Power Adjusting and Bribery Racing: Novel Mining Attacks in the Bitcoin System

Shang Gao
cssgao@comp.polyu.edu.hk
Department of Computing
The Hong Kong Polytechnic University
Hong Kong

Zecheng Li
cszcli@comp.polyu.edu.hk
Department of Computing
The Hong Kong Polytechnic University
Hong Kong

Zhe Peng
cszpeng@comp.polyu.edu.hk
Department of Computing
The Hong Kong Polytechnic University
Hong Kong

Bin Xiao
csbxiao@comp.polyu.edu.hk
Department of Computing
The Hong Kong Polytechnic University
Hong Kong

ABSTRACT

Mining attacks allow attackers to gain an unfair share of the mining reward by deviating from the honest mining strategy in the Bitcoin system. Among the most well-known are block withholding (BWH), fork after withholding (FAW), and selfish mining. In this paper, we propose two new strategies: power adjusting and bribery racing, and introduce two novel mining attacks, Power Adjusting Withholding (PAW) and Bribery Selfish Mining (BSM) adopting the new strategies. Both attacks can increase the reward of attackers. Furthermore, we show PAW can avoid the "miner's dilemma" in BWH attacks. BSM introduces a new "venal miner's dilemma", which results in all targets (bribes) willing to help the attacker but getting less reward finally. Quantitative analyses and simulations are conducted to verify the effectiveness of our attacks. We propose some countermeasures to mitigate the new attacks, but a practical and efficient solution remains to be an open problem.

CCS CONCEPTS

• **Security and privacy → Distributed systems security**; *Economics of security and privacy*.

KEYWORDS

Bitcoin, blockchain, mining attacks, selfish mining, block withholding, fork after withholding, bribery attack.

ACM Reference Format:
Shang Gao, Zecheng Li, Zhe Peng, and Bin Xiao. 2019. Power Adjusting and Bribery Racing: Novel Mining Attacks in the Bitcoin System. In *2019 ACM SIGSAC Conference on Computer and Communications Security (CCS '19), November 11–15, 2019, London, United Kingdom.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3319535.3354203

1 INTRODUCTION

Bitcoin is a decentralized cryptocurrency based on the blockchain technique [20]. In the Bitcoin system, participants (*miners*) can be rewarded by adding transaction records (recorded in a new *block*) to the "ledger" of past transactions (blockchain). When adding transaction records, a proof of work (PoW) [24] is required, which needs miners to solve cryptographic puzzles. The first miner who solves the puzzles and generates a valid block can be rewarded (with 12.5 bitcoins in 2019) by the system. The process of generating blocks is known as the "mining process". After that, a new round of mining begins and miners try to generate the next block. When two or more miners simultaneously generate and propagate their blocks, it will cause a *fork* on the blockchain. To ensure the consensus, the system will choose the firstly extended branch (i.e., the longest branch) as the main chain. Miners on other branches will move to the main chain when they are aware of a longer branch.

In the Bitcoin system, the difficulty of solving cryptographic puzzles is adjusted in every two weeks to ensure the average time of the mining process in each round is a constant (10 minutes). As the hash rate of today's mining power is more than 7×10^{19} Hash/s [26], the probability of a solo miner discovering a block is very small. Therefore, miners join together as mining pools to reduce the reward variance. When a pool discovers a block, the reward will be shared by pool miners based on their contributions (the number of submitted *shares*). Most of the pools have a manager for pool management, including work allocation and reward distribution.

Though the Bitcoin is designed with security in mind, previous research points out attackers can increase their reward of mining when deviating from honest mining strategies, such as adopting selfish mining [9], block withholding (BWH) [5, 37], and fork after withholding (FAW) [13]. In selfish mining, an attacker withholds the discovered block and continues mining on this block as a private chain. When other miners find a block, the attacker selectively propagates the withheld blocks on the private chain to cause a fork. In BWH attacks, an attacker splits her computational power into solo mining (innocent mining) and in-pool mining (infiltration mining). When attacker's infiltration mining finds a valid block (full proof of work, FPoW), she withholds it and continues submitting other shares (partial proof of work, PPoW) which makes her seem to contribute to the pool. Since the pool can never be rewarded from the attacker's infiltration mining, the victim pool will suffer from a loss. While other miners, including the attacker's innocent mining, will get more rewards because of the loss of the victim pool. Previous study shows BWH attacks can be more profitable than honest mining when the attacker splits her power properly [5], but will encounter a "miner's dilemma" when two pools use BWH attacks against each other (both pools earn less than the reward of honest mining) [7]. FAW attacks work similar to BWH attacks, but the attacker will propagate the withheld FPoW when another miner (not in the victim pool) finds a valid block to cause an intentional fork. Since the attacker may earn an extra reward from the fork, FAW attacks can be more profitable than BWH attacks. Besides, the "miner's dilemma" can be avoided in FAW attacks, since the larger pool can earn an extra reward. These mining attacks can also affect other PoW-based cryptocurrencies such as Litecoin [31] and Dogecoin [30], and may become more powerful when multiple coin systems are involved [14].

In this paper, we take an in-depth study of the mining attack strategies and discover several interesting revelations (Section 4). Based on our analysis, we propose novel attacking strategies to increase the reward of attackers (Section 5 and 6). We also discuss potential strategy space (Section 7.1). Finally, to mitigate these attacks, we propose practical countermeasures (Section 7.2 and 7.3). However, fully preventing such attacks remains to be an open problem.

We conclude our revelations as follows:

- *BWH and FAW are not optimal for a large parameter space.* We separate the mining process into two phases and analyze the reward of BWH and FAW. Our analysis shows that BWH and FAW are not optimal when considering power adjusting strategies (Section 4.1).
- *Power adjusting can improve the attacker's reward in FAW without falling into the "miner's dilemma".* We propose Power Adjusting Withholding (PAW) which combines power adjusting and FAW, and show PAW can increase the attacker's reward (up to 2.5 times of the extra reward in FAW) and avoid the "miner's dilemma" in BWH. (Section 5)
- *Attackers can adopt bribery racing for a higher reward in selfish mining.* We discuss the "0-lead" racing (two branches with the same length races each other) and find the attacker can bribe other miners (targets) to work on her branch. In most cases, the targets will choose to extend the attacker's branch for a higher reward (Section 4.2). We further propose Bribery Selfish Mining (BSM) which combines bribery racing and selfish mining, and shows BSM can result in 10% extra gains for an attacker in comparison with selfish mining (Section 6).
- *Targets can fall into a "venal miner's dilemma" in bribery racing.* We discuss multiple targets situations in bribery racing and find out targets may face a "venal miner's dilemma": all targets will decide to help the attacker (extending the attacker's branch), but they will suffer from a loss compared with their rewards when no one helps (Section 6.3).

2 PRELIMINARIES

2.1 Bitcoin Background

Mining process. In the Bitcoin system, the participants (miners) record the transfer information in the blocks of a blockchain. The header of each block contains the hash of previous block header, a Merkle root [34] of all recorded transactions, a timestamp, the target (representing the difficulty D of current Bitcoin system), a nonce, etc. Simply speaking, the mining process is to solve cryptographic puzzles: generating nonces (PoWs) which make the hash value of the header satisfy the difficulty constraint, i.e., $SHA256(SHA256(Block.Header)) < D$. The value of D is adjusted by the Bitcoin system to ensure the time of finding a new block is about 10 minutes. Once a miner discovers a valid nonce, it broadcasts the result to allow others to verify. When the new block is selected as the main (longest) chain, the Bitcoin system will reward the miner who finds a valid nonce with 12.5 bitcoins (BTC).

Pooled mining. Since the probability of a solo miner generating a valid block is very small, solo miners can join together as mining pools to find nonces, and share the reward based on their contributions. Though the expected rewards of pooled mining and solo mining are same (ignoring the small pool management fee), pooled mining can reduce the reward variance. Most of the mining pools are maintained by a manager to distribute work to different pool miners. Pool miners find and submit shares (FPoW and PPoW) to show their work done for the pool. Pool managers can use an FPoW to generate a valid block and get the reward for the pool. Though a PPoW is useless for generating a new block, finding a PPoW is much easier than an FPoW, which can be used to show the effort of pool miners (sharing rewards based on the number of submitted FPoWs and PPoWs).

Forks. When multiple miners simultaneously broadcast their discovered blocks, it can cause a fork in the blockchain since other miners will use the firstly received block as the new blockchain header [6]. In this scenario, miners continue mining based on the firstly received block and regard the heaviest branch (the longest branch has the most accumulated blocks) as the main chain. Notice that only the miner who finds a block in the main chain can be rewarded. Forks can also be intentionally generated, such as in selfish mining or FAW attacks.

2.2 Related Work

Selfish mining. An attacker can intentionally generate a fork to earn an extra reward through selfish mining [1, 9]. Specifically, when an attacker discovers a valid block, she withholds the block as a private chain and continues to find the next block on the private chain. When other miners propagate a valid block, the attacker selectively propagates the withheld blocks to cause a fork. The attacker can earn an extra reward when the private branch is selected as the main chain. However, the attacker can also suffer from a loss when her branch is not chosen. Several approaches have been proposed to optimize the reward of selfish mining [4, 22, 38].

BWH attacks. An attacker can sabotage the reward of a victim pool by BWH attacks [5, 37]. Specifically, an attacker splits her power into innocent mining (mines solely) and infiltration mining (mines in a pool). The infiltration mining only submits PPoWs to the manager and withholds (discards) all discovered FPoWs to make the pool suffer from a loss (and make innocent mining more profitable at the same time). When the mining power is split properly, the attacker can get a higher reward than honest mining [16]. In 2014, "Eligius" suffered from a BWH attack and lost 300 BTC [35]. The attacker was finally found (via statistics) since she only uses two accounts. However, an attacker can avoid being detected by using many accounts and frequently replaceing her pooled miners with new accounts. Pools can also use BWH attacks against each other (a BWH game). In such scenarios, attackers will encounter a "miner's dilemma": both of them will suffer from a loss under the Nash equilibrium (similar to the "prisoner's dilemma") [7].

FAW attacks. FAW attacks combine selfish mining and BWH attacks [13]. At first, FAW attacks work similar to BWH attacks (splitting power into solo mining and pooled mining). However, when an attacker finds an FPoW in the victim pool, she withholds the FPoW until other honest solo miners discover a valid block. Then, the attacker submits the withheld FPoW to the pool manager to cause a fork as with selfish mining [13]. By rewarding from intentional forks, FAW attacks can be more profitable than BWH attacks (the lower bound of FAW attacks is BWH attacks). Besides, the miner's dilemma may not hold in FAW games (pools use FAW attacks against each other), since the two-pool FAW game becomes a pool-size game (the larger pool can get an extra reward).

Bribery attacks. Bribery attacks can increase the probability of the attacker's branch selected as the main chain [2, 15]. Since bribery attacks only help the attacker to win in forks, they have to cooperate with other fork-related attacks, such as double-spending [12] (i.e., plain bribery attacks cannot bring any profit to an attacker). In bribery attacks, an attacker will charge other miners with bribes when extending her branch. For instance, the attacker pre-mines a block which contains a transaction T_B^A (A is the attacker's address and B is an address which anyone can claim [2]) and broadcasts a transaction $T_{A'}^A$ which transfers the same money to another address A' of the attacker. After $T_{A'}^A$ is recorded on the blockchain, the attacker propagates the pre-mined block to cause a fork. When other miners adopt the attacker's branch, they can claim the bribes in B. Bribing can also be made in a less visible way [17]. Bonneau *et al.* [2] show the possibility of a successful double-spending increases with bribery attacks. However, the the amount of bribes is non-trivial especially when the attacker's branch is far behind the main chain [2]. Besides, forking for other purposes (e.g. blocking transactions [18, 21] or getting higher rewards [9, 13]) has not been discussed with bribery attacks.

3 THREAT MODEL AND ASSUMPTION

3.1 Threat Model

An attacker can be a miner or a pool manager in a closed/open pool. Besides, an attacker can forge different identities to join multiple open pools with different accounts and IDs via Sybil attacks. The computational power of an attacker is finite. She can distribute her power into innocent mining (mining as an honest solo miner) and infiltration mining (mining in open pools as in BWH attacks), and adjust the power allocation dynamically. If the attacker is a manager of an open pool, her infiltration mining power should be "loyal mining power" [7]. Lastly, an attacker can plant Sybil nodes in the network to preferentially propagate the attacker's block to increase the probability of the attacker's branch selected as the main chain when forks occur.

3.2 Assumptions

We made the following assumptions to simplify our analysis, which are consistent with the assumptions of other Bitcoin mining attacks, such as selfish mining [1, 9], BWH attacks [7, 9, 16], FAW attacks [13], and bribery attacks [2, 15].

1. We normalize the total computational power of the Bitcoin system as 1 [7, 9, 13, 16]. The computational power of each miner/pool is a fraction of this total, which should be less than or equal to 0.5 to avoid "51% attacks" [3].

2. No unintentional forks in the Bitcoin system. This assumption is reasonable since the probability of unintentional forks is negligible, about 0.41% [10] (this assumption is also made in [13]). Therefore, the expected reward of a miner equals to the probability of finding a valid block in each round. Since the time of a miner finds a valid block has an exponential distribution with mean inversely to his computational power [8], the probability of a miner finding a valid block equals to his normalized computational power.

3. Miners are selfish but honest except the attackers. Honest miners can choose their best strategy (i.e. mining on which branch) to get a higher reward, but will not launch any attacks [7, 9, 13, 16]. The assumption of profit-driven miners is also made in [7, 9]. It is acceptable since mining on different branches brings no different rewards for followers (discussed in Section 4.2).

4. We normalize the reward of finding a valid block to 1 instead of 12.5 BTC. The reward in our analysis is the expected reward [13, 16].

5. When a pool manager earns a reward (via propagating a valid block from an FPoW), he distributes the reward to pool miners based on the number of submitted shares (FPoWs and PPoWs) in this round [7, 13, 16].

4 OBSERVATION AND MOTIVATION

4.1 FAW Reward

In FAW attacks, an attacker's innocent mining power will help the attacker to win a whole profit. The infiltration mining power will help to share the reward from the victim pool's profit. As a result, after an FPoW is discovered, the infiltration mining power only contributes to earning a share (via submitting PPoWs). Therefore, when the mining pool is relatively large and the pool's (attacker's) branch has less chance to be chosen as the main chain, the infiltration mining reward (shared profit) is "less attractive" to the attacker after an FPoW is found. Allocating more power to innocent mining, which is the "more attractive" part with higher reward, would be more profitable.

