The Performance of Selfish Mining in GHOST

Qing Xia[†‡], Wensheng Dou[†‡¶*], Fengjun Zhang[†¶], Geng Liang[†¶]
[†]Institute of Software, Chinese Academy of Sciences, Beijing, China
[‡]University of Chinese Academy of Sciences, Beijing, China
[¶]State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China
[†]{xiaqing2018, wensheng, fengjun, lianggeng}@iscas.ac.cn

Abstract—**The blockchain technology is regarded as a significant trust-building technology and has attracted much attention from the public. The longest chain rule has been widely applied in blockchain systems to reach consensus on the distributed ledger. However, the longest chain rule cannot support a higher transaction throughput due to its lower security. As an alternative solution to the longest chain rule, GHOST is proposed as a safer consensus rule. Existing studies show that the longest chain rule can suffer from selfish mining attacks. However, it is unclear how selfish mining attacks perform on GHOST.**

In this paper, we explore the performance of selfish mining on GHOST. We first propose the original selfish mining (GHOST-SM) and stubborn mining (GHOST-StuM) for GHOST. We then evaluate these two selfish mining strategies on our blockchain simulation system. The experimental result shows that GHOST achieves better security than the longest chain rule. However, when the block generation rate increases, the security of GHOST is close to the longest chain rule. For example, the threshold for selfish mining attacks of GHOST is increased by 47.55% and 0.60% compared to the longest chain rule corresponding to the block generation interval of 1 second and 15 seconds.

Index Terms—**GHOST, longest chain rule, consensus, blockchain**

I. Introduction

Bitcoin is the first fully decentralized electronic payment system released by Satoshi Nakamoto in 2009. As the core technology of Bitcoin, blockchain has received great attention from both academy and industry. In the blockchain, each participant (miner) maintains a continuously-growing list of records, in which all transactions are put into so-called blocks. Each block contains a cryptographic hash pointer referring to its parent block and hence all blocks form a chain. Blockchain is regarded as a disruptive technology for secure data sharing among untrusted parties due to its characteristics of immutability and traceability. In recent years, blockchain has been successfully applied to various domains, such as financial services, cross-border trade and supply chain management.

Nakamoto consensus [1] is one of the most widely-used protocols to guarantee the consistency and security of blockchain. In Nakamoto consensus, all miners compete against each other to generate new blocks through finding the solution of the PoW (Proof of Work) puzzle. Due to the randomness of the PoW puzzle, miners may generate conflicting blocks at the same height. Hence, all blocks form a block tree instead of a single chain. To guarantee consistency, Nakamoto consensus adopts the longest chain rule that regards the longest chain of the block tree as the main chain. Only the blocks in the main chain are valid and accepted by all miners. The blocks outside of the longest chain are discarded as orphan blocks, in which the transactions will be put back into the waiting area, i.e., the transaction memory pool. To compensate for miners' computing and storage cost, the blockchain system generally provides some cryptocurrency rewards for the miners who generate the main chain blocks.

[*]Corresponding author.

To improve transaction throughput, blockchain systems usually decrease their block generation intervals to speed up the block generation rate [2]–[5]. However, as the block generation interval decreases, the security of the longest chain rule against double-spend attack is also decreased below the theoretical security boundary, i.e., 50%. To address the security concern, GHOST [6] is proposed as an alternative solution to the longest chain rule. GHOST selects the heaviest chain instead of the longest chain when the blockchain forks. Since orphan blocks still contribute to the weight of the main chain, GHOST achieves better security. Previous study shows that the threshold of GHOST against double-spend attack always maintains at 50% even when the orphan rate varies [6].

Existing studies [7], [8] show that the longest chain rule can suffer from various selfish mining attacks, e.g., original selfish mining [6] and stubborn mining [9]. Instead of broadcasting new blocks immediately, a selfish miner first withholds new blocks and reveals them later based on specific strategies. By deliberately forking the blockchain with hidden blocks, the selfish miner forces honest miners to waste their mining power on the stale states and hence obtains a higher revenue. The experimental result shows that a selfish miner with 33% fraction of mining power can gain more revenue than its fair share. Therefore, the security boundary of the longest chain rule is regarded as 33% under the selfish mining attacks. However, no study has evaluated selfish mining on GHOST. How does GHOST perform in selfish mining? Does GHOST have better security than the longest chain rule under selfish mining? These questions have not been answered yet.

In this paper, we present the first study to explore the performance of selfish mining on GHOST. We first propose two selfish mining strategies for GHOST, i.e., GHOST-SM (original Selfish Mining for GHOST) and GHOST-StuM (Stubborn Mining for GHOST). Second, we evaluate the performance of these two mining strategies on our blockchain simulation system. Finally, we compare the security of the GHOST rule and the longest chain rule. From our study, we obtain some

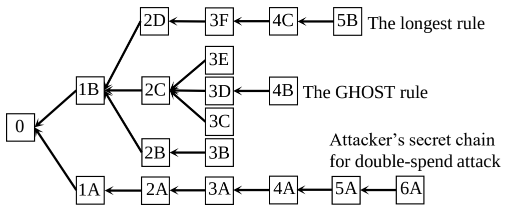


Fig. 1. The longest chain rule and the GHOST rule.

interesting findings. First, GHOST can also suffer from selfish mining attacks, e.g., the security boundary of GHOST is 24.86% when the block generation interval is 15 seconds. Second, GHOST has better security than the longest chain rule, especially in the system with a short block generation interval, e.g., the security boundary of the longest chain rule and GHOST is 7.15% and 10.55% when the block generation interval is 1 second.

