorithm. Also, an example flow of the BM attack is shown in Figure 2. The attacker starts from two blocks behind the head of the public chain and finds a block at the time 86.7. He mines on this block until the time 91.4 when he catches the public chain. He releases his private chain to kick out three blocks of the public chain as the former has fresher timestamps. At the time 97.8 by mining on just one block behind the head. Then he immediately releases the block to kick out the public one.

As a side note, the bold mining attack is applicable to Authority FP as well as Decentralized FP. This is because unlike oracle mining, the attacker mines the alternative chain to the public one using the current actual timestamp. Thus, all results of this attack are achievable even in the presence of a timestamp authority. Although our concern is the decentralized setting, this attack and its analysis and optimization given in Section 6.2 apply to the settings with timestamp authorities.

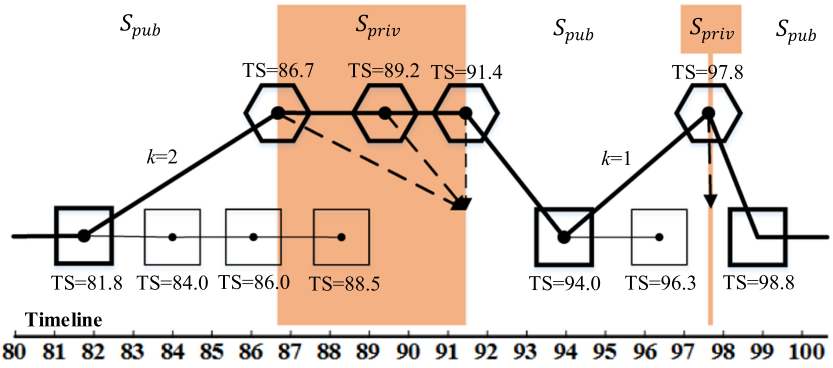


Fig. 2. An example flow of Bold mining attack. The squares are honest miners' blocks, while the hexagons are those of the attacker. Straight lines shows parent-child relation between two blocks. The thick lined blocks are the ones that remains in the main chain and gets rewarded, while the thin lined 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 orange, respectively. Note that in case $k = 1$, the attacker immediately releases the block in S_{priv}.

ALGORITHM 3: Oracle Mining Attack Algorithm

procedure S_{pub}
  Set $\Delta \leftarrow 0$, $flag \leftarrow 0$
  while $flag = 0$ **do**
    Find t_m for α
    MyCh $\leftarrow$ PubCh, Mine$(\tau + t_m)$
    if A $\twoheadrightarrow b$ **then**
      Append(MyCh,b)
      Set $t \leftarrow \tau + t_m$, $\Delta \leftarrow 1$, $flag \leftarrow 1$
    end if
  end while
  Go to procedure S_{priv}
end procedure

procedure S_{priv}
  while $\tau < t + (\Delta - 1) \cdot \epsilon$ **do**
    Mine$(t + \Delta \cdot \epsilon)$
    if A $\twoheadrightarrow b$ **and** $\Delta > 0$ **then**
      Append(MyCh,b), Set $\Delta \leftarrow \Delta + 1$
    end if
  end while
  Publish(MyCh,head), Go to procedure S_{pub}
end procedure

Optimum parameters for the attacks. The values t_m and K in Oracle and Bold Mining are the parameters that the attacker should choose, respectively. As a rational party, he would naturally be interested in setting them in a way that it would maximize his share in the system. However, this requires a complete analysis of both attacks' revenues. In particular, the analysis of oracle mining is difficult, and although the attack idea exists

ALGORITHM 4: Bold Mining Attack Algorithm

procedure S_{pub}
  Set $flag \leftarrow 0$
  while $flag = 0$ **do**
    Set $\Delta \leftarrow \min(\texttt{Last}, K) - 1$
    $\texttt{MyCh} \leftarrow \texttt{Trun}(\texttt{PubCh}, \Delta), \texttt{Mine}(\tau)$
    if A $\twoheadrightarrow b$ **and** $\Delta = 0$ **then**
      Publish(MyCh,head)
    else if A $\twoheadrightarrow b$ **and** $\Delta > 0$ **then**
      Append(MyCh,b)
      Set $k \leftarrow \Delta$, $\Delta \leftarrow \Delta - 1$, $flag \leftarrow 1$
    else if O $\twoheadrightarrow b$ **and** $\Delta < K$ **then**
      Set $\Delta \leftarrow \Delta + 1$
    end if
  end while
  Go to procedure S_{priv}
end procedure

procedure S_{priv}
  while $0 < \Delta < k + 1$ **do**
    $\texttt{Mine}(\tau)$
    if A $\twoheadrightarrow b$ **and** $\Delta > 1$ **then**
      Append(MyCh,b), Set $\Delta \leftarrow \Delta - 1$
    else if A $\twoheadrightarrow b$ **then**
      Set $\Delta \leftarrow \Delta + 1$
    end if
  end while
  Publish(MyCh,head), Go to procedure S_{pub}
end procedure

in a basic form in the work of [31], it has never been analyzed ever since. Thus we will return setting the optimum of these parameters, after their analyses in Section 6.

5 GENERIC FORMULAS FOR SM ATTACKS

The formulation. We now derive a formulation that we will use for analyzing the attacks given in Section 4. We start by observing that the known SM-type attacks (including our BM and OM) show a pattern for the attacker alternating between mining on the publicly known chain and his own private one. Let f denote expected frequency of returning to one of the states S_{priv} or S_{pub}. We formally define it as

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

where ES_{priv} (or ES_{pub}) is the expected number of total blocks mined by both the attacker and the honest parties during state S_{priv} (or S_{pub}, respectively). If the attacker only executes honest mining all the time during an expected alternation period $ES_{priv} + ES_{pub}$, the expected number of blocks that he has found and is rewarded for would be $\alpha \times (ES_{priv} + ES_{pub})$ (where α is the attacker's hashing power), and those that all miners have found and are rewarded for would be $ES_{priv} + ES_{pub}$.

When an attacker applies an SM-type attack, we observe that the block losses (i.e., the found but not rewarded blocks) occur only during S_{priv}, as this is when the attack effectively takes place. Here, the attacker aims at inflicting as much loss as possible to other miners. In return, the attacker may also risk his own found blocks during S_{priv}. To calculate the expected number of blocks that the attacker is rewarded for, the attacker's expected block loss (denoted by ℓ_A) in a given period $ES_{priv} + ES_{pub}$ is extracted from his expected number of found blocks, i.e., $\alpha \times (ES_{priv} + ES_{pub}) - \ell_A$. Also, the expected number of total rewarded blocks is $ES_{priv} + ES_{pub} - (\ell_A + \ell_H)$, where ℓ_H is the expected block loss of the honest parties in S_{priv}. Thanks to the difficulty adjustment, the expected number of total rewarded blocks will be the same in a given time period whether there exist an attack or not. We derive the expected revenue R_{period} of the attacker in a period by dividing the expected number of blocks that the attacker is rewarded for by the expected number of total rewarded blocks as

$$R = \frac{\alpha \times (ES_{priv} + ES_{pub}) - \ell_A}{ES_{priv} + ES_{pub} - (\ell_A + \ell_H)}. \tag{2}$$

For generalizing out of a period, in the equation above, we divide both numerator and denominator by the expected number $ES_{priv} + ES_{pub}$ of total found blocks in a period, and obtain the expected revenue R of the attacker as

$$R = \frac{\alpha - f \cdot \ell_A}{1 - f \cdot (\ell_A + \ell_H)}. \tag{3}$$

Based on the losses of the attacker and the other miners, the attack may increase or decrease the attacker's revenue. Whether the attacking strategy is expected to benefit the attacker depends on the ratio of $\frac{\ell_A}{\ell_H}$. More concretely, if and only if the attacking strategy benefits the attacker, we have

$$\frac{\ell_A}{\ell_A + \ell_H} < \alpha \quad \text{or equivalently} \quad \frac{\ell_A}{\ell_H} < \frac{\alpha}{1 - \alpha}. \tag{4}$$

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 (i.e., f is also needed), but is rather a tool for his decision-making in choosing between applying an SM-type mining strategy and honest mining. We note that solving for $R > \alpha$ in Equation (3) leads to Equation (4).