Note that power adjusting does not always mean shifting infiltration power to innocent power. Shifting reversely is also acceptable when winning a share is more attractive (in a smaller pool with better chance to win in forks). We discuss

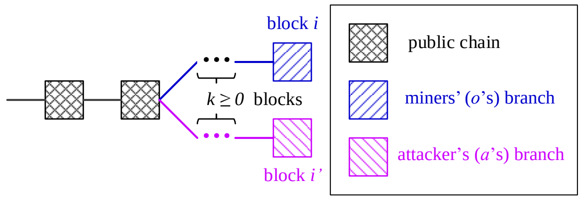


Figure 1: "0-lead" racing. The blockchain is forked by two branches of the same length (caused by the attacker a and some other miners o).

the detailed power shifting strategies under different scenarios in Section 5.

4.2 "0-lead" Racing

"0-lead" racing indicates two branches of the same length race on the blockchain, as depicted in Fig. 1. This situation can occur in selfish mining [9], stubborn mining [22], and FAW attacks [13], when the attacker withholds block i' till another miner finds block i. In this situation, the attacker (a) will continue mining on block i' and the miners which find blocks on block i's branch (o) will continue mining on block i. For other miners (b), they can choose to mine on block i or i', since it has no difference which branch is extended (normally they choose to mine on the firstly received block). When any branch is extended by b, b can get one profit, and no profit for other cases.

Now consider the attacker lures b to mine on block i' with bribery attacks (containing bribery transactions in block i'). The racing will become unfair (bribery racing) since choosing to mine on block i' branch becomes more profitable for b than on block i branch, The best strategy for miners in b is to extend attacker's branch. Moreover, since the attacker's reward increases with the probability of her branch selected as the main chain in some mining attacks (e.g. selfish mining), the attacker can get more rewards than not adopting bribery racing when choosing a proper bribing fee.

5 POWER ADJUSTING WITHHOLDING

5.1 Overview

We introduce novel power adjusting withholding (PAW) attacks against mining pools which combines power adjusting strategy and FAW attacks. In our observation in FAW attacks (Section 4.1), we point out FAW adopts fixed power splitting strategy, which may result in wasting too much mining power on the less attractive reward. The key insight behind our new PAW attacks is *allowing the attacker dynamically adjust the mining power between innocent and infiltration mining.* Therefore, the attacker can always increase her reward by allocating more power on the more attractive reward (whole profit or a share). Besides, PAW can also preserve the advantage of FAW to avoid the "miner's dilemma".

In one victim pool scenario, the attacker first splits her computational power into innocent mining and infiltration mining, and mines both solely (via innocent mining) and in-pool (via infiltration mining). There are four possible cases when a valid block (FPoW) is found:

Case 1. *Found by innocent mining.* The attacker propagates it and earns a legitimate profit.

Case 2. *Found by other miners not in the victim pool.* The attacker accepts the block and continues mining the next block. No profit will be earned by the attacker.

Case 3. *Found by other victim pool miners.* The attacker earns a shared profit from the pool.

Case 4. *Found by infiltration mining.* When the infiltration mining finds an FPoW, the attacker withholds the FPoW and reallocates her computational power. There are further three subcases when a new block/FPoW is found.

Case 4-1. *Found by innocent mining.* The attacker discards the withheld FPoW and propagates the new one.

Case 4-2. *Found by other miners not in the victim pool.* The attacker immediately submits the withheld FPoW to the victim pool manager. A fork can be generated in the Bitcoin network when the manager propagates her FPoW.

Case 4-3. *Found by other victim pool miners.* The attacker discards the withheld FPoW and earns a share.

In summary, the strategies of PAW attacks are similar to FAW attacks. The difference is that an attacker will reallocate her computational power after an FPoW is found by infiltration mining in PAW attacks, while in FAW attacks, an attacker never adjusts her computational power. Clearly, PAW attacks can earn as much reward as FAW attacks (when not adjusting computational power). Besides, we also show that PAW attacks can be more profitable than FAW attacks when the computational power is adjusted properly in the Section 5.2 and 5.3.

In multiple victim pools scenario, PAW attacks adjust the power allocation every time when an FPoW is found by innocent mining, and submit all k withheld FPoWs when other miners not in the victim pools find a valid block to cause a fork with $(k+1)$ branches ($1 \leqslant k \leqslant n$ when targeting at n pools). Since k branches are generated by infiltration mining, the probability of the attacker winning a shared profit via infiltration mining increases.

5.2 PAW Against One Pool

Theoretical analysis. Refereing to the cases in Section 5.1, we use the following parameters to analyze PAW attacks:

α:	Total computational power of the attacker;
β:	Computational power of the victim pool;
τ_1:	Attacker's original infiltration mining power as a proportion of α before Case 4;
τ_2:	Attacker's reallocated infiltration mining power as a proportion of α after Case 4;
$\bar{\tau}$:	Attacker's average portion of computational power allocated to infiltration mining in a mining process;
c:	Probability of the attacker's FPoW will be selected as the main chain in Case 4-2.

The attacker allocates $(1-\tau_1)\alpha$ computational power for innocent mining and $\tau_1\alpha$ for infiltration mining before Case 4; and will use $(1-\tau_2)\alpha$ and $\tau_2\alpha$ for innocent mining and infiltration mining respectively after Case 4. The computational power of the victim pool (β) does not include the infiltration mining power ($\tau_1\alpha$ or $\tau_2\alpha$). c is a coefficient related to attacker's network capability and the Bitcoin network topology [19]. The calculation of c can be referred to [13]. Attacker can also get an extra reward even when c is unknown (e.g. by setting $c = 0$ or $c = \alpha + \beta$ for an honest/rational pool manager, which is same as the strategies in FAW attacks [13]).

Attacker's reward. Based on our analysis in Section 4.1, the attacker has $(1-\tau_1)\alpha$ probability to fall in Case 1; $(1-\alpha-\beta)$ probability for Case 2; β probability for Case 3; and $\tau_1\alpha$ for Case 4. The total probability of these four cases is 1 as expected. In Case 4, after adjusting the computational power, the total mining power will become $(1-\tau_2)\alpha$ since the infiltration mining will not propagate any FPoWs. Therefore, the attacker has $\tau_1\alpha \cdot \frac{(1-\tau_2)\alpha}{1-\tau_2\alpha}$ probability to fall in Case 4-1; $\tau_1\alpha\cdot\frac{1-\alpha-\beta}{1-\tau_2\alpha}$ probability in Case 4-2; and $\tau_1\alpha\cdot\frac{\beta}{1-\tau_2\alpha}$ probability in Case 4-3. As expected, the sum of the probability in Case 4-1, 4-2, and 4-3 is $\tau_1\alpha$ (same as Case 4). We can further derive the reward of a PAW attacker $R_a^P(\tau_1,\tau_2)$ as follows:

$$R_a^P(\tau_1,\tau_2) = (1-\tau_1)\alpha + \beta\cdot\frac{\tau_1\alpha}{\beta+\tau_1\alpha}+ \\ \tau_1\alpha\cdot\left(\frac{(1-\tau_2)\alpha}{1-\tau_2\alpha}+(c\cdot\frac{1-\alpha-\beta}{1-\tau_2\alpha}+\frac{\beta}{1-\tau_2\alpha})\cdot\frac{\bar\tau\alpha}{\beta+\bar\tau\alpha}\right). \quad (1)$$

$\bar\tau$ can be regarded as a function related to τ_1 and τ_2. We discuss the calculation of $\bar\tau$ in Theorem 5.1.

Equation (1) is derived by separating the attacker's reward into innocent mining reward and infiltration mining reward. We have two cases for innocent mining reward: Case 1 and Case 4-1. The innocent mining reward is $(1-\tau_1)\alpha + \tau_1\alpha\frac{(1-\tau_2)\alpha}{1-\tau_2\alpha}$. For infiltration mining reward, we have three cases: Case 3, Case 4-2, and Case 4-3. Since the attacker will share the profit with other miners in pool, the reward will be $\beta\cdot\frac{\tau_1\alpha}{\beta+\tau_1\alpha}$, $c\tau_1\alpha\cdot\frac{1-\alpha-\beta}{1-\tau_2\alpha}\cdot\frac{\bar\tau\alpha}{\beta+\bar\tau\alpha}$, and $\tau_1\alpha\cdot\frac{\beta}{1-\tau_2\alpha}\cdot\frac{\bar\tau\alpha}{\beta+\bar\tau\alpha}$ in the three cases respectively. Therefore, we can derive the attacker's reward in Equation (1).

Clearly, PAW downgrades to FAW when $\tau_1=\tau_2$.

THEOREM 5.1. *The average portion of computational power for infiltration mining in i-th pool, Pool$_i$, is ($\bar\tau^{(i)}_{1,\cdots,k}, k\geqslant 2$):*

$$\bar\tau^{(i)}_{1,\cdots,k} = \frac{(1-\sum_{i\in\mathcal{P}}\tau_k^{(i)}\alpha)\sum_{j=1}^{k-1}\tau_j^{(i)}+\tau_k^{(i)}}{(1-\sum_{i\in\mathcal{P}}\tau_k^{(i)}\alpha)(k-1)+1},$$

where $\tau_j^{(i)}$ is the infiltration mining power in Pool$_i$ as a portion of α between $(j-1)$-th and j-th FPoWs are found. $\mathcal{P}$ is the victim pool set.

Specifically, in one victim scenario, when $k=2$, $\bar\tau=\bar\tau^{(1)}_{1,2}$, $\tau_1=\tau_1^{(1)}$, and $\tau_2=\tau_2^{(1)}$, we have:

$$\bar\tau=\bar\tau^{(1)}_{1,2}=\frac{\tau_1+\tau_2-\tau_1\tau_2\alpha}{2-\tau_2\alpha}. \quad (2)$$

Proof of Theorem 5.1 is presented in Appendix-A.

How to adjust mining power? Based on Equation (1), we formalize the best power adjusting strategy by finding the optimal portion of attacker's computational power ($\widehat\tau_1$ and $\widehat\tau_2$) to maximize the expected reward ($R_a^P(\tau_1,\tau_2)$):

$$\begin{aligned} &\arg\max_{\tau_1,\tau_2} R_a^P(\tau_1,\tau_2), \\ \text{s.t.}\quad & 0\leqslant\tau_1\leqslant 1,\quad 0\leqslant\tau_2\leqslant 1. \end{aligned} \quad (3)$$

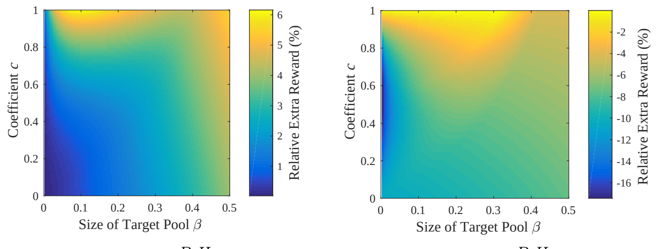
Equation (3) can be solved by using Lagrange multipliers (presented in Appendix-B). Besides, the maximized reward with $\widehat\tau_1$ and $\widehat\tau_2$ satisfies the following theorem.

THEOREM 5.2. *A PAW attacker can always earn more rewards than honest mining. The reward of a PAW attack has a lower bound defined by the reward of an FAW attack.*

PROOF. Previous research [13] has shown that the reward of an FAW attack is always greater than that of honest mining (α) when a proper τ_1 is chosen. Suppose the optimal τ_1 for an FAW attack is $\dot\tau_1$. The reward of the FAW attack under $\dot\tau_1$ equals to the reward of a PAW attack when $\tau_1=\tau_2=\dot\tau_1$. Now considering a τ_1-preset PAW attack with fixed $\tau_1=\dot\tau_1$, and adjusting $\tau_2=\ddot\tau_2$ to maximize the reward in Case-4. The total reward of this τ_1-preset PAW attack should be greater than or equal to the FAW attack. Furthermore, considering a regular PAW attack that adjusts τ_1 and τ_2 simultaneously to maximize the reward at $\widehat\tau_1$ and $\widehat\tau_2$, the reward should at least equal to the τ_1-preset PAW attack. Therefore, we have:

$$\alpha < R_a^F(\dot\tau_1) = R_a^P(\dot\tau_1,\dot\tau_1)\leqslant R_a^P(\dot\tau_1,\ddot\tau_2)\leqslant R_a^P(\widehat\tau_1,\widehat\tau_2),$$

where $R_a^F(\dot\tau_1)$ represents the maximized reward of an FAW attacker at $\dot\tau_1$. □



(a) $RER_a^{P,H}$. (b) $RER_p^{P,H}$.

Figure 2: A PAW attack against one pool. RERs of an attacker and the victim pool, according to the size of victim pool β and coefficient c when the attacker has 0.2 computational power $\alpha=0.2$.

Quantitative analysis and simulation. We use a specific case to show the additional reward of an attacker who launches PAW attacks against one pool. Referring to [13], we also use the expected relative extra reward (RER) to show the performance of PAW attacks. The expected RER can be expressed as

$$RER_x^{S_1,S_2}=\frac{R_x^{S_1}-R_x^{S_2}}{R_x^{S_2}}, \quad (4)$$

where S_1 and S_2 indicate different strategies (can be honest mining H, PAW P, or FAW F), and x represents an entity (can be the attacker a or victim pool p), and $R_x^{S_1}$ represents the reward of x when adopting S_1 strategy. Clearly, we have $R_a^H=\alpha$ and $R_p^H=\beta+\tau_1\alpha$.

Considering a specific case: the attacker has 0.2 computational power ($\alpha=0.2$), and the victim pool has no greater than 0.5 computational power ($0<\beta\leqslant 0.5$). $RER_a^{P,H}$ and $RER_p^{P,H}$ are depicted in Fig. 2.

The expected RER of an attacker is depicted in Fig. 2-a. We can infer that PAW attacks can always be more profitable than honest mining ($RER_a^{P,H}>0$), regardless of β and c. Furthermore, similar to FAW attacks, the reward of PAW attacks increases with the network coefficient c. While when $c=0$, the PAW attacks can still earn more rewards than FAW/BWH attacks (FAW attacks will downgrade to BWH attacks).

Fig. 2-b shows the victim pool will always suffer from a loss because of the PAW attacks. Different from the results presented in [13], the reward of the victim pool does not always increase with c. It is because $RER_p^{P,H}$ *increases with c if τ_1 and τ_2 are fixed.* For instance, suppose the reward of the victim pool is $R_p^P(\dot\tau_1,\dot\tau_2,c_1)$ at $\dot\tau_1$ and $\dot\tau_2$. When c_1 increases to c_2, $R_p^P(\dot\tau_1,\dot\tau_2,c_2)$ can be greater than $R_p^P(\dot\tau_1,\dot\tau_2,c_1)$, but

Table 1: Attacker's optimal infiltration mining power. The values $(\widehat{\tau}_1, \widehat{\tau}_2), \widehat{\tau}$ indicate the optimal infiltration mining power $(\widehat{\tau}_1, \widehat{\tau}_2)$ of PAW and $\widehat{\tau}$ of FAW respectively.