We summarize the main contributions as follows.

- We propose two selfish mining strategies for GHOST.
- We evaluate the selfish mining strategies for GHOST, and observe some interesting finding.
- We find that GHOST can still suffer from selfish mining, and its security boundary is 24.86% when the block generation interval is 15 seconds.

II. Related Work

Some research efforts have been put on exploring new selfish mining strategies after selfish mining for the longest chain rule is first proposed by Eyal and Sirer [7]. Nayak et al. [9] proposed stubborn mining. Sapirshtein et al. [10] optimized selfish mining in a wider parameter space. Niu and Feng [11], Ritz and Zugenmaier [12] considered the impact of uncle reward on selfish mining in Ethereum, which shows that Ethereum is more vulnerable to selfish mining than Bitcoin.

Some work evaluated the practical performance of selfish mining in more complex scenarios. Liu et al. [13] and Bai et al. [14] considered multiple selfish miners, which shows that there exists unexpected competition among them. Göbel et al. [15] considered the effect of propagation delay, which shows that the selfish miner performs better in the system with a larger delay. Xia et al. [8] evaluated the performance of selfish mining in the practical scenarios with multiple selfish miners, multiple honest miners and varied propagation delay. Negy et al. [16] considered the effect of difficulty adjustment, which found the selfish miner can adopt the intermittent selfish mining to gain more revenue.

However, all of these works are proposed for the longest chain rule. Currently, there is no research on selfish mining in GHOST. In this paper, we first propose the selfish mining strategies for the GHOST rule, including GHOST-SM and GHOST-StuM.

III. Blockchain Consensus

In this section, we briefly introduce the longest chain rule and the GHOST rule.

A. The Longest Chain Rule

The two most popular blockchain systems, i.e., Bitcoin and Ethereum, both utilize the longest chain rule to guarantee consistency and security. To achieve consistency, all miners regard the longest chain as the main chain when the blockchain forks. To achieve security, only the attacker with at least half of the mining power can generate an alternative longest chain to subvert the main chain [1]. Hence, the security boundary of the longest chain rule is regarded as 50%.

The security boundary of the longest chain rule can be reduced when the blockchain system scales to support a high volume of transactions via decreasing the block generation interval [2]–[5], i.e., from 15 seconds to 1 second. However, a short block generation interval results in a high orphan rate since more conflicting blocks are generated. In such a system, the security boundary of the longest chain will not hold as 50%.

Fig. 1 shows the block tree generated in the blockchain system with a high orphan rate. According to the longest rule, the block list (0, 1B, 2D, 3F, 4C, 5B) is the main chain. In this scenario, the attacker only needs less than 50% of mining power to launch the double-spend attack to subvert the main chain. Specifically, the attacker first constructs two transactions from the same sender to different recipients, e.g., tx is from the attacker's address to the merchant's address, while tx' is from the attacker's address to the attacker's another address. Then, the attacker broadcasts tx, waiting for it to be included by the main chain block, e.g., block 1B. Meanwhile, the attacker includes tx' in its secret block, e.g., block 1A. Finally, when the merchant accepts tx after several confirmations in the main chain, the attacker publishes its longer secret chain (0, 1A, 2A, 3A, 4A, 5A, 6A) to invalidate tx and gets its money back. The attack is successful because honest miners waste their mining power on the orphan blocks, and only a small fraction of mining power contributes to the longest chain.

B. The GHOST Rule

As an alternative consensus rule to select the main chain, GHOST (the Greedy Heaviest-Observed Sub-Tree) is proposed to address the security concern of the longest chain rule. In GHOST, each block has the weight information. When a new block is generated, its weight is initialized to 1, and the weight of its parent block and all ancestor blocks is added by 1. When the blockchain forks, GHOST selects the heaviest block at each height. All heaviest blocks form the main chain. For example, in Fig. 1, the block list (0, 1B, 2C, 3D,4B) is regarded as the main chain according to the GHOST rule. Note that, if there exists multiple blocks with the same weight, miners will randomly work after the first arriving one.

Even in the system with a high orphan rate, GHOST can always maintain its security boundary against double-spend attack at 50%. The reason is that, the abandoned orphan blocks in GHOST still contribute to the weight of the main chain. For example, when the attacker publishes its secret chain, the main chain block 2C will not be subverted by the secret block 2A since the orphan blocks 3E and 3C increase its weight. Due

TABLE I
THE SELFISH MINER'S ACTIONS IN THE ORIGINAL SELFISH MINING FOR THE LONGEST CHAIN RULE.

lead	Generate a block	Receive new block(s)
0	withhold	update
0'	release new	update
1	withhold	release(1)
2	withhold	release(2)
≥ 3	withhold	release(1)

TABLE II
THE SELFISH MINER'S ACTIONS IN THE STUBBORN MINING FOR THE LONGEST CHAIN RULE.

lead	Generate a block	Receive new block(s)
-1	**release new**	**update**
0	withhold	update
0'	**withhold**	update
1	withhold	release(1)
2	withhold	**release(1)**
≥ 3	withhold	release(1)

to the security property, GHOST has been widely applied as the main chain selection rule in various consensus protocols, such as Fruitchains [17], Conflux [18] and Monoxide [19].