Example application. To show that our formulas indeed work, we apply them to the basic selfish mining algorithm of [22]. Algorithm 5 provides the same SM algorithm given by [22], but it shows S_{priv} and S_{pub} states as distinct procedures.

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 Helen finds a block first in S_{priv}, and then one of the $\gamma(1-\alpha)$ fraction of honest miners that work on Helen's block finds the next block. In this case, the attacker Adam loses 1 block. As the probability that this case occurs is $(1-\alpha)^2(1-\gamma)$, we obtain

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

where the subscript SM denotes selfish mining.

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 $(1-\alpha)(\gamma(1-\alpha)+\alpha)$. In this case, Helen loses 1 block. (2) Adam finds the next block with probability α, eventually Helen catches up with Adam by mining 1 fewer blocks than him, Adam publishes his chain and Helen loses all the blocks she has found within S_{priv}. For the second outcome, we can formulate this as a "monkey at the cliff" problem (see Section 2 of [4]), starting when Adam's chain is 2 blocks ahead of Helen's chain and ending when his chain is 1 block ahead of hers. The monkey goes towards the cliff with probability $p = 1 - \alpha$ and away from it with probability $q = \alpha$. For the monkey at the cliff problem, the expected number of steps to the end shown by Theorem 2 of [4] is $\frac{k}{p-q}$ (for the sake of non-redundancy we refer to [4] for its proof). Here, $k = 2 - 1$ as the walk starts with 2 blocks

ALGORITHM 5: Selfish Mining Attack Algorithm of [22]

procedure S_{pub}
  Set $\Delta \leftarrow 0$, $flag \leftarrow 0$
  while $flag = 0$ **do**
    if |MyCh| = |PubCh| and MyCh $\neq$ PubCh **then**
      Mine()
      if A $\twoheadrightarrow b$ **then**
        Append(MyCh,b), Publish(MyCh,head)
      end if
    else
      MyCh $\leftarrow$ PubCh, Mine()
      if A $\twoheadrightarrow b$ **then**
        Append(MyCh,b), Set $\Delta \leftarrow 1$, $flag \leftarrow 1$
      end if
    end if
  end while
  Go to procedure S_{priv}
end procedure

procedure S_{priv}
  Set $flag \leftarrow 0$
  while $flag = 0$ **do**
    Mine()
    if A $\twoheadrightarrow b$ **and** $\Delta > 0$ **then**
      Append(MyCh,b), Set $\Delta \leftarrow \Delta + 1$
    else if O $\twoheadrightarrow b$ **and** $\Delta \leq 2$ **then**
      Publish(MyCh,head), $flag \leftarrow 1$
    else if O $\twoheadrightarrow b$ **and** $\Delta > 2$ **then**
      Publish(MyCh,1), Set $\Delta \leftarrow \Delta - 1$
    end if
  end while
  Go to procedure S_{pub}
end procedure

difference and ends 1 block difference. Hence, we obtain the number blocks to be found as $\frac{k}{p-q} = \frac{2-1}{(1-\alpha)-\alpha} = \frac{1}{1-2\alpha}$. Since the expected number of steps *towards* the cliff should be *one more than* those *away from* it for the walk to end, we calculate Helen's expected block loss in this case as $\frac{1}{2}(\frac{1}{1-2\alpha}+1) = \frac{1-\alpha}{1-2\alpha}$. At the end, we obtain

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

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 $ES_{priv} = (1-\alpha)+\alpha(1+\frac{1}{1-2\alpha}) = 1+\frac{\alpha}{1-2\alpha}$. 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

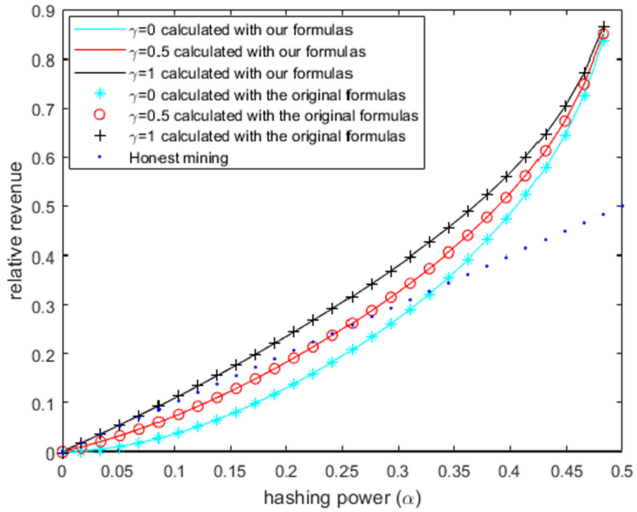


Fig. 3. Revenue from selfish mining [22] with respect to hashing power α for network powers $\gamma = 0$, $\gamma = 0.5$, and $\gamma = 1$ calculated using our Equation (3) and the original equation of [22] given at Equation (5).

of $\frac{1}{\alpha}$ blocks, since it is a geometric random variable. Hence, we calculate $ES_{pub} = 1 - \alpha + \frac{1}{\alpha}$. From Equation (1), we obtain that

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

If we plug $\ell_{A,SM}$, $\ell_{H,SM}$, and f_{SM} into Equation (3), we obtain an equation equivalent to the revenue equation of [22] as

$$R_{SM} = \frac{\alpha(1-\alpha)^2(4\alpha + \gamma(1-2\alpha)) - \alpha^3}{1 - \alpha(1 + (2-\alpha)\alpha)}, \tag{5}$$

which is also deducible numerically from Figure 3, that shows the revenues of the attacker with respect to his hashing power for network powers $\gamma = 0$, $\gamma = 0.5$, and $\gamma = 1$ for the original and our generalized equations.

6 ANALYSES OF ORACLE AND BOLD MINING

We now show that there exists parameters for t_m in Oracle Mining and K in Bold Mining, such that the attacker is better off compared to honest mining regardless of his hashing power. The simulation results in Section 8 also confirm our theoretical results obtained in this section.

6.1 Oracle Mining Attack Optimization

Analysis. We highlight that the smaller the unit increment ϵ value is, the more option the attacker will have to set the future time stamp. Hence, the attacker is strongest if $\epsilon \approx 0$. Here we set $\epsilon = 0$ and conduct the analysis in continuous time. We model the mining in a given time range as a Poisson random variable as in [56]. 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, and μ is a success rate (see below). Let T_c denote the current expected time interval in between blocks found by any miner. This differs from the public expected inter-block time by including the blocks that do not end up in the main chain. The attacker can determine this value from his hashing power, or can estimate it from statistics as he knows his and honest miners' found blocks. We let $\mu_A = \frac{\alpha t_m}{T_c}$ and $\mu_H = \frac{(1-\alpha)t_m}{T_c}$ denote success rates of Adam and Helen, respectively.