β	c=0	c=0.25	c=0.5	c=0.75	c=1
0.1	(0.06, 0), 0.06	(0.07, 0.29), 0.08	(0.11, 0.78), 0.10	(0.18, 1), 0.16	(0.33, 1), 0.39
0.2	(0.14, 0), 0.12	(0.15, 0), 0.15	(0.18, 0.51), 0.19	(0.25, 1), 0.26	(0.38, 1), 0.44
0.3	(0.22, 0), 0.18	(0.25, 0), 0.23	(0.28, 0.01), 0.26	(0.35, 0.56), 0.33	(0.41, 1), 0.46

Table 2: $RER_a^{P,H}$ (%) against one pool. The values $x(y)$ indicate $RER_a^{P,H}$ in simulation and theoretical analysis respectively.

β	c=0	c=0.25	c=0.5	c=0.75	c=1
0.1	0.59(0.59)	0.75(0.75)	1.44(1.44)	3.11(3.12)	6.17(6.16)
0.2	1.28(1.27)	1.41(1.41)	1.84(1.84)	3.13(3.13)	5.50(5.50)
0.3	2.04(2.04)	2.25(2.25)	2.51(2.51)	3.13(3.13)	4.71(4.72)

the attacker will adjust the optimal τ_1 and τ_2 to $\ddot{\tau}_1$ and $\ddot{\tau}_2$ to maximize her reward. $R_p^P(\ddot{\tau}_1, \ddot{\tau}_2, c_2) > R_p^P(\dot{\tau}_1, \dot{\tau}_2, c_1)$ does not always hold, especially when β is small. We also think the victim pool's reward under FAW attacks does not always increase with c. The analysis is presented in Appendix-C. However, if the victim pool's manager is rational, the manager can try to increase c (if possible) as the best response of the attacker. Besides, when β is large enough, R_p^P becomes an increasing function with c. The pool manager should always increase c even when he is aware of PAW attacks, which also increases the attacker's reward.

Moreover, we compare the optimal infiltration mining power of FAW ($\widehat{\tau}$) and PAW ($\widehat{\tau}_1$ and $\widehat{\tau}_2$) in Table 1. When c is relatively small, the attacker will allocate less power in infiltration mining after finding an FPoW (e.g. $\widehat{\tau}_2 = 0$ when $c = 0$). It is because the attacker has less chance to earn a reward from causing a fork. When c is a large value, the attacker will allocate more power for infiltration mining after finding an FPoW (e.g. $\widehat{\tau}_2 = 1$ when $c = 1$). In such scenarios, the attacker has a very high chance to get a reward from a fork.

Furthermore, we present a more intuitive comparison between PAW attacks and FAW attacks. Considering the attacker has 0.2 computational power, we show the expected RERs of the attacker using PAW and FAW attacks in three cases: $\beta = 0.1$, $\beta = 0.2$, and $\beta = 0.3$. The results are depicted in Fig. 3. Clearly, the PAW attacks are at least as profitable as FAW attacks. When the optimal $\widehat{\tau}_1$ and $\widehat{\tau}_2$ are same, the reward of PAW attacks equals to that of FAW attacks. In other cases, the reward of PAW attacks is always higher. When $c = 1$ in case 1 ($\beta = 0.1$), the RER of PAW attacks is more than twice as FAW attacks. Besides, the smaller victim pool gives the attacker more rewards with large c. It is because the attacker can earn a higher share in smaller pools when c is large. Referring to Table 1, the attacker will reallocate all her mining power to infiltration mining $\widehat{\tau}_2 = 1$ when c is large (e.g. $c = 1$). After Case 4 occurs, the reward of innocent mining (in Case 4-1) is 0 since no power is allocated to innocent mining. The reward of other miners (in Case 4-2) is also 0 since $c = 1$. Therefore, the reward goes to Case 4-2, and the attacker will get a higher share in small pools.

Finally, we implement a Monte Carlo simulator in Matlab to verify the accuracy of our theoretical analysis of PAW attacks. We run the simulator over 10^9 rounds to show an attacker's RER in three cases (the attacker with 0.2 computational power and the victim pool with 0.1, 0.2, and 0.3).

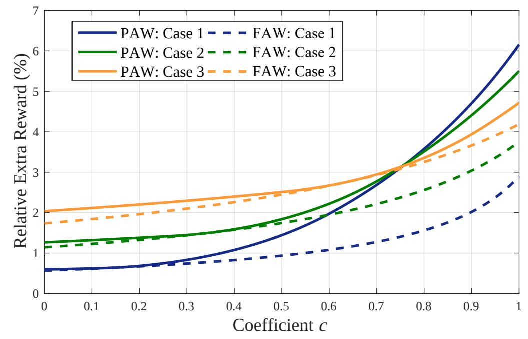


Figure 3: $RER_a^{P,H}$ and $RER_a^{F,H}$ against one pool, according to the coefficient c when the $\alpha = 0.2$ in three cases. Case 1, 2, and 3 represent the victim pool with computational power 0.1, 0.2, and 0.3 respectively.

The result is presented in Table 2. The attacker's RER is almost the same as we expected, and the attacker can always earn an extra reward with PAW attacks.

5.3 PAW Against Multiple Pools

Theoretical analysis. We introduce additional parameters to analyze PAW attacks against n pools (Pool$_{p_1}$, Pool$_{p_2}$, ..., and Pool$_{p_n}$) as follows:

$\beta^{(p_i)}$:	Computational power of Pool$_{p_i}$;
$\tau_j^{(p_i)}$:	Attacker's infiltration mining power in Pool$_{p_i}$ as a proportion of α between $(j-1)$-th and j-th FPoWs are found;
$c_j^{(p_i)}$:	Probability of the attacker's FPoW in Pool$_{p_i}$ will be selected as the main chain among $(j+1)$ branches.

Attacker's reward can be derived by summing up the reward of innocent mining, the share from other victim pool miners, and the reward of generating branches. Prior to deriving the total reward of innocent mining, we first consider the infiltration mining in Pool$_{p_1}$, Pool$_{p_2}$, ..., and Pool$_{p_i}$ finds an FPoW before the innocent mining in order (other miners do not find any FPoWs). The innocent mining reward will be:

$$\begin{cases} R_{inno_0}, & i = 0; \\ R_{inno_i} \prod_{j=1}^{i} \frac{\tau_j^{(p_j)} \alpha}{1 - \sum_{k=1}^{j} \tau_{j+1}^{(p_k)} \alpha}, & i > 0. \end{cases} \tag{5}$$

where R_{inno_i} is the reward of innocent mining when infiltration mining in Pool$_{p_1}$, ..., and Pool$_{p_i}$ only submits PPoWs, $R_{inno_i} = (1 - \sum_{k=1}^{n} \tau_{i+1}^{(p_k)})\alpha$. To simplify the expression, we let $\prod_{j=1}^{i}$ part in Equation (5) be 1 when $i = 0$. Furthermore, we can derive the total reward of innocent mining:

$$\sum_{i=0}^{n} \sum_{\boldsymbol{p}_i \in \mathcal{P}^i} \left((1 - \sum_{k=1}^{n} \tau_{i+1}^{(p_k)})\alpha \prod_{j=1}^{i} \frac{\tau_j^{(p_j)} \alpha}{1 - \sum_{k=1}^{j} \tau_{j+1}^{(p_k)} \alpha} \right), \tag{6}$$

where $\boldsymbol{p}_i$ means the infiltration mining in Pool$_{p_1}$, Pool$_{p_2}$, ..., and Pool$_{p_i}$ finds an FPoW before the innocent mining in order. We have $\boldsymbol{p}_i = (p_1, p_2, \cdots, p_i)$ (when $m \neq n$, $p_m \neq p_n$), in which p_m and p_n are not exchangeable (e.g. $(1, 2, 3) \neq (2, 1, 3)$).

Then we consider the share when a miner in the victim pools finds an FPoW. Referring to Equation (5), we only need to replace R_{inno_i} with the shared reward when miners in the victim pools find an FPoW. The total shared reward

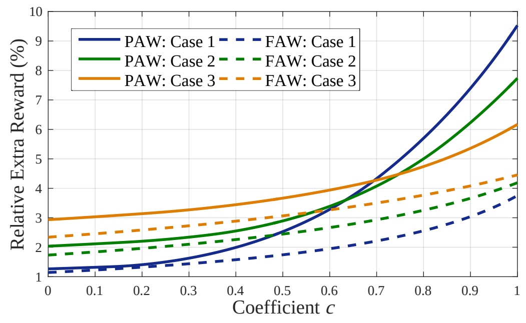


Figure 4: $RER_a^{P,H}$ and $RER_a^{F,H}$ against two pools, according to the coefficient c ($c_1^{(p_1)} = c_1^{(p_2)} = c$ and $c_2^{(p_1)} + c_2^{(p_2)} = c$) when the $\alpha = 0.2$ in three cases. Case 1, 2, and 3 represent two pools with computational power (β_1, β_2) equals to $(0.1, 0.1)$, $(0.2, 0.1)$, and $(0.3, 0.1)$ respectively.

Table 3: $RER_a^{P,H}$ against two pools. The values $x(y)$ indicate $RER_a^{P,H}$ in simulation and theoretical analysis respectively.

(β_1, β_2)	c=0	c=0.25	c=0.5	c=0.75	c=1
$(0.1, 0.1)$	1.27(1.26)	1.52(1.50)	2.54(2.53)	4.98(4.98)	9.53(9.53)
$(0.2, 0.1)$	2.03(2.04)	2.27(2.26)	2.89(2.89)	4.50(4.50)	7.73(7.74)
$(0.3, 0.1)$	2.93(2.93)	3.21(3.20)	3.66(3.67)	4.49(4.49)	6.14(6.17)

can be expressed as:

$$\sum_{i=0}^{n} \sum_{\boldsymbol{p}_i \in \mathcal{P}^i} \left(\sum_{k=1}^{n} \frac{\beta^{(p_k)} \cdot \bar{\tau}_{1,\cdots,k}^{(p_k)} \alpha}{\beta^{(p_k)} + \bar{\tau}_{1,\cdots,k}^{(p_k)} \alpha} \cdot \prod_{j=1}^{i} \frac{\tau_j^{(p_j)} \alpha}{1 - \sum_{k=1}^{j} \tau_{j+1}^{(p_k)} \alpha} \right), \tag{7}$$

where $\bar{\tau}_1^{(p_k)} = \tau_1^{(p_k)}$.

Finally, we can replace R_{inno_i} in Equation (5) to express the reward of causing branches ($k \geqslant 2$ since no reward from causing a fork when innocent mining first finds a valid block). The total reward of causing branches is

$$\sum_{i=0}^{n} \sum_{\boldsymbol{p}_i \in \mathcal{P}^i} \left(\sum_{k=2}^{n} \frac{(1-\alpha-\beta)\bar{\tau}_{1,\cdots,k}^{(p_k)} \alpha c_i^{(p_k)}}{\beta^{(p_k)} + \bar{\tau}_{1,\cdots,k}^{(p_k)} \alpha} \prod_{j=1}^{i} \frac{\tau_j^{(p_j)} \alpha}{1 - \sum_{k=1}^{j} \tau_{j+1}^{(p_k)} \alpha} \right), \tag{8}$$

where β is the total computational power of n victim pools, $\beta = \sum_{i=1}^{n} \beta^{(i)}$. We also set $c_0^{(p_k)} = 0$ to indicate no reward for the attacker when other benign miners (not in victim pools) find a valid block first.

Therefore, we derive the reward of an attacker when launching PAW attacks against n pools by summing up Equation (6), (7), and (8):

$$\sum_{i=0}^{n} \sum_{\boldsymbol{p}_i \in \mathcal{P}^i} \Bigg(\Big((1 - \sum_{k=1}^{n} \tau_{i+1}^{(p_k)}) \alpha + \sum_{k=1}^{n} \frac{\beta^{(p_k)} \cdot \bar{\tau}_{1,\cdots,k}^{(p_k)} \alpha}{\beta^{(p_k)} + \bar{\tau}_{1,\cdots,k}^{(p_k)} \alpha}$$
$$+ \sum_{k=2}^{n} \frac{(1 - \alpha - \beta)\bar{\tau}_{1,\cdots,k}^{(p_k)} \alpha c_i^{(p_k)}}{\beta^{(p_k)} + \bar{\tau}_{1,\cdots,k}^{(p_k)} \alpha} \Big) \prod_{j=1}^{i} \frac{\tau_j^{(p_j)} \alpha}{1 - \sum_{k=1}^{j} \tau_{j+1}^{(p_k)} \alpha} \Bigg).$$

Quantitative analysis and simulation. We use a specific case to show the additional reward of the attacker who launches PAW attacks against two pools. We make the two assumptions to reduce the parameter dimensions: (1) the attacker's computational power is 0.2; and (2) $c_k^{(p_i)} = c/k$ when k infiltration mining finds k different FPoWs ($0 \leqslant c \leqslant 1$). Furthermore, we compare the attacker's RER (%) of PAW attacks and FAW attacks according to c in three scenarios: the two victim pools with computational power (β_1, β_2) equals to $(0.1, 0.1)$, $(0.2, 0.1)$, and $(0.3, 0.1)$ respectively.

The attacker's RERs of the two attacks are depicted in Fig. 4. Clearly, the PAW can always earn a higher reward than FAW. and the RER of PAW is up to 2.5 times as FAW (when $c = 1$ in $(0.1, 0.1)$ scenario).

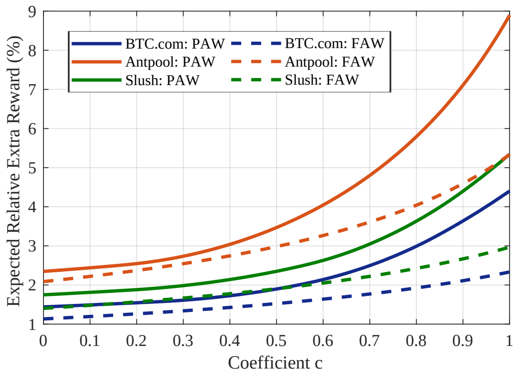


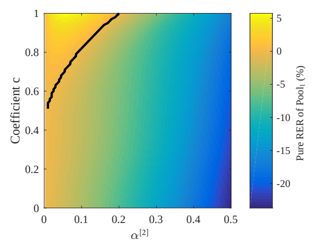
Figure 5: The extra reward of BTC.com (29.6%), Antpool (12.9%), or Slush (9.9%) when one pool attacks the other two with PAW or FAW.

We consider the top three biggest Bitcoin mining pool in a real-world scenario: BTC.com [27] with 29.6% mining power, Antpool [23] with 12.9%, and Slush [33] with 9.9% (mining power is collected by Jan. 2019). Suppose one of them launch PAW or FAW attacks, which the other two remain in honest mining. The extra reward in each attacking scenario is depicted in Fig. 5. Antpool benefits the most with our PAW. The maximum extra reward is nearly 9%, where its extra reward with FAW is only 5.3%. For other pools, PAW also always better than FAW regardless of c.