IV. Selfish Mining

Eyal and Sirer found that the longest chain rule can suffer from the selfish mining attack, which lowers its security boundary to 33% [7]. Then, Nayak et al. proposed stubborn mining, a more effective mining strategy against the longest rule [9]. In this section, we briefly introduce two selfish mining strategies. For clarity, we use SM to denote the original selfish mining [7], and StuM to denote stubborn mining [9].

A. Original Selfish Mining

The selfish miner can gain more reward through deviating from the consensus rule. In the blockchain system, honest miners always obey the consensus rule. Specifically, when generating a new block, they broadcast it immediately. When receiving a new block, they update their local blockchain and mine on the updated main chain. However, when generating a new block, the selfish miner withholds it firstly. When receiving a new block, the selfish miner publishes its private blocks according to the strategy. Through forcing the honest miners to waste their mining power on the stale main chain, the selfish miner obtains more reward than its fair share. Simply, we refer to the selfish miner as *Bob* and the group of honest miners as *Alice*. Note that, *Alice* actually represents all honest miners, while *Bob* behaves as a single selfish miner as in the previous works [7], [9].

In selfish mining, *Bob* takes actions according to its lead advantage. Under the longest chain rule, the lead advantage is the length difference between *Bob's* branch and *Alice's* branch, which is described as follows.

$$lead = length(Bob's\ branch) - length(Alice'\ branch) \quad (1)$$

Table I summarizes *Bob's* actions in SM, which are driven by two events, i.e., generating a new block and receiving new blocks. Note that, a miner can generate only one block at a time. However, it can receive multiple blocks meanwhile due to the propagation mechanism of the peer-to-peer protocol [20]. Both $lead = 0$ and $lead = 0'$ denote that *Bob* has no private block. The difference is that there is a unique longest chain in $lead = 0$, while a fork happens in $lead = 0'$. According to the lead advantage, *Bob* takes the following four actions when an event happens.

Withhold. When generating a new block, *Bob* withholds it and keeps working after the new block.

Update. When receiving the new blocks, *Bob* updates its blockchain and works after the updated main chain.

Release new. When generating a new block, *Bob* publishes it to win the longest chain.

Release(1). When receiving the new blocks, *Bob* releases one private block to catch the length of the honest branch and create a fork.

Release(2). When receiving the new blocks, *Bob* releases two private blocks to exceed the length of the honest branch and win the longest chain.

Fig. 2 shows an example of the original selfish mining for the longest chain rule. The orange block denotes Alice's public blocks. The white block, blue block with symbols denote Bob's public blocks and private blocks, respectively. At first, *Bob* withholds four secret blocks. When receiving *Alice's* two blocks, *Bob* creates a fork intentionally by releasing its first private block S1. Hence, there are three branches. According to the consensus rule, when there are multiple branches, honest miners work after the first arrived one. We call *Bob's* ability to propagate its block faster than others as its network advantage, denoted as γ. Therefore, when a fork happens, γ fraction of honest power mines on *Bob's* branch.

At $lead = 3$, when receiving two new blocks, *Bob* continues to create branches by releasing the private block S2. At $lead = 2$, when a new block extends the honest branch, *Bob's* lead advantage decreases to one block. Hence, *Bob* releases the rest two blocks together, i.e., block S3 and S4, to win the longest chain. Existing study [7] shows that a miner owning 33% of mining power can benefit from the original selfish mining when it has no network advantage, i.e., $\gamma = 0$.

B. Stubborn Mining

Stubborn mining (StuM) is a variant of the original selfish mining (SM). Compared to the original selfish mining, the revenue of stubborn mining can increase by up to 25% [9]. Stubborn mining consists of three sub-strategies, i.e., lead stubborn, equal-fork stubborn and trial stubborn. Table II summarizes *Bob's* actions in StuM, in which the bold actions denote its differences between SM. We describe the details of three sub-strategies as follows.

Lead Stubborn. In SM, when *Bob* has two private blocks ($lead = 2$), it will publish both blocks when the its lead decreases to one block. However, following the idea of "catch the length" instead of "win the length", *Bob* only publishes the first block. As Fig. 3 shows, in lead stubborn, *Bob* publishes block S1 to create two branches. Instead of winning the longest

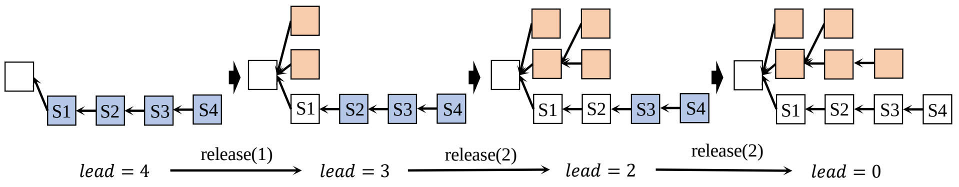


Fig. 2. An example of the original selfish mining for the longest rule. The orange block denotes Alice's public blocks. The white block with symbols denotes Bob's public blocks. The blue block with symbols denotes Bob's private blocks.