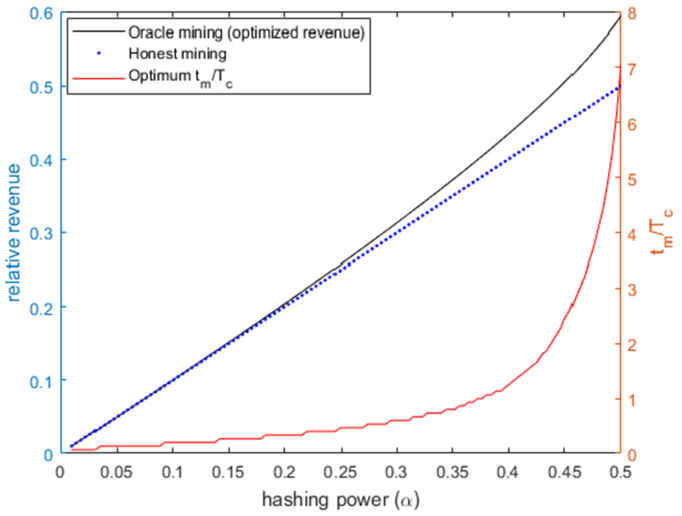


Fig. 4. Revenue from Oracle mining on Decentralized FP w.r.t. the hashing power α when the timestamps set optimally. Also the optimum t_m/T_c values for all α of interest are provided.

We can state

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

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

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_m, we calculate $ES_{priv} = \frac{t_m}{T_c}$. Also, $ES_{pub} = \frac{1}{\alpha}$, as S_{pub} finishes when Adam finds a block. Hence, from Equation (1), we obtain

$$f_{OM} = \frac{1}{\frac{t_m}{T_c} + \frac{1}{\alpha}} = \frac{\alpha T_c}{\alpha t_m + T_c}.$$

Plugging $\ell_{A,OM}$, $\ell_{H,OM}$, and f_{OM} into Equation (3), it is straightforward to obtain Adam's revenue. By setting t_m 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_m 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_m 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 himself. We note that both α and T_c may vary over time, but the attacker can estimate them from the difficulty and the statistics of the publicly and privately mined blocks. Figure 4 also shows that the Oracle Mining strategy (using optimized values) is always better than the honest mining strategy.

Impact of the attack. The most important implication of this attack is that for any α value, there exists a value $t_m > 0$ 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

a Decentralized FP application, since it may be attractive for *any* rational miner, while the other attacks usually require 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.

6.2 Bold Mining Attack Optimization

Analysis of profitability ratio for all bounds K. Remember that a Bold Mining attacker mines starting from the k^{th} past block to the last. We know from the previous analyses (including the one by [48]) of PoW mining that it is not a good idea to mine on top of long past blocks; therefore, there must be a good bound K for the plausible values of k. In what follows, we show that for $\alpha < 0.432$ the only plausible value is $K = 1$ (and in the range $\alpha \in (0.432, 0.5)$, the only other plausible bound is $K = 2$), and then calculate the revenue equation based on this choice.

For simplicity, we define $\ell_{A,BM,k}$ and $\ell_{H,BM,k}$ as expected losses of Adam and Helen, respectively, in case Adam goes to S_{priv} with parameter k. The corresponding profitability ratio $B_{BM,k} = \frac{\ell_{A,k}}{\ell_{H,k}}$ still needs to be less than $\frac{\alpha}{1-\alpha}$. We state that

$$\ell_{A,BM,k} = P[\text{Helen surpasses } k+1 \text{ blocks}](1 + E_{HA})$$

$$\ell_{H,BM,k} = P[\text{Adam catches up}](1 + E_{AH}),$$

where E_{HA} and E_{AH} denote the expected number of blocks found by Adam in S_{priv} given that Helen wins, and those by Helen in S_{priv} given that Adam wins, respectively. The flow of S_{priv} can be modeled as the "Gambler's ruin" problem (see [4]). Let E_H and E_A denote the expected number of blocks found by all parties in S_{priv} given that Helen wins (Adam loses) and those given that Adam wins (Helen loses), respectively. $E_{AH} = \frac{E_A-(k-1)}{2}$ and $E_{HA} = \frac{E_H-2}{2}$. We utilize Corollary 2.12 of [45], and deduce that

$$E_A = \frac{r+1}{r-1} \cdot \left(N \cdot \frac{r^N+1}{r^N-1} - i \cdot \frac{r^i+1}{r^i-1} \right) \text{ if } k > 1,\ 0 \text{ for } k > 1$$

$$E_H = \frac{r+1}{r-1} \cdot \left(N \cdot \frac{r^N+1}{r^N-1} - (N-i) \cdot \frac{r^{N-i}+1}{r^{N-i}-1} \right) \text{ for } k > 1,$$

where $r = \frac{\alpha}{1-\alpha}$ is the ratio of Adam's and Helen's probabilities of finding the next block, $i = 2$ is the starting point of the gambler's ruin, and $N = k + 1$ is its ending point. From [4], we also deduce that

$$P[\text{Adam catches up}] = \frac{\left(\frac{1-\alpha}{\alpha}\right)^2 - 1}{\left(\frac{1-\alpha}{\alpha}\right)^{k+1} - 1} \text{ for } k > 1$$

$$P[\text{Helen surpasses } k+1 \text{ blocks}] = 1 - \frac{\left(\frac{1-\alpha}{\alpha}\right)^2 - 1}{\left(\frac{1-\alpha}{\alpha}\right)^{k+1} - 1} \text{ for } k > 1.$$

By plugging the above formula, it is straightforward to calculate the profitability ratio $B_{BM,k} = \frac{\ell_{A,BM,k}}{\ell_{H,BM,k}}$. Using Equation (4) we calculate 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, by setting $K = 1$, and the attack in Algorithm 4 simplifies to the attack in Algorithm 6 as at S_{priv}, $\Delta = 0$. We note that this form of the attack as $K = 1$ may seem straightforward, but it is obtained from the more general form of the Bold mining via optimization of the revenue.

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

ALGORITHM 6: Bold Mining Attack Algorithm as $K = 1$

procedure S_{pub}
  $flag \leftarrow 0$
  while $flag = 0$ **do**
    Set $\Delta \leftarrow \min(\texttt{Last}, 1) - 1$
    $\texttt{MyCh} \leftarrow \texttt{Trun}(\texttt{PubCh}, \Delta), \texttt{Mine}(\tau)$
    if $A \twoheadrightarrow b$ **and** $\Delta = 0$ **then**
      Publish(MyCh,head)
    else if $A \twoheadrightarrow b$ **and** $\Delta = 1$ **then**
      Append(MyCh,b)
      Set $\Delta \leftarrow \Delta - 1$, $flag \leftarrow 1$
    else if $O \twoheadrightarrow b$ **and** $\Delta = 0$ **then**
      Set $\Delta \leftarrow \Delta + 1$
    end if
  end while
  Go to procedure S_{priv}
end procedure

procedure S_{priv}
  Publish(MyCh,head), Go to procedure S_{pub}
end procedure

block, which will take expected $\frac{1}{1-\alpha}$ blocks. Then, Adam needs to find a block, which will take expected $\frac{1}{\alpha}$ blocks. From Equation (1), we obtain $f_{BM} = \frac{1}{\frac{1}{1-\alpha}+\frac{1}{\alpha}} = \alpha(1-\alpha)$. Applying Equation (3), we deduce

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

Figure 5 shows the expected revenue outcome of this attack for various α values, clearly demonstrating that it outperforms the honest mining strategy for any $\alpha < 0.5$. Note that for $\forall\alpha \in [0.432, 0.5)$, $K = 1$ yields a higher benefit to the attacker than $K = 2$.