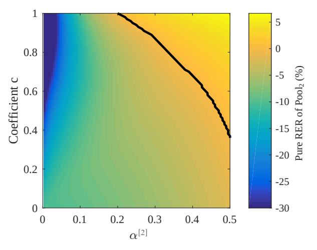
We further use a Monte Carlo simulator in Matlab to verify our analysis. We show the attacker's average RER in three cases over 10^9 rounds (same as the three cases above, which $\alpha = 0.2$ and the (β_1, β_2) equals to $(0.1, 0.1)$, $(0.2, 0.1)$, and $(0.3, 0.1)$ respectively). The result is presented in Table 3. The attacker's RER is almost identical in simulation and theoretical analysis.

5.4 "Miner's Dilemma" Analysis

Mining pools can also launch PAW attacks against each other simultaneously. We prove that PAW can avoid the "miner's dilemma" (similar to FAW), and the outcome becomes a size game (i.e. the larger pool can win). In a two-pool PAW game (Pool$_1$ vs Pool$_2$), each pool will propagate the result when its innocent mining firstly finds an FPoW. If Pool$_1$'s infiltration mining discovers an FPoW first (and vice versa for Pool$_2$), it withholds the FPoW. After that, if Pool$_1$'s innocent mining discovers a new FPoW, it discards the withheld FPoW and propagates the new one. If Pool$_2$'s innocent mining finds a new FPoW, it discards the withheld one. Otherwise, when other miners broadcast a valid block, it propagates the withheld FPoW to cause a fork. Notice that since Pool$_2$ also uses PAW attacks against Pool$_1$ at the same time, the fork can have three branches when both of the infiltration mining in Pool$_1$ and Pool$_2$ discover an FPoW before other miners. In this scenario, the manager of one pool will choose the FPoW generated from the opponent's infiltration mining. For instance, when Pool$_1$'s infiltration mining finds FPoW$_1$ (in Pool$_2$), and Pool$_2$'s infiltration mining finds FPoW$_2$ (in Pool$_1$), Pool$_1$'s manager will choose FPoW$_2$ when other miners find a valid block, and vice versa for Pool$_2$. The blockchain



(a) Pool$_1$'s pure RER and winning condition.



(b) Pool$_2$'s pure RER and winning condition.

Figure 6: Results of a PAW game according to Pool$_2$'s size $\alpha^{[2]}$ and coefficient c ($c_1^{[1]} = c_1^{[2]} = c$ and $c_2^{[1]} = c_2^{[2]} = c/2$) when $\alpha^{[1]} = 0.2$.

will have three branches caused by FPoW$_1$, FPoW$_2$, and the valid block.

We define the winning condition as pool miners (not including the opponent's infiltration power) earning an extra reward.

Nash equilibrium point. We use the following parameters to analyze a PAW game between Pool$_1$ and Pool$_2$ ($i \in \{1, 2\}$).

$\alpha^{[i]}$:Total computational power of Pool$_i$;

$f_1^{[i]}$:Pool$_i$'s original infiltration mining power;

$f_2^{[i]}$:Pool$_i$'s reallocated infiltration mining power after its infiltration mining finds an FPoW;

$c_1^{[i]}$: Probability of the Pool$_i$'s withheld FPoW is selected as the main chain in two-branch cases;

$c_2^{[i]}$: Probability of the Pool$_i$'s withheld FPoW is selected as the main chain in three-branch cases.

We present the detailed calculation of Pool$_i$'s reward $R^{[i]}$ in Appendix-D. Now we analyze the Nash equilibrium in a two-pool PAW game. Under the Nash equilibrium point, one pool may satisfy the winning condition (i.e. earning more rewards than honest mining), which breaks the "miner's dilemma" in BWH attacks.

THEOREM 5.3. *The two-pool PAW game has a unique Nash equilibrium* $(\boldsymbol{f}^{[1]}, \boldsymbol{f}^{[2]})$ *which either satisfies* $\nabla_{\boldsymbol{f}^{[1]}} R^{[1]} = 0$, *and* $\nabla_{\boldsymbol{f}^{[2]}} R^{[2]} = 0$; *or a point on a borderline which maximizes* $R^{[1]}$ *with* $\boldsymbol{f}^{[1]}$ *and* $R^{[2]}$ *with* $\boldsymbol{f}^{[2]}$ $(\boldsymbol{f}^{[1]} = (f_1^{[1]}, f_2^{[1]})$ *and* $\boldsymbol{f}^{[2]} = (f_1^{[2]}, f_2^{[2]}))$.

Detailed proof is presented in Appendix-E.

Winning conditions. We quantitatively analyze the reward in a two-pool PAW game under the Nash equilibrium point. For simplicity, we assume $c_1^{[1]}$ and $c_1^{[2]}$ are symmetric, as well as $c_2^{[1]}$ and $c_2^{[2]}$ (i.e. $c_1^{[1]} = c_1^{[2]} = c$ and $c_2^{[1]} = c_2^{[2]} = c/2$, where $0 \leqslant c \leqslant 1$ [13]). Before presenting the results, we define the pure reward of Pool$_i$, which means the total reward of Pool$_i$ miners not including the infiltration mining of the opponent pool (Pool$_{\neg i}$). Therefore, the pure reward of Pool$_i$ under honest mining is $\alpha^{[i]}$, which can be used to calculate its pure RER.

Fig. 6 shows the results of a two-pool PAW game in terms of $\alpha^{[2]}$ and c when $\alpha^{[1]} = 0.2$. The pure RERs of Pool$_1$ and Pool$_2$ are presented in Fig. 6-a and Fig. 6-b respectively, where the black lines represent the same pure RER as honest

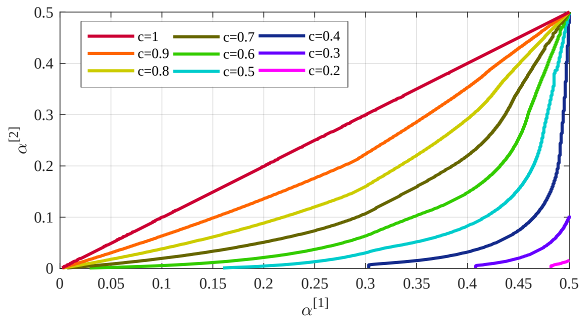


Figure 7: Winning conditions of Pool$_1$. The right side of each line represents the winning range of Pool$_1$ under the corresponding c.

mining (i.e. Pool$_1$'s pure RER $= 0$ in Fig. 6-a and Pool$_2$'s pure RER $= 0$ in Fig. 6-b). Each pool can earn an extra reward above the black lines while taking a loss below the lines. When $\alpha^{[2]} > 0.2$, Pool$_2$ may get more rewards under the two-pool PAW game. However, when $\alpha^{[2]} < 0.2$, Pool$_2$ will always take a loss (same as the result in a two-pool FAW game). Therefore, the PAW game becomes a pool size game, and the "miner's dilemma" [7] (each pool will always take a loss when its computational power is less than 0.5) may not hold.

We further analyze the winning conditions. Pool$_1$'s winning conditions are depicted in Fig. 7 (Pool$_2$'s winning conditions can be found by swapping $\alpha^{[1]}$ and $\alpha^{[2]}$). The nine lines represent the same reward as honest mining when c varies from 0.2 to 1. The right side of each line represents the winning range of Pool$_1$ under the corresponding c. When $c = 1$, the borderlines are exactly same for Pool$_1$ and Pool$_2$ ($\alpha^{[1]} = \alpha^{[2]}$). Therefore, as we expected, the larger pool can earn an extra reward, while the smaller pool will always take a loss. Furthermore, even when the computational power of each pool is less than 0.5 and c is less than 1, there can still be a winner, which can avoid the "miner's dilemma" [7]. In summary, the winning condition is related to $\alpha^{[1]}$, $\alpha^{[2]}$, and c. The larger pool can earn an extra reward (related to c), while the smaller pool will always take a loss (regardless of c).

6 BRIBERY SELFISH MINING

6.1 Overview

We introduce novel bribery selfish mining (BSM) attacks which combines bribery racing strategy and selfish mining. In our observation in bribery racing (Section 4.1), we point out when "0-lead" racing occurs, an attacker can bribe other miners to work on her branch to increase the probability of winning in forks. Therefore, BSM can increase the attacker's reward via *including bribery transactions in the attacker's branch*. Furthermore, when bribery racing occurs, targets (bribes) can encounter a "venal miner's dilemma", which all targets will choose to accept the attacker's branch for a higher reward, but still suffer from a loss (comparing with no one accepting).

We define three entities in BSM, attacker (a), the target (venal miner, b), and other miners (o). In BSM, an attacker a will include bribes in each block of her private chain (T_B^A). After the attacker discovering a block, she withholds the block and broadcasts a transaction to spend the bribes on the public chain ($T^{A}_{A'}$, where A' can be another address of the attacker). When another miner (o or b) finds a block on the public chain, the attacker selectively publishes the withheld blocks in the private branch to cause a fork. A target pool (the venal pool) can choose either to extend the attacker's branch or the public branch. When choosing to extend the attacker's branch, the target can claim the bribes in B. When choosing the other one, the target cannot claim the bribes, since the bribes have been transferred to A' on the public chain.

We consider "honest" opponents in our analysis. When opponents are rational, it will become a bribery racing game (Appendix-F). Besides, bribing can also be done via smart contracts [17], out-of-band payment, or negative-fee mining pool [2]. Here we only consider in-band payment since it is the most practical method.

6.2 Modeling BSM

We first define the additional parameters in BSM:

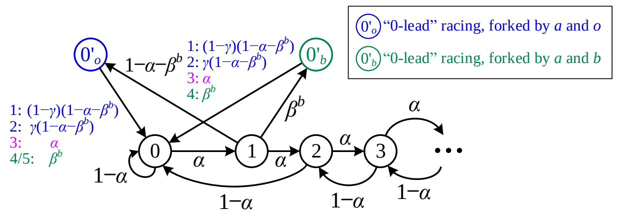
β^b: Computational power of the target pool (venal pool) to be analyzed;

γ: The ratio of other miners that choose to mine on the attacker's branch;

ε: The fraction of reward in each round as bribes of the attacker willing to pay per block when the target pool chooses to accept bribes.

Notice that since the bribes are available to any miners adopting the attacker's branch, there can be multiple targets in our model. An attacker's reward will increase with more targets (discussed latter). Here β^b only represents one target since we analyze the optimal strategy of one target. Other targets are determined by γ (a larger γ with more targets). The relationship between γ and c (in Section 5.2) is $c = \alpha + \gamma(1 - \alpha - \beta^b)$.

State machine and probability. For simplicity, we use the selfish mining strategy in [9] to analyze the BSM. The same analysis can also be applied to other optimal selfish mining strategies [22, 38]. We show the state machine of BSM in Fig. 8. The states represent the "lead of the attacker" (i.e., the difference between the length of attacker's private branch and the public branch). State 0 means that there is no branch, and state $0'_x$ means "0-lead" racing: there are two branches of length one (attacker's branch and x's branch). Specifically, $0'_b$ means the two branches are from the target pool and attacker, and $0'_o$ means the two branches are from other miners and the attacker. From state $0'_x$, there are five possible transitions, all leading to state 0: **(1)** other miners mine a block on x's branch (probability $(1 - \gamma)(1 - \alpha - \beta)$); **(2)** other miners mine a block on attacker's branch (probability $\gamma(1 - \alpha - \beta)$); **(3)** attacker mines a block on attacker's branch (probability α); **(4)** the target mines a block on x's branch (probability β); and **(5)** the target mines a block on attacker's branch (probability β). At state $0'_o$, the target can choose either (4) or (5). Choosing (4) means the target denying the bribes, and choosing (5) means accepting. While at state $0'_b$, the target *must choose (4)* to avoid the loss from the previously mined block. Regardless of the target's strategies, the probability of transition from state $0'_x$ to 0 is 1 as expected.



$0'_o$ "0-lead" racing, forked by a and o
$0'_b$ "0-lead" racing, forked by a and b
$0'_o$
$0'_b$
$1{-}\alpha{-}\beta^b$
1: $(1{-}\gamma)(1{-}\alpha{-}\beta^b)$
2: $\gamma(1{-}\alpha{-}\beta^b)$
3: α
4: β^b
1: $(1{-}\gamma)(1{-}\alpha{-}\beta^b)$
2: $\gamma(1{-}\alpha{-}\beta^b)$
3: α
4/5: β^b
β^b
0
1
2
3
α
α
α
α
$1{-}\alpha$
$1{-}\alpha$
$1{-}\alpha$
$1{-}\alpha$

Figure 8: State machine of BSM. The value on each arrow indicates the probability of a state transition. From state $0'_o$ and $0'_b$ to state 0 should have four arrows. We only draw one arrow here to simplify the expression.

Based on the state machine in Fig. 8, we have the following equations:

$$\begin{cases} \alpha p_0 = (1-\alpha-\beta)p_1 + \beta p_1 + (1-\alpha)p_2 \\ p_{0'_o} = (1-\alpha-\beta)p_1 \\ p_{0'_b} = \beta p_1 \\ \alpha p_k = (1-\alpha)p_{k+1}, \text{when } k \geqslant 2 \\ \sum_{k=0}^{+\infty} p_k + p_{0'_o} + p_{0'_b} = 1. \end{cases}$$

We could further derive the probability of each state:

$$p_0 = \frac{1-2\alpha}{2\alpha^3 - 4\alpha^2 + 1};$$

$$p_{0'_o} = \frac{(1-\alpha-\beta)(\alpha - 2\alpha^2)}{2\alpha^3 - 4\alpha^2 + 1};$$

$$p_{0'_b} = \frac{\beta(\alpha - 2\alpha^2)}{2\alpha^3 - 4\alpha^2 + 1};$$

$$p_k = \frac{\alpha - 2\alpha^2}{2\alpha^3 - 4\alpha^2 + 1}\left(\frac{\alpha}{1-\alpha}\right)^{k-1}, \text{when } k \geqslant 1.$$

Reward. We analyze the rewards in all possible events (miners finding a block). We focus on the differences comparing with [9] (i.e., "0-lead" racing in event b, c, and d in [9]), and briefly describe other events. Details of other events could refer to [9]. The rewards here represent the system rewards. Bribes will be discussed later.

(a) Any state but two branches of length 1, the attacker finds a block. The attacker withholds the block to increase the lead. The reward will be determined later.

(b) Was two branches of length 1, the attacker finds a block. We split it into two events as depicted in Fig. 9-a and 9-b: (b-1) the two branches were from the attacker and other miners; and (b-2) the two branches were from the attacker and target. For both cases, when the attacker publishes her branch, she can get 2 rewards.