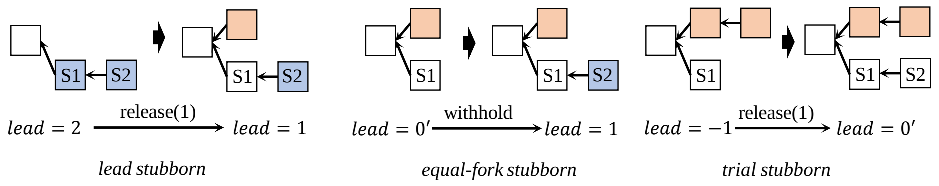


Fig. 3. The example of stubborn mining for the longest chain rule.

chain, *Bob* tries to create as many forks as possible to waste honest mining power.

Equal-fork Stubborn. In SM, when a fork happens ($lead = 0'$), *Bob* will release the newly generated block to solve the fork and win the longest chain. However, following the idea of "catch the length" instead of "win the length", *Bob* withholds the new block. As Fig. 3 shows, in equal-fork stubborn, *Bob* withholds the new block S2 to extend the fork time. Instead of winning the longest chain, *Bob's* goal is to make the fork last longer to waste the honest mining power.

Trial Stubborn. In SM, when the honest branch becomes the longest chain, *Bob* will give up its shorter chain. However, following the idea of "insist on the shorter selfish chain", *Bob* works on its shorter chain if it just lags behind by one block and hopes to catch up. This situation is denoted by the negative $lead = -1$. Instead of extending the longest chain, *Bob's* goal is to make more of its blocks included by the main chain. Note that, at $lead = -1$, when *Bob* luckily generates a new block, e.g., block S2, to catch up from behind, to create a fork. However, there is no honest power mining on the selfish branch, i.e., $\gamma = 0$, since all honest miners have accepted the earlier produced branch.

All the stubborn mining strategies have their pros and cons. For the lead stubborn and equal-fork stubborn, *Bob* has the chance to create more forks while has the risk of losing the longest chain. For the trial stubborn, *Bob* has the chance to gain more reward while has the risk of doing useless work.

V. Selfish Mining in GHOST

The GHOST rule has been widely applied as the backbone in various consensus protocols [17]–[19]. In this section, we describe the original selfish mining and stubborn mining that are specially designed for the GHOST rule, i.e., GHOST-SM and GHOST-StuM.

A. Original Selfish Mining

As mentioned before, GHOST selects the heaviest block at each height to form the main chain. Therefore, to waste the honest mining power, *Bob* aims to create a branch, in which all blocks have the same weight as other honest blocks at the same height. *Bob* takes actions according to its lead advantage. As Equation (2) shows, the lead advantage is the minimum weight difference between the blocks in *Bob's* branch and the blocks at the same height in *Alice's* branch.

TABLE III
The selfish miner's actions in the original selfish mining for the GHOST rule.

lead	Generate a block	Receive new block(s)
0	withhold	update
0'	release new	update
1	withhold	release(1)
2	withhold	release(2)
≥ 3	withhold	release(n), s.t., weight(Bob's block) $\geq$ weight(Alice's block)

$$\begin{aligned} lead = Min(weight(B_i) - weight(A_i)),\\ B_i \in Bob's\ branch, A_i \in Alice's\ branch,\\ height(B_i) = height(A_i) \end{aligned} \quad (2)$$

Table III summarizes *Bob's* actions in GHOST-SM. At $lead = 0/0'$, *Bob* has no private block and hence takes the same action as in the longest chain. At $lead \geq 1$, when generating the new block, *Bob* withholds the blocks to extend its private chain. At $lead = 1$, when receiving new blocks, *Bob* only releases one block and tries to create a branch. At $lead = 2$, when receiving new blocks, *Bob* releases both blocks and tries to win the heaviest chain. At $lead = 3$, *Bob* releases n blocks to ensure the weight of its block is not less than the honest blocks.

Fig. 4 shows an example of the state transition in GHOST-SM. At first, *Bob* withholds four private blocks. When receiving *Alice's* two blocks, *Bob* releases its first private block S1 to match the weight of honest blocks. Hence, there are three branches and some honest miners will work after block S1. At $lead = 3$, after receiving two new blocks, the weight of block H1 increases to 3. To match the weight of block H1, *Bob* releases two private blocks, i.e., S2 and S3. Hence, the weight of the block S1, S2 and S3 changes to 3, 2, 1,

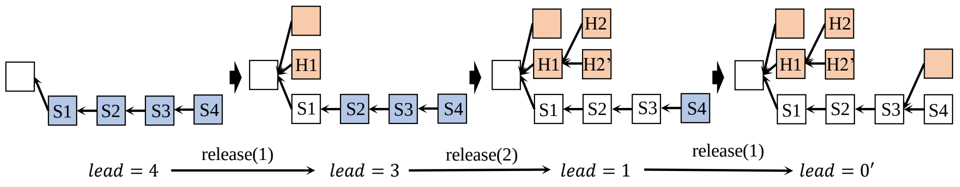


Fig. 4. An example of the original selfish mining in the GHOST rule.

TABLE IV
THE SELFISH MINER'S ACTIONS IN THE STUBBORN MINING FOR THE GHOST RULE.

lead	Generate a block	Receive new block(s)
-1	**release new**	**update**
0	withhold	update
0'	**withhold**	update
1	withhold	release(1)
2 ≥ 3	withhold	**release(n), s.t., weight(Bob's block) $\geq$ weight(Alice's block)**

respectively. According to the GHOST rule, the block list (S1, S2, S3) forms the unique main chain since the block S2 has the maximum weight at the same height. Although *Bob* aims to create more branches to waste the honest mining power, sometimes it may solve the fork accidentally due in GHOST. We call the situation *Unexpected Fork Solving*. Later, at $lead = 1$, when receiving one honest block, *Bob* releases the rest private block S4 and the lead changes to 0'.