We note that [56] has previously stated that if freshness preferred is applied, mining on one block behind the head is profitable for any attacker. However, this deduction seems merely intuitive, as they do not provide any analysis of the attack. Here, our analysis will help us in our defense proposal.

Impact 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 FP 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.

6.3 Combining Attacks

Consider an attacker capable of conducting BM and OM attacks, switching between them and setting the timestamp to any time for any block he mines. Our analysis shows that the two attacks can be combined to obtain greater rewards. Let us limit the attacker to only conduct BM with $K = 1$. We consider two cases based on whether the last block is found by the attacker.

Regarding the case where the last block is found by an honest miner, note that our analyses in Sections 6.1 and 6.2 have already showed that the attacker is better off with BM instead of OM. So, in this case, the attacker

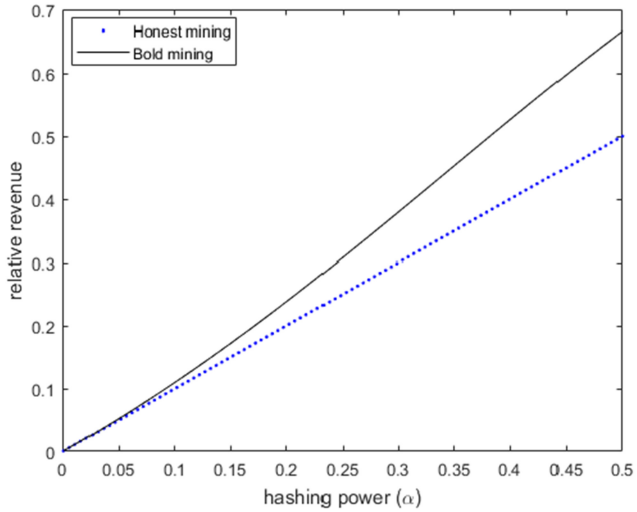


Fig. 5. Revenue from Bold mining on Decentralized FP (or Authority FP) vs. revenue from honest mining with respect to α.

will be mining on the block previous to the last one. Immediately, the question of how to set the timestamp arises. We consider separate strategies for this.

Strategy 1. Adam chooses to set the timestamp to a future time $\tau + t$, where τ is the current time (so it keeps increasing) t is a fixed time difference. If he finds a block at a specific time T; he does not publish it (unlike BM), keep extending this fork privately until $\mathsf{T} + t$ by setting the timestamps of blocks to $\mathsf{T} + t$, and releases his private chain at $\mathsf{T} + t$ (just like OM). As a result, if in the time interval $(\mathsf{T}, \mathsf{T} + t)$, Helen finds more blocks than those of Adam, all of Adam's blocks are kicked off. Otherwise, his chain kicks of Helen's chain.

Strategy 2. While bold mining on the block previous to the last one (found by Helen), Adam chooses to set its timestamp as τ (not $\tau + t$ as in Strategy 1). When he finds a block at time T, he directly publishes it kicking of Helen's last block. For the blocks that he mines in the interval $(\mathsf{T}, \mathsf{T} + t)$, he sets their timestamp as $\mathsf{T} + t$. He will mine privately until then, and release his found blocks at $\mathsf{T} + t$.

In terms of revenue, Strategy 2 beats Strategy 1 since in all the possible cases Adam could win with Strategy 1, he wins with the same reward outcome with Strategy 2. Also, if Helen wins at the time $\mathsf{T} + t$), Adam at least secures his block found at T to be on the finalized public chain.

Strategy 3. Just like Strategy 2, while bold mining on the block previous to the last one (found by Helen), Adam chooses to set its timestamp as τ (not $\tau + t$). When he finds a block at the time T, he directly publishes it kicking off Helen's last block. Unlike Strategy 2, Adam keeps oracle mining with the maximum revenue yielding t_m as shown in Section 6.1. When Helen finds a block while Adam is at S_{pub} or beats Adam at the end of S_{priv}, Adam switches back to bold mining.

Instead of a fixed future time $\mathsf{T} + t$ for oracle mining, it is better to use the timestamp $\tau + t_m$, where t_m yields the maximum revenue. Hence, Adam's revenue from Strategy 3 is greater than that from Strategy 2.

Regarding the case where the last block is found by the attacker, the attacker will continue with oracle mining. This is completely aligned with Strategy 3. We also note that although our analysis can be extended for BM with $K > 1$, BM with $K = 1$ has already shown to be much more rewarding for the attacker. Hence, for maximized revenue we propose Algorithm 7.

We deduce that the attacker's revenue is maximized by altering between BM and OM, but not by setting the timestamp to a future time while doing BM. Hence, at a given time, the attacker is actually conducting one of the

ALGORITHM 7: Combination of BM and OM

while true **do**
  Keep BM until finding a block
  Keep OM until another miner finds a block during S_{pub} or wins at the end of S_{priv}
end while

two attacks. In the rest of the article, for simplicity, we take them as separate attacks, and our defence mechanism addresses both of them (as well as their combination as in Algorithm 7).

7 OUR SM MITIGATION ALGORITHM

In the light of the analyses in Section 6, we now propose an algorithm, `Fortis`, with a tie-breaking strategy such that if applied, an attacker with $\alpha < 27.0\%$ cannot benefit from any of the SM-type attacks known to date, including OM, BM, and OSM [56]. This proposal is a relaxation of Decentralized FP, biasing some forks with respect to timestamps, unlike Uniform Tie-breaking [22]. For the sake of proper operation of `Fortis`, we assume that miners have synchronized clocks. Yet, in Section 8, we also provide results that show even with loose synchronization, our proposal enjoys the highest security. Also, we stress that synchronized clocks have already been used in proof-of-stake based consensus algorithms as in the works of [6, 37].