(c) Was two branches of length 1, others/target finds a block on attacker's branch. We split it into three events as depicted in Fig. 9-c, 9-d, and 9-e: (c-1) forked by the attacker and other miners, and other miners find a block on attacker's branch; (c-2) forked by the attacker and other miners, and the target finds a block on attacker's branch; and (c-3) forked by the attacker and target, and other miners find a block on attacker's branch. Notice that the target will not extend the attacker's branch when the fork is caused by himself. The attacker will always get 1 reward in all cases. Other miners will get 1 reward in event (c-1) and (c-3), and the target will get 1 reward in event (c-2).

(d) Was two branches of length 1, others/target finds a block on other's/target's branch. We split it into four events as depicted in Fig. 9-f, 9-g, 9-h, and 9-i. The attacker can cause a fork with the target or others. The finder of the next block could be either the target or others. For all cases,

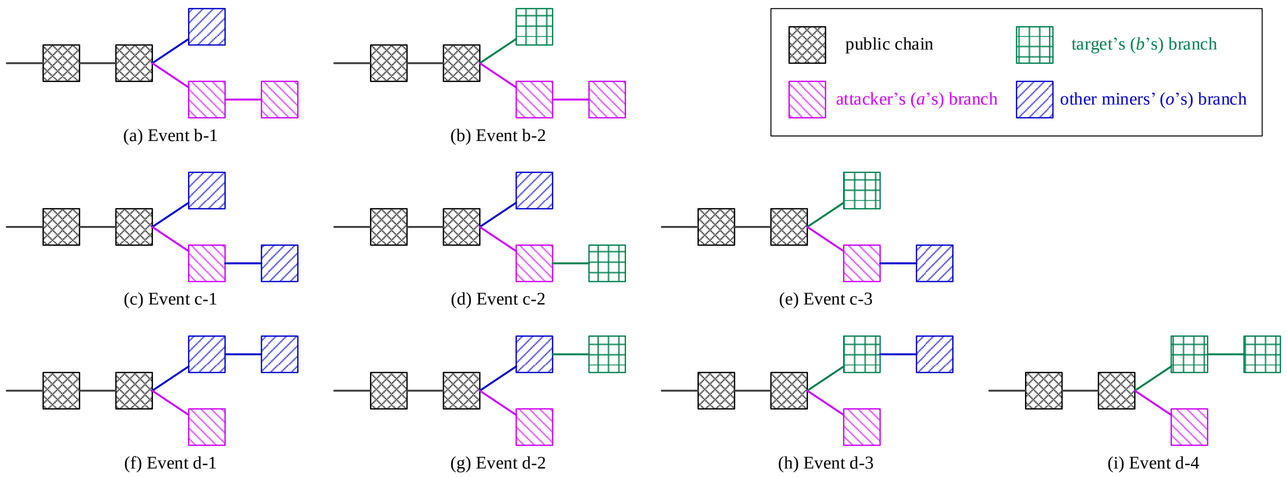


Figure 9: Possible events after "0-lead" racing in BSM. Event b: a's branch is extended by a. Event c: a's branch is extended by o or b. Event d: o's or b's branch is extended by o or b.

the attacker cannot get any reward since her branch is not extended and will not be selected as the main chain. Other miners will get 2 rewards in event (d-1) and 1 in (d-2) and (d-3). The target will get 2 in event (d-4) and 1 in (d-2) and (d-3).

(e) No private branch, others/target finds a block. The finder will publish the block and get 1 reward.

(f) Lead was 1, others/target finds a block. The attacker publishes her private branch to cause a fork. The reward will be determined later.

(g) Lead was 2, others/target finds a block. The attacker publishes her private branch to get 2 rewards.

(h) Lead was more than 2, others/target finds a block. The attacker publishes one block to get 1 reward since the attacker's branch will be selected as the main chain eventually.

We first derive the attacker's reward when the target chooses to accept the bribes. She can be rewarded in event (b), (c), (g), and (h). The attacker's system reward R_a is

$$\begin{aligned} R_a = &(p_{0'_o}+p_{0'_b})\cdot\alpha\cdot 2 + p_{0'_o}\cdot(\gamma(1-\alpha-\beta^b)+\beta^b) \\ &+ p_{0'_b}\cdot\gamma(1-\alpha-\beta^b) + p_2\cdot(1-\alpha)\cdot 2 + \sum_{i=3}^{+\infty} p_i\cdot(1-\alpha), \end{aligned} \tag{9}$$

which is an increasing function with γ (i.e., bribing more targets will bring the attacker more reward).

When considering the bribes (a fraction ε of the total system reward), The attacker's reward R_a^B becomes:

$$R_a^B = (1-\varepsilon)R_a, \tag{10}$$

which is a decreasing function with ε.

For the reward of the target pool, since he will help the attacker to extend the branch, event (d-2) will not happen. The target can get a system reward in (c-2), (d-3), (d-4), and (e). Considering the bribes, the total reward R_b^B is

$$R_b^B = p_{0'_o}\cdot\beta^b + p_{0'_b}\cdot\gamma(1-\alpha-\beta^b) + p_{0'_b}\cdot\beta^b\cdot 2 + p_0\cdot\beta^b + \varepsilon\cdot R_a. \tag{11}$$

Similarly, we could derive the reward of other miners R_o^B from event (c-1), (c-3), (d-1), (d-3), and (e):

$$\begin{aligned} R_o^B = &(p_{0'_o}+p_{0'_b})\cdot\gamma(1-\alpha-\beta^b) + p_{0'_o}\cdot(1-\gamma)(1-\alpha-\beta^b)\cdot 2 \\ &+ p_{0'_b}\cdot(1-\gamma)(1-\alpha-\beta^b) + p_0\cdot(1-\alpha-\beta^b). \end{aligned} \tag{12}$$

Now we analyze the reward of the target when not choosing to accept bribes. Event (c-2) will not happen since the target will not help to extend the attacker's branch. The target pool can be rewarded in (d-2), (d-3), (d-4), and (e). The reward $R_b^{B'}$ is

$$R_b^{B'} = p_{0'_o}\cdot\beta^b + p_{0'_b}\cdot\gamma(1-\alpha-\beta^b) + p_{0'_b}\cdot\beta^b\cdot 2 + p_0\cdot\beta^b. \tag{13}$$

The reward of other miners $R_o^{B'}$ from event (c-1), (c-3), (d-1), (d-2), (d-3), and (e) is:

$$\begin{aligned} R_o^{B'} = &(p_{0'_o}+p_{0'_b})\cdot\gamma(1-\alpha-\beta^b) + p_{0'_o}\cdot(1-\gamma)(1-\alpha-\beta^b)\cdot 2 \\ &+ p_{0'_o}\cdot\beta^b + p_{0'_b}\cdot(1-\gamma)(1-\alpha-\beta^b) + p_0\cdot(1-\alpha-\beta^b). \end{aligned} \tag{14}$$

The attacker can be rewarded in event (b), (c-1), (c-3), (g), and (h). The reward is

$$\begin{aligned} R_a^{B'} = &(p_{0'_o}+p_{0'_b})\cdot\alpha\cdot 2 + p_{0'_o}\cdot\gamma(1-\alpha-\beta^b) \\ &+ p_{0'_b}\cdot\gamma(1-\alpha-\beta^b) + p_2\cdot(1-\alpha)\cdot 2 + \sum_{i=3}^{+\infty} p_i\cdot(1-\alpha). \end{aligned} \tag{15}$$

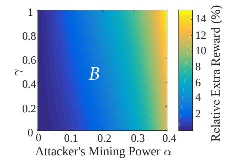
THEOREM 6.1. *When launching BSM, the target can always get a higher reward when accepting the bribes at $0'_o$ state. The attacker can get a higher reward than that in selfish mining when she pays proper bribes.*

PROOF. Comparing Equation (11) with Equation (13), we can obtain $R_b^B \geqslant R_b^{B'}$ since $0 \leqslant \varepsilon \leqslant 1$ and $R_a > 0$. When the attacker adopts $\varepsilon > 0$, the target ensures $R_b^B > R_b^{B'}$. Therefore, extending the attacker's private branch is always the optimal strategy for targets at $0'_o$ state.

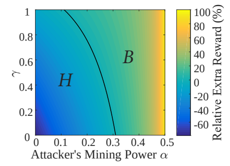
The rewards in selfish mining are the same as the rewards in BSM when the target chooses to deny the bribes. Therefore, to get a higher reward, the attacker must ensure $R_a^B > R_a^{B'}$. Referring to Equation (10) and Equation (15), we can derive

$$R_a^B > R_a^{B'} \quad \Rightarrow \quad 0 < \varepsilon < \frac{p_{0'_o}\cdot\beta^b}{p_{0'_o}\cdot\beta^b + R_a^{B'}}. \tag{16}$$

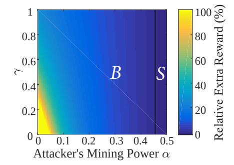
The upper bound of the attacker's reward is R_a in Equation (9) when $\varepsilon = 0$. □



(a) RER for a target when accepting bribes ($RER_b^{B,B'}$).



(b) BSM vs honest mining ($RER_a^{B,H}$). Solid line indicates $RER_a^{B,H} = 0$.



(c) BSM vs selfish mining ($RER_a^{B,S}$). Solid line indicates $RER_a^{B,S} = 0$.

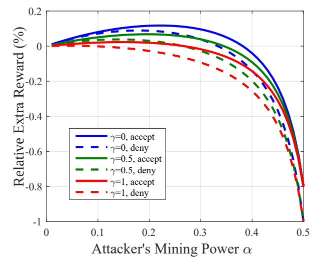
Figure 10: Dominant strategies for a target (a) and an attacker (b) and (c).

Quantitative analysis and simulation. Previous approaches have pointed out the total block generation rate decreases due to selfish mining (i.e., $R_a^B + R_b^B + R_o^B < 1$) [9, 22, 38]. Therefore, we first normalize the reward of entity x ($\frac{R_x^B}{R_a^B+R_b^B+R_o^B}$ and $\frac{R_x^{B'}}{R_a^{B'}+R_b^{B'}+R_o^{B'}}$) and then use the RER in Equation (4) to evaluate BSM (strategies can be selfish mining S, accepting bribes in BSM B, denying bribes in BSM B', and honest mining H).

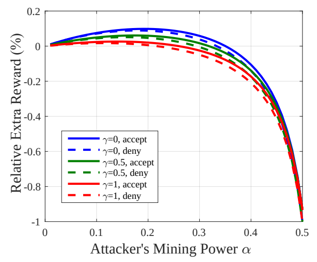
Dominant strategies. First, we consider the target's reward under different strategies (accepting or denying bribes) in BSM. We consider $\beta^b = 0.1$ and set $\varepsilon = 0.02$, which is $0.02 \times 12.5 = 0.25$ BTC per block (about 2500 USD/block in Aug. 2019 [29]). The rewarding difference $RER_b^{B,B'}$ (i.e. extra reward when accepting bribes comparing with denying) in terms of α and γ is depicted in Fig. 10-a. As expected, the dominant strategy for a target is always to accept the bribes, regardless of α and γ. Besides, $RER_b^{B,B'}$ increases with α, since the attacker will have more reward (and more bribes according) with a larger α.

Furthermore, we consider the attacker's reward of different strategies (H, S, or B). We set $\varepsilon = 0.02$ and $\beta^b = 0.2$. The attacker's dominant strategies ($RER_a^{B,H}$ and $RER_a^{B,S}$) are depicted in Fig. 10-b and 10-c. In Fig. 10-b, the right side of the line indicates honest mining is the dominant strategy, while the left side means BSM is the dominant one. Comparing with the results in [9], the winning area of BSM is larger than selfish mining. By properly setting ε, BSM gives a less threshold of launching attacks (e.g. when $\gamma = 0$, BSM only requires $\alpha > 0.31$, while selfish mining requires $\alpha > 0.33$). Furthermore, comparing with selfish mining, BSM is dominant in most cases, as depicted in Fig. 10-c. Selfish mining only outperforms BSM when α is large enough ($\alpha > 0.46$ in this case). Based on Theorem 6.1, the winning area of BSM can be extended with a smaller ε.

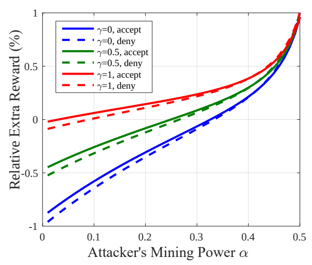
More specifically, we consider two cases: target's mining power is 0.1 and 0.3 respectively. We set bribes to $\varepsilon = 0.02$. Comparing with honest mining, the extra relative rewards of the target are depicted in Fig. 11-a and 11-b. When $RER_b^{B,H}$ and $RER_b^{B',H}$ are positive, BSM can bring the target more reward than honest mining. As expected, the rewards of the target when accepting the bribes (solid lines) are always greater than those when denying (dash lines). The optimal strategy for the target is to extend the attacker's branch and accept the bribes. The target can get a higher reward with a smaller α, since the attacker will suffer from a loss under such scenarios, as with [9, 22, 38]. The reward decreases when γ increases. It is because γ represents the ratio of other miners who extend the attacker's branch (i.e., other targets). These miners can damage the target's rewards when they decide to mine on attacker's branch (event c-3). The loss will increase with more other targets (larger γ).



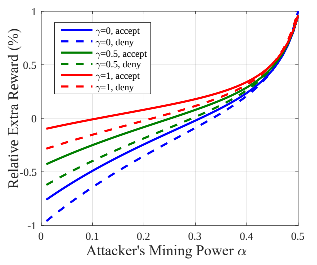
(a) Target's RER when $\beta^b = 0.1$.



(b) Target's RER when $\beta^b = 0.3$.



(c) Attacker's RER when $\beta^b = 0.1$.



(d) Attacker's RER when $\beta^b = 0.3$.

Figure 11: RER of a target and an attacker in BSM. $RER_b^{B,H}$ and $RER_b^{B',H}$ in (a) and (b). $RER_a^{B,H}$ and $RER_a^{B',H}$ in (c) and (d).

Table 4: The target's extra relative reward (%) in BSM ($RER_b^{B,H}$ and $RER_b^{B',H}$). The values $x(y)$ indicate the target's RERs in simulation and theoretical analysis respectively.

	$\gamma = 0$	$\gamma = 0.25$	$\gamma = 0.5$	$\gamma = 0.75$	$\gamma = 1$
Accept	5.85(5.84)	3.76(3.74)	1.64(1.65)	-0.44(-0.44)	-2.55(-2.53)
Deny	3.82(3.84)	1.64(1.65)	-0.54(-0.55)	-2.75(-2.74)	-4.94(-4.94)

For an attacker, her RERs are depicted in Fig. 11-c and 11-d. When satisfying Equation (16) (a relatively small α), the attacker can get a higher reward than selfish mining. When considering positive RER cases, the BSM can bring 10% additional extra reward than selfish mining, which is about $0.22 \times 10\% \times 12.5 = 0.275$ BTC per round (i.e. additional 2750 USD in every 10 minutes). The extra reward (comparing with selfish mining) is more significant with $\beta^b = 0.3$.