B. Stubborn Mining

Similar to the longest chain rule, GHOST can also suffer from stubborn mining, i.e., GHOST-StuM, including lead stubborn, equal-fork stubborn and trial stubborn. Table IV summarizes *Bob's* actions in GHOST-StuM, in which bold actions denote its differences between GHOST-SM. We describe the details of three sub-strategies and discuss its difference between the longest chain rule as follows.

Lead Stubborn. In GHOST-SM, when *Bob* has two private blocks ($lead = 2$), it will publish both blocks when receiving the honest block. However, following the idea of "catch the weight" in lead stubborn, *Bob* only publishes blocks to make the weight of its branch not less than the honest branch. This means that sometimes *Bob* only needs to publish one block. *Bob* adopts lead stubborn to create as many forks as possible.

Equal-fork Stubborn. In GHOST-SM, when a fork happens ($lead = 0'$), *Bob* will release the newly generated block to solve the fork and win the heaviest chain. However, following the idea of "catch the weight" in equal-fork stubborn, *Bob* will withhold the new block. *Bob* adopts the equal-fork stubborn to make the fork last longer.

Trial Stubborn. In GHOST-SM, when the honest branch becomes the heaviest chain, *Bob* will give up its lighter chain. However, following the idea of 'insist on the lighter selfish chain" in trial stubborn, *Bob* works on its lighter chain if it just lags behind by one block weight, i.e., $lead = -1$. *Bob* adopts the trial stubborn to make more of its blocks included by the main chain.

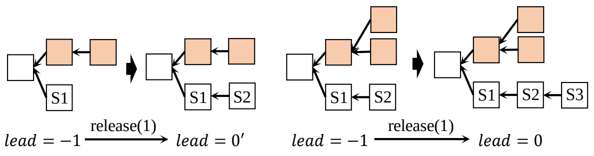


Fig. 5. Two different cases of the trial stubborn mining for the GHOST rule.

TABLE V
PARAMETERS OF THE BLOCKCHAIN SIMULATION SYSTEM.

Parameter	Description	Values
i	Block generation interval	$1 \sim 15$ seconds
α	Selfish miner's mining power	$1\% \sim 40\%$
γ	Selfish miner's network advantage	0.5

Although the idea of trial stubborn for GHOST is similar to that for the longest chain rule, their state transition is different. In the longest chain rule, when *Bob* catches up from behind, its lead advantage changes from -1 to 0' since there are two branches with the same length. However, the state transition is undetermined in GHOST. Fig. 5 shows two different cases of the state transition in trial stubborn for the GHOST rule, both start from $lead = -1$. In the first case, after generating block S2, the lead changes to 0' since there are two branches with the same weight. However, in the second case, after generating block S3, the lead changes to 0 since there is only one heaviest chain and *Bob* withholds no private block.

VI. EVALUATION

We evaluate the performance of original selfish mining and stubborn mining for the longest chain rule and the GHOST rule on the blockchain simulation system to study the following two research questions.

RQ1: *How does selfish mining perform in GHOST?*

RQ2: *Is GHOST more secure than the longest chain rule?*

We simulate the blockchain system and describe performance measurements (Section VI-A). To answer RQ1, we evaluate the performance of GHOST-SM and GHOST-StuM (Section VI-B). To answer RQ2, we compare the performance of GHOST and the longest chain rule (Section VI-C).

A. Experimental Setup

We use the Monte Carlo simulator to simulate the blockchain system, which involves three parameters. As Table V shows, i denotes the block generation interval. To evaluate the performance of GHOST especially in the system with a high throughput, we change i from 1 to 15 seconds. α denotes the selfish miner's mining power. Since the largest mining pool that ever appeared takes about 40% of the network mining power [21], we change α from 1% to 40%. γ denotes the selfish miner's network advantage ranging from 0 to 1. In the experiment, we fix the γ to the mean value of 0.5.

We simulate 1,000 miners equally sharing the total mining power as previous work [7]. The block generation process is simulated with the geometric distribution [22]. When the selfish miner *Bob* owns α fraction of mining power, it actually controls $1000 \cdot \alpha$ miners to form a pool for selfish mining. Due to the network advantage, When the fork happens, γ fraction of honest mining power will mine on the selfish branch.

Experimental measurement. In each simulation, all miners generate 100,000 main chain blocks. We iterate the simulation 10 times to avoid potential randomness. Suppose miner $m_i (i = 1, ..., 1000)$ generates MB_i main chain blocks, i.e., $\sum_{i=1}^{1,000} MB_i = 100,000$. Hence, *Bob's* selfish mining revenue under a certain mining strategy is calculated as follows. *Alice's* honest mining revenue is calculated in the same way.

$$R_{Strategy} = \sum_{i=1}^{1000 \cdot \alpha} MB_i / 100,000 \tag{3}$$

When *Bob* behaves honest, its mining revenue equals to the mining power. However, *Bob's* selfish mining revenue deviates from its fair share. As Equation (4) shows, the minimum mining power required by *Bob* to gain more revenue than its power is called *profit threshold*, which represents the security boundary of the blockchain system.