Intuition. Observe that both OM and BM are applicable to FP, since in case of ties the freshest block is the one that always wins. However, neither of them is possible against UT, where $\gamma = 0.5$ (yet UT prevents OSM only for adversaries with hashing power less than 0.232): Regarding OM against UT, 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 against UT, 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 probability that an honest miner chooses the fresher chain in case of a tie. FP sets $\psi = 1$ for defending against SM. UT uniformly sets $\psi = 0.5$. In essence, increasing ψ makes the system more susceptible to BM and OM, while getting better off against OSM. On the other hand, decreasing ψ provides more defense against OSM, but makes the system more susceptible to BM and OM. Therefore, we need an optimized value of $0.5 \leq \psi < 1$ that would defend against an attacker with the lowest bound of α, such that the attacker would benefit from any of OM, BM, and OSM. Let us reconsider the OM attack with the introduction of $\psi < 1$. Also, we need to consider that now the honest miners will be divided upon going back to S_{pub} with a tie. In this case 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 standard OM on FP. We state the expected block loss $\ell^{\psi}_{A,OM}$ of the attacker in S_{priv} and the expected block loss $\ell^{\psi}_{H,OM}$ of the honest parties in S_{priv} as follows:

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

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

ALGORITHM 8: 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
```

Moreover, as ES_{pub} of OM is elongated with 1 block due to the above-mentioned procedural change, we obtain from Equation (1) the expected frequency f^{ψ}_{OM} of returning S_{priv} in this attack as

$$f^{\psi}_{OM} = \frac{1}{\frac{t_m}{T_c} + 1 + \frac{1}{\alpha}} = \frac{\alpha T_c}{\alpha t_m + T_c + \alpha T_c}.$$

Also, when BM is applied on `Fortis` and a tie occurs upon going back to S_{pub}, the above-mentioned block race and additional procedure takes place. We calculate the expected block loss $\ell^{\psi}_{A,BM}$ of the attacker in S_{priv}, the expected block loss $\ell^{\psi}_{H,BM}$ of the honest parties in S_{priv}, and the expected frequency f^{ψ}_{BM} of returning S_{priv} in this attack as follows:

$$\ell^{\psi}_{A,BM} = (1-\psi)(1-\alpha), \quad \ell^{\psi}_{H,BM} = \alpha + \psi(1-\alpha),$$

$$f^{\psi}_{BM} = \frac{\alpha(1-\alpha)}{1+\alpha(1-\alpha)}.$$

We end up with an optimization problem regarding the revenues from the attacks OM, BM, and OSM, which can be stated as follows:

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

For R^{ψ}_{OM} and R^{ψ}_{BM} we apply Equation (3) with the above values, and for R^{ψ}_{OSM} we utilize the implementation given in [67]. We obtain the solution for this problem as $\alpha \approx 0.27$ at $\psi \approx 0.63$. We denote the number 0.63 as ϒ and provide our `Fortis` algorithm in Algorithm 8. 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.[7] Therefore, uniform picking results in less than $1-\Upsilon$ or ϒ chance for the attacker's chain, i.e., $\forall n > 2$ 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.

Figure 6 shows the revenues of an attacker from OM, BM, and OSM attacks applied on a PoW blockchain where honest miners practice our `Fortis` algorithm, showing that `Fortis` defends against any attacker with $\alpha < 0.27$. We note that the simulation results in Section 8 confirm our theoretical results.

[7] In case of multiple attackers, there may be multiple longest branches belonging to attackers. Yet, this has been shown as less beneficial than collusion [41, 44, 47].

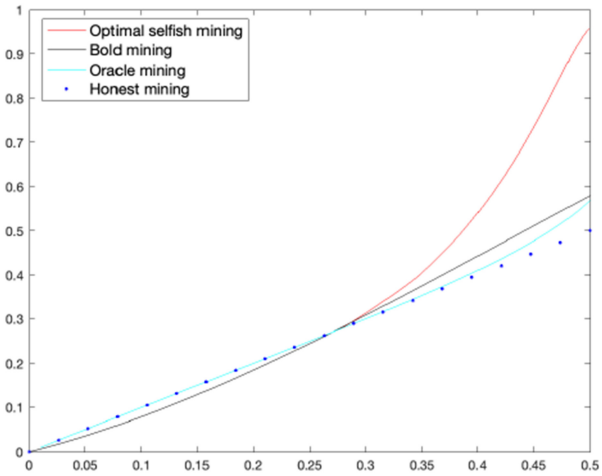


Fig. 6. The relative revenues of the optimal selfish mining of [56], our Bold mining, and our Oracle mining on our `Fortis` algorithm. The attacker cannot benefit from these attacks (and therefore honest mining is the best strategy), if his hashing power $\alpha < 27.0\%$.

8 SIMULATION

We simulate our timestamp-based Oracle and Bold mining attacks on Freshness Preferred of [31] and `Fortis`, by implementing these algorithms on top of the blockchain simulator BlockSim [2, 3]. The Python code for our simulation can be found in the repository.[8] We note that the simulation results given in this section are probabilistic and independent of the hardware platform on which the simulation runs.

Generic setup. The setup of the simulation mimics the Bitcoin protocol parameters of April 2022. Inter-block time and block size limit are set as 600 seconds and 2 MB, respectively. Regarding the block rewards, we assume a fixed block reward regime, since it is well-known that selfish mining attacks become much more minacious in the presence of transaction fees [13]. We leave the study of our attacks and defenses in this setting as future work. We enabled a single simulation common clock for the miners, as our protocol assumes a synchronized clock. Given time and computational constraints, we used 8 nodes in our simulations. But this is not a random number. We picked eight nodes that are enough for the simulation of miner pools. According to BTC.com data, the 8 pools with highest hashing power add up to 94% of total hashing power.

Propagation delay. Regarding block propagation delay, [29] states that it generally affects the outcome of selfish mining due to the fact that it is closely related to the network power γ of the attacker. The attacker would like to keep the delay low between him and an honest miner, but high between two honest miners. Considering the Bitcoin tie-breaking mechanism in April 2022 (i.e., a miner follows the first block she received), the attacker's aim is to be the first to convey his block. If he achieves this, the effect is obviously devastating, as it is effectively setting $\gamma = 1$. While somewhat counter-intiutive for a timestamp-based scheme, block propagation delays do not have much effect in our attacks and defense. This is because both our defense mechanism `Fortis` and the Freshness Preferred [31] render the network power of the attacker less useful (i.e., γ does not directly affect his revenue). Also, those delays are observed in relatively low duration with respect to the inter-block time for an attacker to obtain a high gain from.

To demonstrate the impact of propagation delay in our simulations, we model it as an exponential random variable (in accordance with the findings of [19]) with a varying average computed as follows. [16] has found that

[8] https://github.com/osmanbicer/FortisSimDLT.git

a 1 MB block is received on average in 15.7 seconds. We adjust this for each block, as this average time varies linearly with the block size [27]. For simplicity, in our simulation, a constant 2.7 transactions per second are generated and each transaction has a size of 615 bytes. We note that these values are derived from the statistics available on the websites[9] per April 2022. We assume that an attacker (Oracle or Bold miner) and seven honest nodes exist in the system to approximate the big mining pools in the real-world cryptocurrencies.[10]