Simulation. We implement a Monte Carlo simulator in Matlab to verify our analysis. Suppose the target's mining power is 0.3, the attacker's mining power is 0.3, and the bribes are 0.02. We run the simulator over 10^8 rounds to show the relative rewards of the target and attacker ($RER_b^{B,H}$, $RER_b^{B',H}$, $RER_a^{B,H}$, and $RER_a^{B',H}$) in Table 4 and Table 5 respectively under different strategies (accepting or denying the bribes).

As we expected, the rewards of the attacker and target are almost the same as those in theoretical analysis. The target's optimal strategy is to accept the bribes to get a higher reward. The attacker can increase her reward with an acceptable cost.

Table 5: The attacker's extra relative reward (%) with BSM ($RER_a^{B,H}$ and $RER_a^{B',H}$). The values $x(y)$ indicate the attacker's RERs in simulation and theoretical analysis respectively.

	$\gamma=0$	$\gamma=0.25$	$\gamma=0.5$	$\gamma=0.75$	$\gamma=1$
Accept	-2.18(-2.18)	2.85(2.84)	7.85(7.85)	12.86(12.87)	17.88(17.89)
Deny	-8.94(-8.96)	-3.85(-3.84)	1.27(1.28)	6.42(6.40)	11.49(11.52)

Table 6: The target's extra relative reward ($RER_{b_i}^{B,H}$ and $RER_{b_i}^{B',H}$). (x,y) indicate the extra relative reward of Target$_1$ and Target$_2$ respectively.

Target$_2$ \ Target$_1$	Accept at $0'_o$	Deny at $0'_o$
Accept at $0'_o$	(**-2.58%**, **-0.62%**)	(-6.44%, **1.63%**)
Deny at $0'_o$	(**3.85%**, -1.85%)	(0.45%, 0.45%)

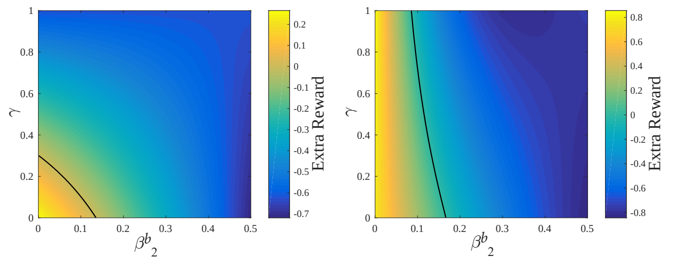
6.3 The Venal Miner's Dilemma

In selfish mining and BSM, an attacker can get extra rewards by causing forks. Previous work has pointed out these extra rewards are from the loss of other miners (b and o) [9, 13]. For a target, the loss occurs when other miners extend the attacker's branch instead of the target's (event c-3). However, the target cannot avoid the loss regardless of different strategies he takes. In other words, these events are controlled by other miners instead of the target. When other miners extend the attacker's branch, the target has to accept the loss (event c-3). Meanwhile, when the target takes the bribes, he actually makes other miners suffer from a loss (event c-2).

We have proved that the optimal strategy for a target is to accept the attacker's branch even with a very small positive value of ε. Therefore, the attacker can have a very high chance to win in forks via bribing multiple targets with little cost and can get extra rewards with BSM. In such scenarios, accepting the bribes becomes "avoiding a higher loss" for the targets. The targets can fall into a "venal miner's dilemma": all targets will suffer from a loss due to the attacks (similar to the "prisoner's dilemma"). Even though all targets can get higher rewards when no one helps the attacker, no target will deny the bribes since accepting the bribes is always the local-optimal strategy at $0'_o$ state. Therefore, we have a single Nash equilibrium for targets under BSM: all targets will choose to accept the bribes and extend the attacker's branch at $0'_o$.

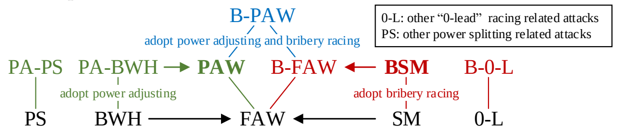
Notice that the "venal miner's dilemma" is different from the "looming tragedy" [2] and "miner's dilemma" [7]. The "looming tragedy" suggests that bribery attacks will damage Bitcoin's reputation. Miners should reject bribes and seek for long-term incentive to avoid *harming* Bitcoin exchange rate. While in BSM, even with a *fixed* Bitcoin exchange rate, a target can suffer from a loss due to the "venal miner's dilemma". The "miner's dilemma" involves multiple *attackers*. Attackers will choose to attack each other and fall into the dilemma to lose their rewards. While for the "venal miner's dilemma", it involves one attacker and multiple *targets*. Targets will choose to accept the attacker's branch and fall into the dilemma to lose their rewards.

Considering two targets Target$_1$ and Target$_2$ with mining power β_1^b and β_2^b in a bribery game under BSM. We define the winning condition for $Target_i$ is "getting a higher reward than honest mining" (i.e., $RER_{b_i}^{B,H} > 0$). Suppose $\alpha = 0.35$, $\beta_1^b = 0.2$, and $\varepsilon = 0.02$. We show the extra rewards and



(a) $RER_{b_1}^{B,H}$ and winning condition. (b) $RER_{b_2}^{B,H}$ and winning condition.

Figure 12: Extra rewards and winning conditions in a bribery game. Solid lines represent no extra reward ($R_{b_i}^B = \beta_i^b$). The left side of each line represents the winning condition of each target pool. The intersected part of the right sides of the two lines represents the "venal miner's dilemma".



B-PAW
adopt power adjusting and bribery racing
0-L: other "0-lead" racing related attacks
PS: other power splitting related attacks
PA-PS
PA-BWH
PAW
B-FAW
BSM
B-0-L
adopt power adjusting
adopt bribery racing
PS
BWH
FAW
SM
0-L

Figure 13: Attacking strategy space when adopting power adjusting and bribery racing. Other power splitting related attacks can be combination of different attacks. Other "0-lead" racing related attacks can be stubborn mining [22].

winning condition of each target in terms of β_2^b and γ (γ represents the ratio other miners mining on attacker's branch, excluding the two targets) in Fig. 12.

The left side of each line is the winning condition of each target. When β_2^b and γ are relatively small, both targets can win since the attacker will suffer from a loss. The extra rewards of Target$_2$ will not be greatly affected by γ when β_2^b is small. It is because the bribes contribute to a great part of Target$_2$'s reward in such scenarios. Even with a large γ, Target$_2$ can still be more profitable than honest mining (rewarding from the bribes) when β_2^b is relatively small. The union part of the winning conditions (i.e., the winning condition of Target$_2$) means there can be at least one winner, where the winner can avoid the "venal miner's dilemma". While for other parts (the left side of Target$_2$'s line), both of targets fall into the dilemma (no winner). For an attacker, when proper values of β_1^b, β_2^b, and ε are chosen, the attacker can ensure a higher reward and make targets fall into the dilemma regardless of γ.

We present a more intuitive example to show the venal miner's dilemma in BSM (we set $\gamma = 0$ to avoid other targets). Suppose the mining power of the attacker, Target$_1$, and Target$_2$ is 0.33, 0.1, and 0.3 respectively. The theoretical extra relative rewards of the targets are shown in Table 6. The optimal strategy for both targets is to accept the bribes at $0'_o$ (bold numbers). However, both of them will suffer from a loss due to the BSM, compared with the rewards when denying.

7 DISCUSSION

7.1 Strategy Space and Bribery PAW

We have discussed power adjusting with FAW and bribery racing with selfish mining. We also believe that power adjusting strategy can be applied to other power splitting related attacks such as BWH or launching different kinds of attacks simultaneously, and bribery racing to other "0-lead" racing related attacks such as stubborn mining [22]. A potential attacking strategy space is presented in Fig. 13. Furthermore, since the "0-lead" racing also occurs in FAW and PAW, is it also possible to combine bribery racing with FAW/PAW. The combination can be easily achieved via bribing through out-of-band payment or negative-fee mining pools. However, for in-band bribing, the requirement is non-trivial since transactions to be recorded are chosen by pool managers in pooled mining. We discuss possible strategies to lure or force a pool manager to choose the bribing transactions in Appendix-G.

In Bribery PAW (B-PAW), an attacker will only include bribes when calculating shares in her infiltration mining power. When an FPoW is found, she withholds it and broadcast a transaction to spend the bribes. When others miners find a block, she immediately submits the withheld block to cause "0-lead" racing. It is worth mentioning that the next round of infiltration mining does not include bribes and will immediately submit the newly discovered FPoW (different from FAW/PAW) since the primary task here is to win in "0-lead" racing. Some analyses on B-PAW are conducted in Appendix-G.

7.2 PAW Countermeasure

Detecting power adjusting. The difference between FAW and PAW attacks is the "power adjusting" mechanism in PAW attacks. Therefore, a pool manager can detect PAW attacks by statistically counting the number of submitted shares for each pool miner. However, we think this approach can be inefficient and error-prone because of two main reasons: (1) power adjusting does not always happen in PAW attacks (i.e. for a single victim scenario, power adjusting occurs only after Case 4); and (2) non-frequent power adjusting is legal and acceptable for honest miners. To more precisely detect power adjusting, we suggest the manager set a less difficulty constraint to find a PPoW, and count the number of shares over time. However, to precisely identify power adjusting, it will increase the workload of pool managers with a smaller constraint. Besides detecting power adjusting, pool managers can resort to methods for FAW/BWH detection. We think detecting PAW via stale FPoWs [13] is much easier than via power adjusting.

Detecting stale FPoWs. After receiving a new block, pool manager can regard FPoWs forked with the new block as stale FPoWs and expel the submitter to avoid FAW/PAW attacks. A beacon value approach has been proposed based on this idea [13]. However, adding a beacon field in a block's header breaks down the compatibility with existing mining hardware and increases the workload of pool managers.

We introduce a new detecting approach based on the timestamp field. Timestamp field is encapsulated in a block header when calculating hash values, which can avoid an attacker modifying this filed after an FPoW is found. Besides, it will be updated in every few seconds, which ensures the freshness of the share. Therefore, a network manager can first synchronize the time of each pool miner and identify stale FPoWs when the timestamp field deviates too much from the current time. Notice that when an attacker uses a forged timestamp field to calculate hash values, she can still be detected since it is hard to anticipate the time of other miners propagating blocks.

Suppose the manager sets a threshold k ($k > 0$), which regards shares submitted with timestamp field in $[X-k, X+k]$ legal at current time X when $X \geqslant k$ (otherwise, the legal period is $[0, X+k]$). Let the attacker sets the timestamp field to $X+t$ when calculating shares. She can avoid being detected when other miners find a block at $[X+t-k, X+t+k]$. The probability of infiltration mining discovers a legal FPoW (only considering the infiltration mining and other miners) is:

$$P = \int_0^{+\infty} \lambda_1 e^{-\lambda_1 x} \int_{x+t-k}^{x+t+k} \lambda_2 e^{-\lambda_2 y} dx dy = \frac{\lambda_1}{\lambda_1+\lambda_2}(e^{-\lambda_2(t-k)} - e^{-\lambda_2(t+k)}),$$

where $\lambda_1 = \tau_1\alpha$ and $\lambda_2 = 1-\alpha-\beta$, when $t \geqslant k$, and $\lambda_1 = \lambda_2 = 0$ when $t < k$. Since P is a decreasing function with t, the maximized probability P_{max} is $\frac{\lambda_1}{\lambda_1+\lambda_2}(1-e^{-2k(1-\alpha-\beta)})$ at $t = k$. Suppose $\alpha = 0.1$, $\beta = 0.3$, $c = 0.5$, and the manager sets k to 0.01 (6 seconds, $\frac{6 \text{ sec}}{10 \text{ min}} = 0.01$). The attacker's probability of submitting a valid FPoW is $P_{max} = 0.047\%$. The priori probability (probability after infiltration mining finding an FPoW) is $\frac{P_{max}}{\lambda_1/(\lambda_1+\lambda_2)} = 1-e^{-2k(1-\alpha-\beta)} = 1.19\%$, and is upper-bounded at 1.98% under our mechanism ($1-\alpha-\beta < 1$).

Approaches based on "detecting stale FPoW" can identify the infiltration miner who submits the stale FPoW. However, when attackers use Sybil nodes to launch PAW attacks distributively (distributing infiltration mining power to many infiltration mining accounts), the attacker can still win an additional reward even the submitter is expelled [13]. Besides, even when attackers do not propagate withheld FPoWs to avoid being detected, PAW can still get more rewards than BWH or FAW attacks.

7.3 Bribery Racing Countermeasure

We propose three countermeasures against bribery racing. First, the Bitcoin system may consider restricting the use of "anyone can claim" transactions. However, this approach will also sacrifice the flexibility and programmability of the Bitcoin. Moreover, the oriented bribery (Appendix-H) attacks or "whale transactions" [15] cannot be prevented by this approach. A further step is to include the receiver's confirmation (e.g. receiver's signature) when creating a transaction. Though this approach can avoid oriented bribery attacks targeting at random pools, it also incurs much communication overhead. For attackers, they can still make out-of-band negotiations with targets to get the confirmation. Besides, both approaches need modifications of the current Bitcoin system, which may be impractical. We hope these approaches can provide some insights into mitigating bribery attacks when designing new cryptocurrencies.

Second, when a fork occurs, miners should preferentially choose the branch containing the transactions which they previously received, and ignore other branches conflicted with these transactions. For instance, when a miner receives $T_{A'}^A$, and a fork with two branches (containing $T_{A'}^A$ and T_B^A respectively) occurs, the miner should extend the branch with $T_{A'}^A$. The bribery attacks can be prevented when all miners adopt this mechanism. However, it is unrealistic to assume that all miners adopt this approach since miners can be selfish (they may choose T_B^A for a higher reward). Nevertheless, this approach can still reduce γ, especially in oriented bribery attacks (a smaller γ indicates a less reward for attackers).

Finally, for pool managers, they should expel pool miners who submit FPoWs containing bribes (avoiding bribery racing in FAW/PAW). Though plain bribery attacks will not damage the pool's reward, and increasing c may be the better strategy for a higher pool's reward even when FAW attacks are detected [13] since bribery attack can increase the chance of winning in forks, pool managers should reject bribery transactions to avoid bribery FAW/PAW attacks for further loss.

8 CONCLUSION

We show that in PoW-based blockchain cryptocurrencies such as Bitcoin, mining attacks can be further extended by combining power adjusting and bribery racing. In PAW, an attacker can increase her reward (up to 2.5 times of FAW) and avoid the "miner's dilemma". In BSM, an attacker can gain 10% extra reward than selfish mining and make other miners (targets) suffer from a "venal miner's dilemma" (all targets will earn less under the Nash equilibrium). Quantitative analysis and simulations have been done to verify our analysis. To mitigate these attacks, we discuss possible countermeasures. But a practical solution to fully prevent these attacks remains to be open.