$$profit\ threshold = min(\alpha), s.t., R_{Strategy} \geq \alpha \tag{4}$$

To compare the performance between the original selfish mining and stubborn mining, we define the relative revenue of the stubborn mining as follows.

$$relative\ revenue = (R_{StuM} - R_{SM})/R_{SM} \tag{5}$$

B. RQ1: How does selfish mining perform in GHOST?

In this experiment, we evaluate GHOST-SM and GHOST-StuM in different blockchain systems. Fig. 6 shows the system throughput and orphan rate with ranging block generation intervals. A shorter block generation interval results in a higher throughput since blocks are generated faster. However, the orphan rate also increases as the number of conflicting blocks increases. For example, when the block interval decreases from 15 seconds to 1 second, the throughput increases from 6.45 TPS to 63.23 TPS [*], while the orphan rate increases from 3.24% to 36.77%.

Fig. 7 shows *Bob's* selfish mining revenue from GHOST-SM and GHOST-StuM in different systems. The black line denotes *Bob's* honest mining revenue. The blue line and orange line denote *Bob's* selfish mining revenue in the system whose block interval is 1 second and 15 seconds. The experimental result shows three interesting findings. First, GHOST can also suffer from selfish mining since *Bob* always gains more

[*]We assume each block contains 100 transactions averagely

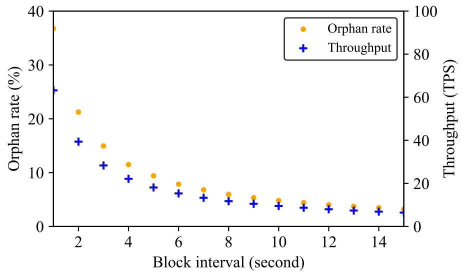


Fig. 6. Throughput and orphan rate of different blockchain systems.

revenue with more mining power. Second, no matter which mining strategy is adopted, i.e., GHOST-SM and GHOST-StuM, *Bob* gains more revenue in the system with a shorter block generation interval. For example, with 30% of mining power, when the block generation interval decreases from 15 seconds to 1 second, *Bob's* revenue increases by 9.30% and 8.60% corresponding to the GHOST-SM and GHOST-StuM. The reason is that a shorter block generation interval causes a higher orphan rate. Therefore, honest miners waste more mining power on these orphan rates. Third, With more mining power, *Bob's* mining revenue from GHOST-StuM is higher than that from GHOST-SM. For example, with 30% of mining power, when the block generation interval is 1 second, *Bob's* revenue from GHOST-StuM improves 2% than GHOST-SM.

Fig. 8 shows *Bob's* relative revenue from GHOST-StuM. The experimental result shows two interesting findings. First, no matter what block generation interval is, with enough mining power, *Bob* can gain more revenue from GHOST-StuM compared to GHOST-SM. The right side of the figure shows that the revenue of GHOST-StuM generally increases by 10% when *Bob's* mining power exceeds 35%. Second, the white space at the bottom left of the figure shows that small miners loses a lot from GHOST-StuM, especially in the system with a shorter interval. To understand which stubborn mining strategy benefits most, we further analyze the frequency and success rate of three stubborn strategies.

Fig. 9 shows the frequency and success rate of three stubborn strategies in GHOST-StuM, i.e., lead stubborn, equal-fork stubborn and trial stubborn. It shows that when *Bob's* mining power increases, the frequency and success rate of all strategies increases. The left figure shows that the frequency of equal-fork stubborn is the highest, which is much greater than lead stubborn and trial stubborn. For example, when generating 100,000 main chain blocks, with 40% of mining power, *Bob* launches stubborn mining 32,166 times averagely. Among them, equal-fork stubborn, lead stubborn and trial stubborn account for 74.10%, 14.24% and 11.66%. The frequency of equal-fork stubborn is highest since *Bob* generally needs only one private block to create a fork and launch the strategy. By contrast, *Bob* needs two private blocks to launch lead stubborn. The trial stubborn has the lowest frequency because it only happens when *Bob* just falls behind one block weight.

The right side of Fig. 9 shows that the success rate of lead stubborn, equal-fork stubborn and trial stubborn are highest,

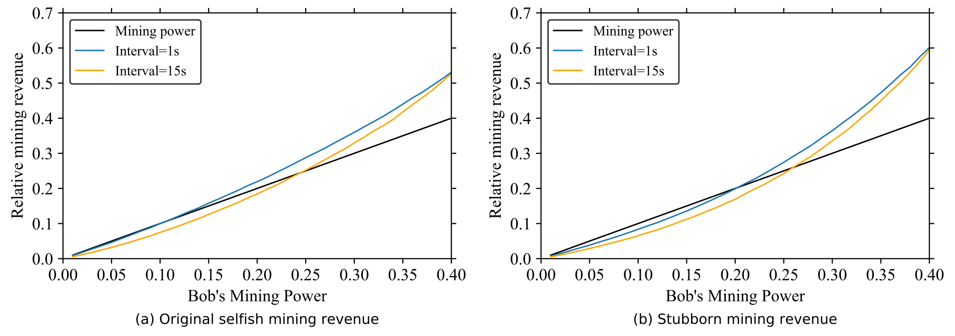


Fig. 7. Bob's selfish mining revenue from GHOST-SM and GHOST-StuM.