Clock asynchronicity. The clock asynchronicity is a well-known problem in distributed systems and inherently in blockchains. There exist centralized (e.g., Network Time Protocol) and decentralized solutions (see [39, 46]). To show the effect of clock asynchronicity in our simulation, we assume the decentralized protocol of [39] is executed among neighboring nodes frequently. According to this work, the expected clock difference between two neighboring nodes is calculated as roughly 0.75 seconds in 12 hours. Modeling Bitcoin, we assume all nodes are not neighboring. Although we do not have any exact knowledge of the distance between Bitcoin nodes, we estimate it as follows. As of January 2021, there exist about 83000 full nodes in Bitcoin [30]. Since each full node, by default, connects to 8 separate full nodes [51]; without any overlaps, the distance between two nodes would be $\log_8 83{,}000 = 5.45$. This would mean a maximum of $0.75 \times 5.45 = 4.09$ seconds clock skew. However, as there would be inevitably overlaps, we assign random clock differences between each node up to 5 seconds.[11]

Connectivity. In the simulations with Freshness Preferred, the attacker's network connectivity is set to the same as the other nodes, since here we aim at showing the strength of our attacks, regardless of the attacker's network power. In contrast, in the simulations with `Fortis` we let the attacker's connectivity be perfect, since we aim at showing the strength of our defense. That is, the attacker receives a block as soon as it is found and the others receive each block mined by the attacker immediately.

Results. In the first simulation, we conduct the Oracle and Bold mining attacks with hashing powers between and including 1% and 50%, with 1% increments, against Freshness Preferred [31]. The simulation is repeated 5,000 times. Figure 7(a) provides attacker's expected relative revenues from each attack.

In the second simulation, we conduct the Oracle and Bold mining attacks with hashing powers between and including 1% and 50%, with 1% increments, against our `Fortis` algorithm. The simulation is repeated 5,000 times. Figure 7(b) provides attacker's expected relative revenues from each attack.

In both simulations, we attribute the small deviations of the simulation results from theory to propagation delay, connectivity, and clock skew. Still, the theoretical and simulation results are very close, which shows that our attacks are effective on Freshness Preferred. Also, our `Fortis` provides security up to the attacker's hashing power being 27% of the whole system.

9 DISCUSSION

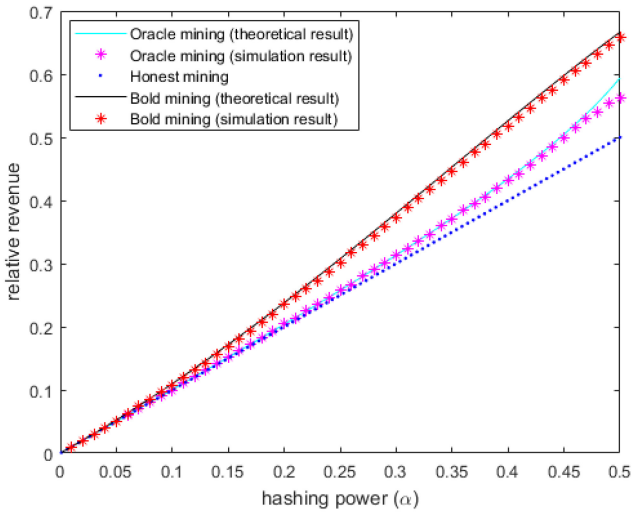
Gradual deployment. Although major changes in existing PoW blockchain systems are hard to deploy, our proposal is not too hard to be deployed due to its backward compatibility (see Section 2). That is, the blocks generated by `Fortis` will be verified by the existing miners (hardware or software), and vice versa. Slight differences on the preference of the main chain may occur in case of a tie, but this would hardly be an obstacle for co-existence of `Fortis` with a currently deployed mining algorithm such as that of Bitcoin.

We expect a gradual transfer to our solution, as honest miners are indeed incentivized to do so. We highlight that current mining algorithms are vulnerable to selfish mining and even covert exploitations have already been detected [42]. Updating the existing mining hardware would require some costs, but the more miners choose it the more simple this process will become as novel techniques for this purpose will be developed.

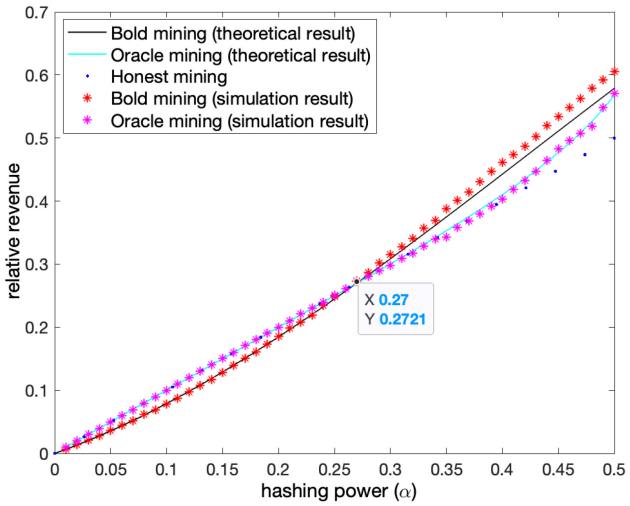
[9] https://bitinfocharts.com/bitcoin/ and https://bitcoinvisuals.com/chain-tx-size

[10] https://www.buybitcoinworldwide.com/mining/pools/

[11] We highlight that deliberately deviating from network clock, as in Oracle mining, has already been disincentivized for the attacker. Further, the effects of deliberate block delaying is covered in S_{priv} state of the attacks.



(a) Our Attacks on Freshness Preferred



(b) Our Attacks on `Fortis`

Fig. 7. Simulation results of relative revenues of the Oracle and Bold miners when the honest mining algorithm is (a) Freshness Preferred and (b) our `Fortis` algorithm. Note that the theoretical revenue calculations from the previous chapters are also provided for reference.

Practicality of selfish mining attacks. One main problem is that in practice many miners are not wary of the selfish mining issue. This is a direct result of the obscure nature of PoW cryptocurrencies, where mining takes place in high safety mining farms and by unknown identities scattered around the world. Thus, the victims of this attack may not know that they are being robbed; instead, they only think that their hashing power is low relative to the whole community (whose hashing power can only be guessed from the number of blocks found in a given time range and difficulty adjustments).

Clock synchronicity. Another issue that needs to be addressed is the use of a synchronous global clock. Although this would indeed be an additional requirement in case of Bitcoin, Ripple already uses this for its consensus mechanism in a different concept. Further, decentralized solutions using clock synchronicity [39, 46] have already been studied. We also show that our defence mechanism does not get highly affected in cases of regular delays and clock skews in our simulation.

10 CONCLUSION AND FUTURE WORK

Selfish mining attacks still continue to be a significant threat against the security and reliability of PoW blockchains. There is an ongoing research both for improving the attack effectiveness and for defending against them. In this article, we have shown the weakness of the previous timestamp based defense [31], by providing attacks even with a low hashing power. Then, we provided a backward-compatible defense algorithm `Fortis`, which is fast to achieve consensus in recovery from network partitions and is 2.0% more incentive-compatible compared to the other known defense [68]. The importance of 2.0% improvement in Bitcoin, as of April 2022, can force a selfish miner to invest an additional 19 billion dollars in hardware for receiving any unfair benefit. Although we are still far from the Nakamato's security goal of honest majority, we consider this as a step towards it. Our solution uses synchronized clocks (with skew) as in the works [6, 37].

We also provided generic formulas that can be used in the analysis of similar attacks and defenses. Without these formulas, it would be quite challenging to analyze our attacks, let alone optimizing them and proposing a defense against them. The application of our formulas for more complicated attacks, e.g., combination of Eclipse attacks with selfish mining, remains as future work. Moreover, generating a single optimized attack that combines our timestamp based attacks and the OSM strategy [56] is a non-trivial task, and is left as future work.