ACKNOWLEDGMENT

We gratefully acknowledge Tao Lu from Erasmus University for helpful technical discussions about conditional expectation. This research is partially supported by HK RGC GRF PolyU 152124/19E and HK ITF ITS/081/18.

REFERENCES

[1] Lear Bahack. 2013. Theoretical Bitcoin Attacks with Less than Half of the Computational Power (draft). In *arXiv preprint arXiv:1312.7013*.

[2] Joseph Bonneau. 2016. Why Buy When You Can Rent?. In *Proc. of the International Conference on Financial Cryptography and Data Security (FC)*. Springer.

[3] Danny Bradbury. 2013. The Problem with Bitcoin. In *Computer Fraud & Security*. Elsevier.

[4] Miles Carlsten, Harry Kalodner, S Matthew Weinberg, and Arvind Narayanan. 2016. On the Instability of Bitcoin without the Block Reward. In *Proc. of the ACM Conference on Computer & Communications Security (CCS)*. ACM.

[5] Nicolas T Courtois and Lear Bahack. 2014. On Subversive Miner Strategies and Block Withholding Attack in Bitcoin Digital Currency. In *arXiv preprint arXiv:1402.1718*.

[6] Christian Decker and Roger Wattenhofer. 2013. Information Propagation in the Bitcoin Network. In *Proc. of the IEEE International Conference on Peer-to-Peer Computing (P2P)*. IEEE.

[7] Ittay Eyal. 2015. The Miner's Dilemma. In *Proc. of the IEEE Symposium onSecurity and Privacy (Oakland)*. IEEE.

[8] Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert Van Renesse. 2016. Bitcoin-NG: A Scalable Blockchain Protocol. In *Proc. of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. USENIX.

[9] Ittay Eyal and Emin Gün Sirer. 2014. Majority is not Enough: Bitcoin Mining is Vulnerable. In *Proc. of the International Conference on Financial Cryptography and Data Security (FC)*. Springer.

[10] Arthur Gervais, Ghassan O Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. 2016. On the Security and Performance of Proof of Work Blockchains. In *Proc. of the ACM Conference on Computer & Communications Security (CCS)*. ACM.

[11] Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. 2015. Eclipse Attacks on Bitcoin's Peer-to-Peer Network. In *Proc. of the USENIX Security Symposium (Security)*. USENIX.

[12] Ghassan O Karame, Elli Androulaki, and Srdjan Capkun. 2012. Double-Spending Fast Payments in Bitcoin. In *Proc. of the ACM Conference on Computer & Communications Security (CCS)*. ACM.

[13] Yujin Kwon, Dohyun Kim, Yunmok Son, Eugene Vasserman, and Yongdae Kim. 2017. Be Selfish and Avoid Dilemmas: Fork After Withholding (FAW) Attacks on Bitcoin. In *Proc. of the ACM Conference on Computer & Communications Security (CCS)*. ACM.

[14] Yujin Kwon, Hyoungshick Kim, Jinwoo Shin, and Yongdae Kim. 2019. Bitcoin vs. Bitcoin Cash: Coexistence or Downfall of Bitcoin Cash?. In *Proc. of the IEEE Symposium onSecurity and Privacy (Oakland)*. IEEE.

[15] Kevin Liao and Jonathan Katz. 2017. Incentivizing Blockchain Forks via Whale Transactions. In *Proc. of the International Conference on Financial Cryptography and Data Security (FC)*. Springer.

[16] Loi Luu, Ratul Saha, Inian Parameshwaran, Prateek Saxena, and Aquinas Hobor. 2015. On Power Splitting Games in Distributed Computation: The Case of Bitcoin Pooled Mining. In *Proc. of the IEEE Computer Security Foundations Symposium (CSF)*. IEEE.

[17] Patrick McCorry, Alexander Hicks, and Sarah Meiklejohn. [n.d.]. Smart Contracts for Bribing Miners. *Cryptology ePrint Archive* 2018.

[18] A Miller. 2013. Feather-Forks: Enforcing a Blacklist with Sub-50% Hash Power.

[19] Andrew Miller, James Litton, Andrew Pachulski, Neal Gupta, Dave Levin, Neil Spring, and Bobby Bhattacharjee. [n.d.]. Discovering Bitcoin's Public Topology and Influential Nodes.

[20] Satoshi Nakamoto. 2008. Bitcoin: A Peer-to-Peer Electronic Cash System.

[21] Arvind Narayanan, Joseph Bonneau, Edward Felten, Andrew Miller, and Steven Goldfeder. 2016. *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton University Press.

[22] Kartik Nayak, Srijan Kumar, Andrew Miller, and Elaine Shi. 2016. Stubborn Mining: Generalizing Selfish Mining and Combining with an Eclipse Attack. In *Proc. of the IEEE European Symposium on Security and Privacy (Euro S&P)*. IEEE.

[23] Antpool. 2019. Antpool. https://www.antpool.com/.

[24] Bitcoin Wiki. 2019. Proof of Work. https://en.bitcoin.it/wiki/Proof_of_work.

[25] Bitfinex. 2019. Cloud Mining. https://www.bitfinex.com/.

[26] Bitinfocharts. 2019. Bitcoin Hash Rate. https://bitinfocharts.com/comparison/bitcoin-hashrate.html.

[27] BTC-pools.com. 2019. BTC.com Pool. https://btc-pools.com/.

[28] CEX.IO. 2019. Cloud Mining. https://cex.io/.

[29] Coinbase. 2019. Bitcoin Exchange Rate. https://www.coinbase.com/charts.

[30] Dogecoin Project. 2019. Dogecoin. http://dogecoin.com/.

[31] Litecoin Project. 2019. Litecoin. https://litecoin.org/.

[32] Pow88. 2019. Cloud Mining. http://pow88.com.

[33] Slush. 2019. Slush. https://slushpool.com/home/.

[34] Wikipedia. 2019. Merkle Root. https://en.wikipedia.org/wiki/Merkle_tree.

[35] wizkid057. 2014. BWH Attacks against Eligius. https://bitcointalk.org/?topic=441465.msg7282674.

[36] J Ben Rosen. 1965. Existence and Uniqueness of Equilibrium Points for Concave n-Person Games. *Econometrica: Journal of the Econometric Society*.

[37] Meni Rosenfeld. 2011. Analysis of Bitcoin Pooled Mining Reward Systems. In *arXiv preprint arXiv:1112.4980*.

[38] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. 2016. Optimal Selfish Mining Strategies in Bitcoin. In *Proc. of the International Conference on Financial Cryptography and Data Security (FC)*. Springer.

A CALCULATION OF $\bar{\tau}^{(i)}_{1,\cdots,k}$ (PROOF OF THEOREM 5.1)

We split the total time of finding a new valid block (FPoW) into k slots, as depicted in Fig. 14. In $(0, t_1)$, the attacker will use $\sum_{i\in\mathcal{P}}\tau_1^{(i)}\alpha$ for infiltration mining, where $\mathcal{P}$ is the set of victim pools. Similarly, in (t_1, t_1+t_2), ..., and in $(\sum_{i=1}^{k-1} t_i, \sum_{i=1}^{k} t_i)$, the infiltration mining power will be $\sum_{i\in\mathcal{P}}\tau_2^{(i)}\alpha$, ..., and $\sum_{i\in\mathcal{P}}\tau_k^{(i)}\alpha$ respectively.

We regard all infiltration mining as one miner. Before t_1, the time of the infiltration miner finds an FPoW (T_1) should have $T_1 \sim \exp(\sum_{i\in\mathcal{P}}\tau_1^{(i)}\alpha)$, and the time for the other miners should have $T_k \sim \exp(1-\sum_{i\in\mathcal{P}}\tau_1^{(i)}\alpha)$.

In the calculation of $E(T_1|T_1<T_k)$, we have:

$$\begin{aligned} E(T_1|T_1<T_k) &= \frac{E(T_1\mathbf{1}_{T_1<T_k})}{P(T_1<T_k)} \\ &= \frac{1}{P(T_1<T_k)}\int_{\mathbb{R}} f_{T_k}(y)\int_{-\infty}^{y} x f_{T_1}(x)dxdy \\ &= \frac{1}{P(T_1<T_k)}\sum_{i\in\mathcal{P}}\tau_1^{(i)}\alpha, \end{aligned} \tag{17}$$

where $f_{T_1}(x) = \sum_{i\in\mathcal{P}}\tau_1^{(i)}\alpha e^{-\sum_{i\in\mathcal{P}}\tau_1^{(i)}\alpha x}$ when $x \geqslant 0$ and $f_{T_k}(y) = (1-\sum_{i\in\mathcal{P}}\tau_1^{(i)}\alpha)e^{-(1-\sum_{i\in\mathcal{P}}\tau_1^{(i)}\alpha)y}$ when $y \geqslant 0$.

Now we calculate $P(T_1<T_k)$. First, we considering two independent random variables X and Y that are exponentially distributed, $X\sim\exp(\lambda_1)$ and $Y\sim\exp(\lambda_2)$. The probability of $X<Y$ ($P(X<Y)$) can be calculated as follows:

$$\begin{aligned} P(X<Y) &= \int_{\mathbb{R}}\int_{-\infty}^{y} f_X(x)f_Y(y)dxdy \\ &= \int_0^{+\infty}\lambda_2 e^{-\lambda_2 y}\int_0^{y}\lambda_1 e^{-\lambda_1 x}dxdy = \frac{\lambda_1}{\lambda_1+\lambda_2}. \end{aligned}$$

Now let us consider $P(T_1<T_k)$, where $T_1' \sim \exp(\sum_{i\in\mathcal{P}}\tau_1^{(i)}\alpha)$ and $T_2' \sim \exp(1-\sum_{i\in\mathcal{P}}\tau_1^{(i)}\alpha)$:

$$P(T_1<T_k) = \frac{\sum_{i\in\mathcal{P}}\tau_1^{(i)}\alpha}{\sum_{i\in\mathcal{P}}\tau_1^{(i)}\alpha + (1-\sum_{i\in\mathcal{P}}\tau_1^{(i)}\alpha)} = \sum_{i\in\mathcal{P}}\tau_1^{(i)}\alpha.$$

Therefore, Equation (17) can be calculated as follows:

$$E(T_1|T_1<T_k) = \frac{1}{\sum_{i\in\mathcal{P}}\tau_1^{(i)}\alpha}\cdot\sum_{i\in\mathcal{P}}\tau_1^{(i)}\alpha = 1. \tag{18}$$

Furthermore, consider the time of the infiltration miner finds the second FPoW (T_2). Between t_1 to (t_1+t_2), $T_2 \sim \exp(\sum_{i\in\mathcal{P}}\tau_2^{(i)}\alpha)$, and $T_k \sim \exp(1-\sum_{i\in\mathcal{P}}\tau_2^{(i)}\alpha)$. Based on the memorylessness property of exponential distributions, we have $t_2 = E(T_2|T_1<T_2<T_k) - t_1 = E(T_2|T_2<T_k) = 1$.

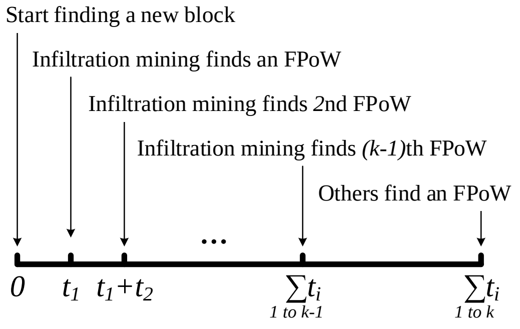
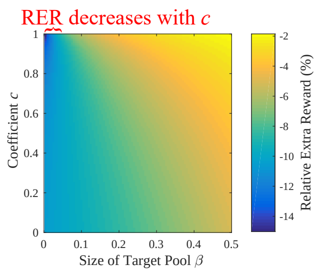
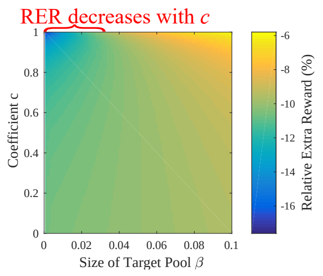

Start finding a new block
Infiltration mining finds an FPoW
Infiltration mining finds 2nd FPoW
Infiltration mining finds (k-1)th FPoW
Others find an FPoW
...
0
t_1
t_1+t_2
$\sum t_i$ 1 to k-1
$\sum t_i$ 1 to k

Figure 14: Splitting the time of finding a new valid block.



(a) $0<\beta\leqslant 0.5$.



(b) $0<\beta\leqslant 0.1$.

Figure 15: RER of the victim pool under FAW attacks when $\alpha = 0.2$. Victim pool's reward does not always increase with c.

Similarly, we have $t_3 = 1$, $t_4 = 1$, ..., $t_{k-1} = 1$, and $t_k = \frac{1}{1-\sum_{i\in\mathcal{P}}\tau_k^{(i)}\alpha}$.

Therefore, we could obtained $\bar{\tau}_{1,\cdots,k}^{(i)}$:

$$\bar{\tau}_{1,\cdots,k}^{(i)} = \frac{(1-\sum_{i\in\mathcal{P}}\tau_k^{(i)}\alpha)\sum_{j=1}^{k-1}\tau_j^{(i)} + \tau_k^{(i)}}{(1-\sum_{i\in\mathcal{P}}\tau_k^{(i)}\alpha)(k-1)+1}.$$

Specifically, when $k=2$, $\bar{\tau} = \bar{\tau}_{1,2}^{(1)} = \frac{\tau_1+\tau_2-\tau_1\tau_2\alpha}{2-\tau_2\alpha}$, where $\tau_1 = \tau_1^{(1)}$ and $\tau_2 = \tau_2^{(1)}$.

B MAXIMIZING $R_a(\tau_1,\tau_2)$

We first rewrite Equation (19) as follows:

$$\begin{aligned} &\arg\min_{\tau_1,\tau_2} \; -R_a(\tau_1,\tau_2), \\ \text{s.t.} \quad & g_1(\tau_1,\tau_2) = -\tau_1 \leqslant 0; \\ & g_2(\tau_1,\tau_2) = \tau_1 - 1 \leqslant 0; \\ & g_3(\tau_1,\tau_2) = -\tau_2 \leqslant 0; \\ & g_4(\tau_1,\tau_2) = \tau_2 - 1 \leqslant 0. \end{aligned} \tag{19}$$

We further introduce four Lagrange multipliers λ_1, λ_2, λ_3, and λ_4 ($\boldsymbol{\lambda} = (\lambda_1,\lambda_2,\lambda_3,\lambda_4)$). The objective function of Equation (19) can be constructed as a Lagrange function $\mathcal{L}(\tau_1,\tau_2,\boldsymbol{\lambda})$:

$$\begin{aligned} \mathcal{L}(\tau_1,\tau_2,\boldsymbol{\lambda}) = &- R_a(\tau_1,\tau_2) - \lambda_1\tau_1 + \lambda_2(\tau_1-1) \\ &- \lambda_3\tau_2 + \lambda_4(\tau_2-1). \end{aligned}$$

The KKT conditions are:

$$\begin{cases} \frac{\partial\mathcal{L}(\tau_1,\tau_2,\boldsymbol{\lambda})}{\partial\tau_1} = 0; \\ \frac{\partial\mathcal{L}(\tau_1,\tau_2,\boldsymbol{\lambda})}{\partial\tau_2} = 0; \\ \lambda_i \geqslant 0; \\ \lambda_i g_i(\tau_1,\tau_2) = 0; \\ g_i(\tau_1,\tau_2) \leqslant 0, \end{cases}$$

where $i \in \{1,2,3,4\}$.