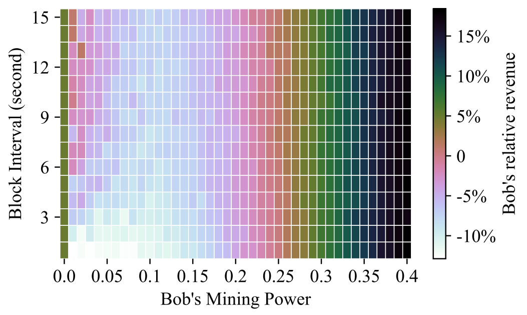


Fig. 8. Relative revenue of GHOST-StuM.

second and lowest, respectively. The reason is that *Bob* is more likely to succeed with more private blocks. In lead stubborn, *Bob* still withholds one private block after releasing one. In equal-fork stubborn, *Bob* has no private block. However, In trial stubborn, *Bob* falls behind by one block weight. There is another reason for the lowest success rate of the trial stubborn. That is, even if *Bob* succeed to generate a new block and create a branch, there is no honest power mining on its branch since it just catches up.

C. RQ2: Is GHOST more secure than the longest chain rule?

In this experiment, we compare the performance of selfish mining between GHOST and the longest chain rule. A consensus rule is considered more secure if it has a higher profit threshold in selfish mining. Fig. 10 shows the profit threshold of the original selfish mining and stubborn mining for these two consensus rules. We observe two interesting findings.

First, in both mining strategies, GHOST is more secure than the longest chain rule, especially in the system with a short block generation interval. However, as the block generation interval increases, their security gap is narrowing. For example, when the block generation interval is 1 second, in the original selfish mining, the profit threshold of the longest chain rule and GHOST are 7.15% and 10.55%. The security of GHOST is increased by 47.55%. However, when the block generation interval increases to 15 seconds, the profit threshold of two rules is 24.70% and 24.85%. The security of GHOST is just increased by 0.6%. We discuss the reason as follows.

With a shorter block generation interval, the number of conflicting blocks increases. In this scenario, the selfish miner in GHOST frequently encounters the situation of *Unexpected Fork Solving* (See Section V-A). Specifically, although the selfish miner aims to create forks and extend the fork time, it sometimes solves the fork unexpectedly according to the GHOST rule. Therefore, the selfish miner needs more mining power to benefit from GHOST than the longest chain rule. As the block generation interval increases, the frequency of *Unexpected Fork Solving* decreases. Therefore, the profit threshold of two rules becomes close.

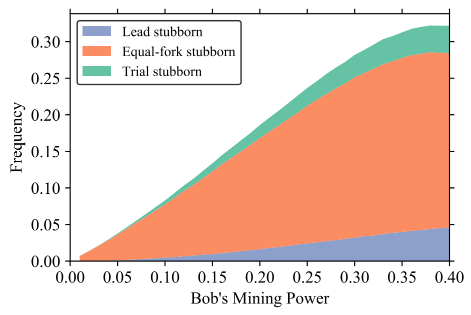
The second observation is that, in both consensus rules, the profit threshold of stubborn mining is higher than the original selfish mining. For example, when the block generation interval is 1 second, the profit threshold is 10.55% and 20.90% corresponding to GHOST-SM and GHOST-StuM, and it is 7.15% and 17.30% for SM and StuM. The reason is that the selfish miner in stubborn mining gives up many chances to win the chain. Fig. 11 shows the selfish mining revenue of two consensus rules, which confirms the observation. Since the longest chain rule has a lower profit threshold than GHOST, no matter what strategy is adopted, it brings more reward to the selfish miner than GHOST.

VII. Conclusion

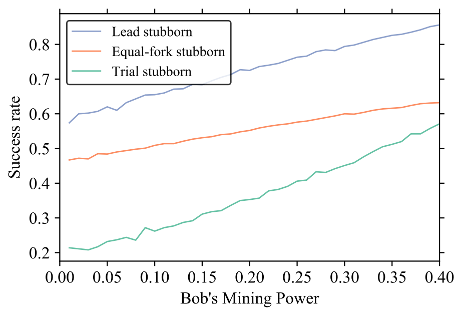
As an alternative to the longest chain rule in Nakamoto consensus, GHOST is regarded as a safer consensus rule. In this paper, we propose two selfish mining strategies for GHOST, and evaluate them on the simulation system. The experimental result shows that GHOST can still suffer from selfish mining. Compared with the longest chain rule, GHOST has better security in the system with a short block generation interval.

Acknowledgement

This work was partially supported by National Key Research and Development Program of China (2020YFC1523203), Strategic Priority Research Program of Chinese Academy of Sciences (XDA20080200), National Natural Science Foundation of China (62072444), Blockchain Technology and Application Joint Laboratory, Guiyang Academy of Information Technology (Institute of Software Chinese Academy of Sciences Guiyang Branch), and Youth Innovation Promotion Association at Chinese Academy of Sciences (2018142).