Depending on the setting, one may need to see more evidence for reliability of `Fortis` deployment. Simulations and analyses regarding different cases, including reward regimes with transaction fees and multiple attackers, are interesting open problems. We highlight that the development of a complete framework for provable security in blockchains would eliminate taking into account each type of attacks. Nevertheless, currently it is a highly non-trivial task, as even modeling the basic features of blockchains are challenging [25, 26], and new attacks keep being discovered at a fast pace. We leave the analysis of our attacks and defense with rational protocol design [5, 7, 24] as future work. We also leave its analysis with the cooperative game theoretic framework [12]. We also leave further improvements of our defence by incorporating ideas from other defense mechanisms, including the idea of acceptable time for a block's arrival [60], as future work. We highlight that in its given form, [60] is not backward compatible with the existing major PoW blockchains such as Bitcoin. Yet, it may be possible to apply its combination with our defense in the newly developed blockchains.

We simulated our attacks and optimal selfish mining attack [56] on the previous timestamp based scheme [31] and our defense algorithm `Fortis`. The simulations show that even with propagation delay and clock asynchronicity, practice closely follows theory.

ACKNOWLEDGMENTS

We thank Shivam Agarwal from Ashoka University, India for his help in simulation of our attacks and defense.

REFERENCES

[1] S. Ai, G. Yang, C. Chen, K. Mo, W. Lv, and A. Sandor. 2022. ESM: Selfish mining under ecological footprint. *Elsevier Information Sciences* 606 (2022), 601–613.
[2] M. Alharbi. 2020. *Blocksim Implementation*. Retrieved November 01, 2020 from https://github.com/maher243/BlockSim
[3] M. Alharby and A. van Moorsel. 2020. BlockSim: An extensible simulation tool for blockchain systems. *Frontiers in Blockchain* 3 (2020).
[4] S. E. Alm. 2002. *Simple Random Walk*. Retrieved 06 June 2022 from http://www2.math.uu.se/~sea/kurser/stokprocmn1/slumpvandring_eng.pdf

[5] C. Badertscher, J. Garay, U. Maurer, D. Tschudi, and V. Zikas. 2018. But why does it work? A rational protocol design treatment of bitcoin. In *Proceedings of the Advances in CryptologyEUROCRYPT 2018: 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques*.
[6] C. Badertscher, P. Gazi, A. Kiayias, A. Russell, and V. Zikas. 2018. Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*.
[7] C. Badertscher, Y. Lu, and V. Zikas. 2021. A rational protocol treatment of 51% attacks. In *Proceedings of the Advances in Cryptology-CRYPTO 2021: 41st Annual International Cryptology Conference*.
[8] S. Bag, S. Ruj, and K. Sakurai. 2017. Bitcoin block withholding attack: Analysis and mitigation. *IEEE Transactions on Information Forensics and Security* 12, 8 (2017), 1967–1978.
[9] L. Bahack. 2013. Theoretical bitcoin attacks with less than half of the computational power (draft). *arXiv preprint arXiv:1312.7013*. Retrieved May 19, 2022 from https://arxiv.org/abs/1312.7013
[10] G. Bissias and B. Levine. 2020. Bobtail: Improved blockchain security with low-variance mining. In *Proceedings of the NDSS'20*.
[11] O. Biçer and A. Küpçü. 2020. Anonymous, attribute based, decentralized, secure, and fair e-donation. In *Proceedings of the PETS/PoPETS'20*.
[12] O. Biçer, B. Yildiz, and A. Küpçü. 2021. m-stability: Threshold security meets transferable utility. In *Proceedings of the ACM CCSW'21*.
[13] M. Carlsten, H. Kalodner, S. M. Weinberg, and A. Narayanan. 2016. On the instability of bitcoin without the block reward. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*.
[14] Chainlink. 2021. *Chainlink's Smart Contract Research*. Retrieved June 01, 2021 from https://chainlinklabs.com
[15] V. Chicarino, C. Albuquerque, E. Jesus, and A. Rocha. 2020. On the detection of selfish mining and stalker attacks in blockchain networks. *Ann. Telecommun.* 75, 3 (2020), 143–152.
[16] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. G. Sirer, D. Song, and R. Wattenhofer. 2016. On scaling decentralized blockchains. In *Proceedings of the International Conference on Financial Cryptography and Data Security*.
[17] A. David, S. Maciej, V. Lorenzino, and P. Francesco. 2019. *Blockchain for Digital Government*. Technical Report. Publications Office of the European Union.
[18] M. Davidson and T. Diamond. 2020. On the Profitability of Selfish Mining Against Multiple Difficulty Adjustment Algorithms. Cryptology ePrint Archive, Report 2020/094.
[19] C. Decker and R. Wattenhofer. 2013. Information propagation in the bitcoin network. In *Proceedings of the IEEE P2P'13*.
[20] X. Dong, F. Wu, A. Faree, D. Guo, Y. Shen, and J. Ma. 2019. Selfholding: A combined attack model using selfish mining with block withholding attack. *Elsevier Computers & Security* 87 (2019), 101584.
[21] S. A. Elliott. 2019. Nash equilibrium of multiple, non-uniform bitcoin block withholding attackers. *Proceedings of the 2019 2nd International Conference on Data Intelligence and Security*.
[22] I. Eyal and E. G. Sirer. 2018. Majority is not enough: Bitcoin mining is vulnerable. *Communications of the ACM* 61, 7 (2018), 95–102..
[23] S. Gao, Z. Li, Z. Peng, and B. Xiao. 2019. Power adjusting and bribery racing: Novel mining attacks in the bitcoin system. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*.
[24] J. Garay, J. Katz, U. Maurer, B. Tackmann, and V. Zikas. 2013. Rational protocol design: Cryptography against incentive-driven adversaries. In *Proceedings of the IEEE 54th Annual Symposium on Foundations of Computer Science*.
[25] J. A. Garay, A. Kiayias, and N. Leonardos. 2015. The bitcoin backbone protocol: Analysis and applications. In *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*.
[26] J. A. Garay, A. Kiayias, and N. Leonardos. 2017. The bitcoin backbone protocol with chains of variable difficulty. In *Proceedings of the Annual International Cryptology Conference*.
[27] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun. 2016. On the security and performance of proof of work blockchains. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*.
[28] Google. 2021. *Certificate Transperancy project*. Retrieved March 21, 2021 from http://www.certificate-transparency.org/
[29] J. Göbel, H.P. Keeler, A.E. Krzesinski, and P.G. Taylor. 2016. Bitcoin blockchain dynamics: The selfish-mine strategy in the presence of propagation delay. *Performance Evaluation* 104 (2016), 23–41.
[30] C. Harper. 2021. *Are You Running a Bitcoin Node?* Retrieved February 02, 2021 from https://www.coindesk.com/are-you-running-a-bitcoin-node
[31] E. Heilman. 2014. One weird trick to stop selfish miners: Fresh bitcoins, a solution for the honest miner (poster abstract). In *Proceedings of the Financial Cryptography and Data Security: FC 2014 Workshops, BITCOIN and WAHC 2014*.