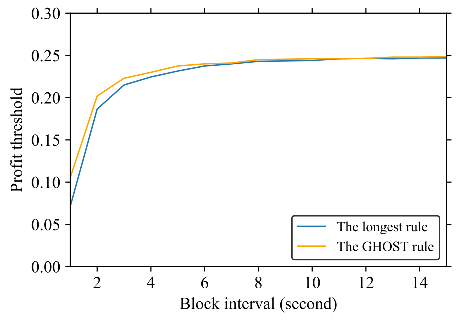


(a) Frequncy of three stubborn mining

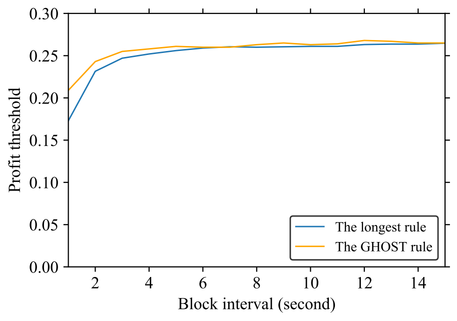


(a) Success rate of three stubborn mining

Fig. 9. Frequency and the success rate of three stubborn strategies in GHOST-StuM



(a) Original selfish mining



(b) Stubborn mining

Fig. 10. The profit threshold of two consensus rules.

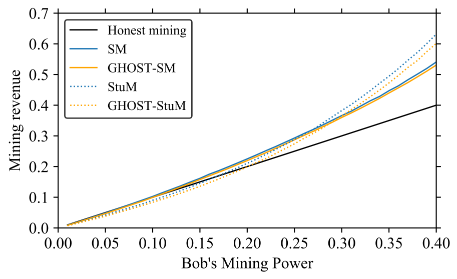


Fig. 11. Selfish mining revenue of two consensus rules.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Manubot, Tech. Rep., 2019.

[2] (2020) Litecoin block generation. [Online]. Available: https://bitcointalk.org/index.php?topic=47417.0

[3] (2020) Dogecoin block generation. [Online]. Available: https://github.com/dogecoin/dogecoin

[4] (2020) Bitcoin cash. [Online]. Available: https://en.wikipedia.org/wiki/Bitcoin_Cash

[5] (2020) Bitcoin sv. [Online]. Available: https://bitcoinsv.io/2018/10/21/bitcoin-sv-version-0-1-goes-live/

[6] Y. Sompolinsky and A. Zohar, "Secure high-rate transaction processing in bitcoin," in *Proceedings of International Conference on Financial Cryptography and Data Security (FC)*, 2015, pp. 507–527.

[7] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," in *Proceedings of International conference on financial cryptography and data security (FC)*, 2014, pp. 436–454.

[8] Q. Xia, W. D. Dou, T. Xi, J. Zeng, F. Zhang, J. Wei, and G. Liang, "The impact analysis of multiple miners and propagation delay on selfish mining," in *Proceedings of IEEE Computers, Software, and Applications Conference (COMPSAC)*, 2021.

[9] K. Nayak, S. Kumar, A. Miller, and E. Shi, "Stubborn mining: Generalizing selfish mining and combining with an eclipse attack," in *Proceedings of IEEE European Symposium on Security and Privacy (EuroS&P)*, 2016, pp. 305–320.

[10] A. Sapirshtein, Y. Sompolinsky, and A. Zohar, "Optimal selfish mining strategies in bitcoin," in *Proceedings of International Conference on Financial Cryptography and Data Security (FC)*, 2016, pp. 515–532.

[11] J. Niu and C. Feng, "Selfish mining in Ethereum," *Proceedings of IEEE International Conference on Distributed Computing Systems (ICDCS)*, pp. 1306–1316, 2019.

[12] F. Ritz and A. Zugenmaier, "The impact of uncle rewards on selfish mining in Ethereum," *Proceedings of IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pp. 50–57, 2018.

[13] H. Liu, N. Ruan, R. Du, and W. Jia, "On the strategy and behavior of bitcoin mining with n-attackers," in *Proceedings of Asia Conference on Computer and Communications Security (CCS)*, 2018, pp. 357–368.

[14] Q. Bai, X. Zhou, X. Wang, Y. Xu, X. Wang, and Q. Kong, "A deep dive into blockchain selfish mining," in *Proceedings of IEEE International Conference on Communications (ICC)*, 2019, pp. 1–6.

[15] J. Göbel, H. P. Keeler, A. E. Krzesinski, and P. G. Taylor, "Bitcoin blockchain dynamics: The selfish-mine strategy in the presence of propagation delay," *Performance Evaluation*, vol. 104, pp. 23–41, 2016.

[16] K. A. Negy, P. Rizun, and E. G. Sirer, "Selfish mining re-examined," 2020.

[17] R. Pass and E. Shi, "Fruitchains: A fair blockchain," in *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, 2017, pp. 315–324.

[18] C. Li, P. Li, D. Zhou, Z. Yang, M. Wu, G. Yang, W. Xu, F. Long, and A. C.-C. Yao, "A decentralized blockchain with high throughput and fast confirmation," in *Proceedings of USENIX Annual Technical Conference (USENIX ATC)*, 2020, pp. 515–528.

[19] J. Wang and H. Wang, "Monoxide: Scale out blockchains with asynchronous consensus zones," in *Proceedings of USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2019, pp. 95–112.

[20] (2020) Headers-first. [Online]. Available: https://btcinformation.org/en/developer-guide#headers-first

[21] (2020) Ghash.io. [Online]. Available: https://en.wikipedia.org/wiki/GHash.io

[22] G. O. Karame, E. Androulaki, and S. Capkun, "Double-spending fast payments in bitcoin," in *Proceedings of ACM conference on Computer and Communications Security (CCS)*, 2012, pp. 906–917.