[32] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg. 2015. Eclipse attacks on bitcoin's peer-to-peer network. In *Proceedings of the 24th USENIX Security Symposium*.
[33] C. Hou, M. Zhou, Y. Ji, P. Daia, F. Tramèr, G. Fanti, and A. Juels. 2021. SquirRL: Automating attack analysis on blockchain incentive mechanisms with deep reinforcement learning. In *Proceedings of the NDSS'21*.
[34] M. J. Jeyasheela Rakkini and K. Geetha. 2022. Comprehensive overview on the deployment of machine learning, deep learning, reinforcement learning algorithms in Selfish mining attack in blockchain. In *Proceedings of the 2022 IEEE 2nd Mysore Sub Section International Conference*. 1–5.

[35] K. M. Khan, J. Arshad, and M. M. Khan. 2021. Empirical analysis of transaction malleability within blockchain-based e-Voting. *Elsevier Computers & Security* 100, C (2021).
[36] A. Kiayias, E. Koutsoupias, M. Kyropoulou, and Y. Tselekounis. 2016. Blockchain mining games. In *Proceedings of the 2016 ACM Conference on Economics and Computation*.
[37] A. Kiayias, A. Russell, Bernardo David, and R. Oliynykov. 2017. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Proceedings of the Annual International Cryptology Conference*.
[38] E. Koutsoupias, P. Lazos, F. Ogunlana, and P. Serafino. 2019. Blockchain mining games with pay forward. In *Proceedings of the World Wide Web Conference*.
[39] H. Kılınç-Alper. 2019. Network time with a consensus on clock. Cryptology ePrint Archive, Report 2019/1348. https://eprint.iacr.org/2019/1348
[40] S. Lee and S. Kim. 2023. Rethinking selfish mining under pooled mining. *Elsevier ICT Express* 9, 3 (2023), 356–361.
[41] T. Leelavimolsilp, L. Tran-Thanh, S. Stein, and V. H. Nguyen. 2019. Selfish mining in proof-of-work blockchain with multiple miners: An empirical evaluation. *arXiv preprint arXiv:1905.06853*. Retrieved April 8, 2022 from https://arxiv.org/abs/1905.06853
[42] S.-N. Li, C. Campajola, and C. J. Tessone. 2022. Twisted by the pools: Detection of selfish anomalies in proof-of-work mining. *arXivpreprint arXiv:2208.05748* (2022). Retrieved Jun 30, 2022 from https://arxiv.org/abs/2208.05748
[43] X. Li, P. Jiang, T. Chen, X. Luo, and Q. Wen. 2020. A survey on the security of blockchain systems. *Elsevier FGCS* 107 (2020), 841–853.
[44] H. Liu, N. Ruan, R. Du, and W. Jia. 2018. On the strategy and behavior of bitcoin mining with n-attackers. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*.
[45] P. Lorek and P. Markowski. 2018. Conditional gambler's ruin problem with arbitrary winning and losing probabilities with applications. *arXiv preprint arXiv:1812.00687* (2018).
[46] M. Maggs, S. O'Keefe, and D. Thiel. 2012. Consensus clock synchronization for wireless sensor networks. *Sensors Journal, IEEE* 12 06 (2012), 2269–2277.
[47] F. J. Marmolejo-Cossío, E. Brigham, B. Sela, and J. Katz. 2019. Competing (Semi-)selfish miners in bitcoin. *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*.
[48] S. Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system.
[49] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder. 2016. *Bitcoin and Cryptocurrency Technologies* (3rd ed.). Fundamental Algorithms, Vol. 1. Princeton University Press. (book).
[50] K. Nayak, S. Kumar, A. Miller, and E. Shi. 2016. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In *Proceedings of the 2016 IEEE European Symposium on Security and Privacy*.
[51] S. Park, S. Im, Y. Seol, and J. Paek. 2019. Nodes in the bitcoin network: Comparative measurement study and survey. *IEEE Access* 7 (2019), 57009–57022.
[52] R. Pass, L. Seeman, and A. Shelat. 2017. Analysis of the blockchain protocol in asynchronous networks. In *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*.
[53] S. Reno and S. Sultana. 2022. Preventing selfish mining in public blockchain using alarming block and block interval time approach. In *Proceedings of the 2022 International Conference on Augmented Intelligence and Sustainable Systems*. 988–993.
[54] F. Ritz and A. Zugenmaier. 2018. The impact of uncle rewards on selfish mining in ethereum. In *Proceedings of the 2018 IEEE European Symposium on Security and Privacy Workshops*.
[55] M. Saad, L. Njilla, C. A. Kamhoua, and A. Mohaisen. 2019. Countering selfish mining in blockchains. *Proceedings of the 2019 International Conference on Computing, Networking and Communications*.
[56] A. Sapirshtein, Y. Sompolinsky, and A. Zohar. 2016. Optimal selfish mining strategies in bitcoin. In *Proceedings of the Financial Cryptography and Data Security: 20th International Conference*.
[57] S. Shi, D. He, L. Li, N. Kumar, M. Khurram Khan, and K.-K. Raymond Choo. 2020. Applications of blockchain in ensuring the security and privacy of electronic health record systems: A survey. *Elsevier Computers & Security* 97 (2020), 101966.
[58] B. L. Shultz. 2015. Certification of witness: Mitigating blockchain fork attacks. (2015). Available at Retrieved from http://bshultz.com/paper/ShultzThesis.pdf
[59] P. K. Singh, R. Singh, S. K. Nandi, and S. Nandi. 2019. Managing smart home appliances with proof of authority and blockchain. In *Proceedings of the Innovations for Community Services: 19th International Conference*.
[60] S. Solat and M. Potop-Butucaru. 2017. Zeroblock: Timestamp-free prevention of block-withholding attack in bitcoin. In *Proceedings of the SSS'17*.
[61] Y. Sompolinsky and A. Zohar. 2015. Secure high-rate transaction processing in bitcoin. In *Proceedings of the Financial Cryptography and Data Security: 19th International Conference*.
[62] I. Tsabary and I. Eyal. 2018. The gap game. In *Proceedings of the 2018 ACM SIGSAC conference on Computer and Communications Security*.
[63] Taotao Wang, Soung Chang Liew, and Shengli Zhang. 2019. When blockchain meets AI: Optimal mining strategy achieved by machine learning. *arXiv preprint arXiv:1911.12942* (2019).
[64] Z. Wang, J. Liu, Q. Wu, Y. Zhang, H. Yu, and Z. Zhou. 2019. An analytic evaluation for the impact of uncle blocks by selfish and stubborn mining in an imperfect Ethereum network. *Elsevier Computers & Security* 87 (2019), 101581.

[65] Guoyu Yang, Yilei Wang, Zhaojie Wang, Youliang Tian, Xiaomei Yu, and Shouzhe Li. 2020. Ipbsm: An optimal bribery selfish mining in the presence of intelligent and pure attackers. *International Journal of Intelligent Systems* 35, 11 (2020).
[66] R. Yang, X. Chang, J. Mišić, and V. B. Mišić. 2020. Assessing blockchain selfish mining in an imperfect network: Honest and selfish miner views. *Elsevier Computers & Security* 97 (2020), 101956.
[67] R. Zhang. 2017. Retrieved March 03, 2022 from https://github.com/nirenzang/Optimal-Selfish-Mining-Strategies-in-Bitcoin
[68] R. Zhang and B. Preneel. 2017. Publish or Perish: A backward-compatible defense against selfish mining in bitcoin. In *Proceedings of the Topics in CryptologyCT-RSA 2017: The Cryptographers Track at the RSA Conference 2017.*
[69] C. Zhou, L. Xing, Q. Liu, and H. Wang. 2023. Effective selfish mining defense strategies to improve bitcoin dependability. *Applied Sciences* 13, 1 (2023).

Received 23 August 2022; revised 8 May 2023; accepted 24 June 2023