Since the objective function $-R_a(\tau_1,\tau_2)$ is a convex function when $\tau_1,\tau_2 \in [0,1]$ (the Hessian matrix is positive definite), the optimal $\widehat{\tau}_1$ and $\widehat{\tau}_2$ can be found by solving the KKT conditions.

C VICTIM POOL'S REWARD UNDER FAW

Referring to [13], we could obtain the expected reward of the victim pool under FAW attacks:

$$R_p^F = \frac{\beta}{1-\tau_1\alpha} + c\tau_1\alpha\frac{1-\alpha-\beta}{1-\tau_1\alpha}, \tag{20}$$

[13] regards R_p^F as an increasing function with c. However, we think R_p^F *increases with c if τ_1 is fixed.* When considering the optimal $\tau_1 = \widehat{\tau}_1$ to maximize R_p^F, $\widehat{\tau}_1$ ($\bar{\tau}$ in [13]) becomes a function related to c (Equation (2) in [13]). Therefore, we need to involve $\frac{\partial \widehat{\tau}_1}{\partial c}$ when calculating $\frac{\partial R_p^F}{\partial c}$, rather than simply regarding $\frac{\partial \widehat{\tau}_1}{\partial c} = 0$.

We present a specific case to show $R_p^F(\tau_1)$ decreases when c increases in Fig. 15. In this case, we consider $\alpha = 0.2$ and $0 < \beta \leqslant 0.5$. We use the RER of the victim pool under FAW attacks to show the pool's reward.

Fig. 15-a shows the same RER as in [13]. However, when β is small (e.g. $\beta \leqslant 0.02$), victim pool's reward decreases with c. Furthermore, we "zoom in" the area when $\beta \leqslant 0.1$ in Fig. 15-b. The decreasing is more clear. Actually, whether the pool's reward increases or decreases with c is related to α and β (we do not show the expression here since it is too complex).

D CALCULATION OF $R^{[i]}$ IN A TWO-POOL PAW GAME

We calculate the reward of Pool$_i$. Suppose an FPoW from Pool$_1$ is selected as the main chain. Pool$_1$ can earn a profit via innocent mining in five cases: (Case 1) Pool$_1$'s innocent mining finds an FPoW; (Case 2) Pool$_1$'s infiltration mining (in Pool$_2$) first finds an FPoW and Pool$_1$'s innocent mining then finds another FPoW; (Case 3) Pool$_2$'s infiltration mining (in Pool$_1$) first finds an FPoW and Pool$_1$'s innocent mining then finds another FPoW; (Case 4) three FPoWs found by Pool$_1$'s infiltration mining, Pool$_2$'s infiltration mining, and Pool$_1$'s innocent mining in order; and (Case 5) three FPoWs found by Pool$_2$'s infiltration mining, Pool$_1$'s infiltration mining, and Pool$_1$'s innocent mining in order.

The reward of Pool$_1$ in the five cases are:

$$R_1^{[1]}(\text{Case 1}) = \alpha^{[1]} - f_1^{[1]}; \qquad R_1^{[1]}(\text{Case 2}) = f_1^{[1]} \cdot \frac{\alpha^{[1]} - f_2^{[1]}}{1 - f_2^{[1]}};$$

$$R_1^{[1]}(\text{Case 3}) = f_1^{[2]} \cdot \frac{\alpha^{[1]} - f_1^{[1]}}{1 - f_2^{[2]}}; R_1^{[1]}(\text{Case 4}) = f_1^{[1]} \cdot \frac{f_1^{[2]}}{1 - f_2^{[1]}} \cdot \frac{\alpha^{[1]} - f_2^{[1]}}{1 - f_2^{[1]} - f_2^{[2]}};$$

$$R_1^{[1]}(\text{Case 5}) = f_1^{[2]} \cdot \frac{f_1^{[1]}}{1 - f_2^{[2]}} \cdot \frac{\alpha^{[1]} - f_2^{[1]}}{1 - f_2^{[1]} - f_2^{[2]}}.$$

We further consider Pool$_1$'s reward of causing a fork when an FPoW from Pool$_1$ is selected as the main chain. Specifically, we have three cases: (Case 6) Pool$_2$'s infiltration mining (in Pool$_1$) first finds an FPoW, and other miners then find a valid block; (Case 7) three FPoWs found by Pool$_1$'s infiltration mining, Pool$_2$'s infiltration mining, and other miners in order; and (Case 8) three FPoWs found by Pool$_2$'s infiltration mining, Pool$_1$'s infiltration mining, and other miners in order. The expected reward of three cases are:

$$R_1^{[1]}(\text{Case 6}) = c_1^{[2]} \cdot f_1^{[2]} \cdot \frac{1 - \alpha^{[1]} - \alpha^{[2]}}{1 - f_2^{[2]}};$$

$$R_1^{[1]}(\text{Case 7}) = c_2^{[2]} \cdot f_1^{[1]} \cdot \frac{f_1^{[2]}}{1 - f_2^{[1]}} \cdot \frac{1 - \alpha^{[1]} - \alpha^{[2]}}{1 - f_2^{[1]} - f_2^{[2]}};$$

$$R_1^{[1]}(\text{Case 8}) = c_2^{[2]} \cdot f_1^{[2]} \cdot \frac{f_1^{[1]}}{1 - f_2^{[2]}} \cdot \frac{1 - \alpha^{[1]} - \alpha^{[2]}}{1 - f_2^{[1]} - f_2^{[2]}}.$$

The reward of Pool$_1$ (not including the shared reward from Pool$_2$) is the sum of the eight cases $R_1^{[1]} = \sum_{i=1}^{8} R_1^{[1]}(\text{Case } i)$.

Now we consider the shared reward of Pool$_2$ from $R_1^{[1]}$. For each case, Pool$_1$ will share his reward with Pool$_2$. In Case 1 and Case 2, Pool$_2$ will share $\frac{f_1^{[2]}}{\alpha^{[1]}+f_1^{[2]}}$ reward; in Case 3 and Case 6 will share $\frac{\bar{f}_{1,2}^{[2]}}{\alpha^{[1]}+\bar{f}_{1,2}^{[2]}}$ reward; in Case 4 and Case 7 will share $\frac{\bar{f}_{1,1,2}^{[2]}}{\alpha^{[1]}+\bar{f}_{1,1,2}^{[2]}}$ reward; and in Case 5 and Case 8 will share $\frac{\bar{f}_{1,2,2}^{[2]}}{\alpha^{[1]}+\bar{f}_{1,2,2}^{[2]}}$ reward ($\bar{f}_{1,2}^{[2]}$, $\bar{f}_{1,1,2}^{[2]}$, and $\bar{f}_{1,2,2}^{[2]}$ are Pool$_2$'s average infiltration mining power in the respective cases). The share of Pool$_2$ from $R_1^{[1]}$ is:

$$\begin{aligned} R_2^{[2]} = \sum_{i=1}^{8} R_2^{[2]}(\text{Case } i) = & \frac{f_1^{[2]}}{\alpha^{[1]}+f_1^{[2]}} \Big(R_1^{[1]}(\text{Case 1}) + R_1^{[1]}(\text{Case 2})\Big) \\ & + \frac{\bar{f}_{1,2}^{[2]}}{\alpha^{[1]}+\bar{f}_{1,2}^{[2]}} \Big(R_1^{[1]}(\text{Case 3}) + R_1^{[1]}(\text{Case 6})\Big) \\ & + \frac{\bar{f}_{1,1,2}^{[2]}}{\alpha^{[1]}+\bar{f}_{1,1,2}^{[2]}} \Big(R_1^{[1]}(\text{Case 4}) + R_1^{[1]}(\text{Case 7})\Big) \\ & + \frac{\bar{f}_{1,2,2}^{[2]}}{\alpha^{[1]}+\bar{f}_{1,2,2}^{[2]}} \Big(R_1^{[1]}(\text{Case 7}) + R_1^{[1]}(\text{Case 8})\Big). \end{aligned}$$

When Pool$_2$ earns a shared reward, it will also share the reward with Pool$_1$'s infiltration mining, which is $\frac{f_1^{[1]}}{\alpha^{[2]}+f_1^{[1]}}$ shared reward in Case 1, Case 3, and Case 6; $\frac{\bar{f}_{1,2}^{[1]}}{\alpha^{[2]}+\bar{f}_{1,2}^{[1]}}$ in Case 2; $\frac{\bar{f}_{1,1,2}^{[1]}}{\alpha^{[2]}+\bar{f}_{1,1,2}^{[1]}}$ in Case 5 and Case 8; and $\frac{\bar{f}_{1,2,2}^{[1]}}{\alpha^{[2]}+\bar{f}_{1,2,2}^{[1]}}$ in Case 4 and Case 7. Therefore, the reward of Pool$_1$ from this share is:

$$\begin{aligned} R_3^{[1]} = & \frac{f_1^{[2]}}{\alpha^{[1]} + f_1^{[2]}} \frac{f_1^{[1]}}{\alpha^{[2]} + f_1^{[1]}} R_1^{[1]}(\text{Case 1}) \\ & + \frac{f_1^{[2]}}{\alpha^{[1]} + f_1^{[2]}} \frac{\bar{f}_{1,2}^{[1]}}{\alpha^{[2]} + \bar{f}_{1,2}^{[1]}} R_1^{[1]}(\text{Case 2}) \\ & + \frac{\bar{f}_{1,2}^{[2]}}{\alpha^{[1]} + \bar{f}_{1,2}^{[2]}} \frac{f_1^{[1]}}{\alpha^{[2]} + f_1^{[1]}} \Big(R_1^{[1]}(\text{Case 3}) + R_1^{[1]}(\text{Case 6})\Big) \\ & + \frac{\bar{f}_{1,1,2}^{[2]}}{\alpha^{[1]} + \bar{f}_{1,1,2}^{[2]}} \frac{\bar{f}_{1,2,2}^{[1]}}{\alpha^{[2]} + \bar{f}_{1,2,2}^{[1]}} \Big(R_1^{[1]}(\text{Case 4}) + R_1^{[1]}(\text{Case 7})\Big) \\ & + \frac{\bar{f}_{1,2,2}^{[2]}}{\alpha^{[1]} + \bar{f}_{1,2,2}^{[2]}} \frac{\bar{f}_{1,1,2}^{[1]}}{\alpha^{[2]} + \bar{f}_{1,1,2}^{[1]}} \Big(R_1^{[1]}(\text{Case 5}) + R_1^{[1]}(\text{Case 8})\Big). \end{aligned}$$

Similarly, we can derive the reward of Pool$_1$ from $R_n^{[1]}$, and the reward of Pool$_2$ from $R_n^{[2]}$ as follows. To simplify the expression, we introduce $Q(a,b) = \frac{a}{\alpha^{[1]}+a}\frac{b}{\alpha^{[2]}+b}$.

$$\begin{aligned}R^{[1]}_{n+2} &= Q(f^{[2]}_1, f^{[1]}_1)R^{[1]}_n(\text{Case 1}) + Q(f^{[2]}_1, \bar{f}^{[1]}_{1,2})R^{[1]}_n(\text{Case 2})\\ &+ Q(\bar{f}^{[2]}_{1,2}, f^{[1]}_1)\left(R^{[1]}_n(\text{Case 3}) + R^{[1]}_n(\text{Case 6})\right)\\ &+ Q(\bar{f}^{[2]}_{1,1,2}, \bar{f}^{[1]}_{1,2,2})\left(R^{[1]}_n(\text{Case 4}) + R^{[1]}_n(\text{Case 7})\right)\\ &+ Q(\bar{f}^{[2]}_{1,2,2}, \bar{f}^{[1]}_{1,1,2})\left(R^{[1]}_n(\text{Case 5}) + R^{[1]}_n(\text{Case 8})\right),\end{aligned}$$

when n is a positive odd number.

$$\begin{aligned}R^{[2]}_{n+2} &= Q(f^{[2]}_1, f^{[1]}_1)R^{[2]}_n(\text{Case 1}) + Q(f^{[2]}_1, \bar{f}^{[1]}_{1,2})R^{[2]}_n(\text{Case 2})\\ &+ Q(\bar{f}^{[2]}_{1,2}, f^{[1]}_1)\left(R^{[2]}_n(\text{Case 3}) + R^{[2]}_n(\text{Case 6})\right)\\ &+ Q(\bar{f}^{[2]}_{1,1,2}, \bar{f}^{[1]}_{1,2,2})\left(R^{[2]}_n(\text{Case 4}) + R^{[2]}_n(\text{Case 7})\right)\\ &+ Q(\bar{f}^{[2]}_{1,2,2}, \bar{f}^{[1]}_{1,1,2})\left(R^{[2]}_n(\text{Case 5}) + R^{[2]}_n(\text{Case 8})\right),\end{aligned}$$

when n is a positive even number.

When n approaches infinity, $\sum R^{[1]}_n$ (with an odd n) and $\sum R^{[2]}_n$ (with an even n) can be calculated as:

$$\begin{aligned}\sum_{\substack{n=2k+1\\k\in\mathbb{N}}} R^{[1]}_n = &\frac{R^{[1]}_1(\text{Case 1})}{1-Q(f^{[2]}_1, f^{[1]}_1)} + \frac{R^{[1]}_1(\text{Case 2})}{1-Q(f^{[2]}_1, \bar{f}^{[1]}_{1,2})} + \frac{R^{[1]}_1(\text{Case 3})+R^{[1]}_1(\text{Case 6})}{1-Q(\bar{f}^{[2]}_{1,2}, f^{[1]}_1)}\\ &+ \frac{R^{[1]}_1(\text{Case 4})+R^{[1]}_1(\text{Case 7})}{1-Q(\bar{f}^{[2]}_{1,1,2}, \bar{f}^{[1]}_{1,2,2})} + \frac{R^{[1]}_1(\text{Case 5})+R^{[1]}_1(\text{Case 8})}{1-Q(\bar{f}^{[2]}_{1,2,2}, \bar{f}^{[1]}_{1,1,2})};\\ \sum_{\substack{n=2k\\k\in\mathbb{N}^*}} R^{[2]}_n = &\frac{R^{[2]}_2(\text{Case 1})}{1-Q(f^{[2]}_1, f^{[1]}_1)} + \frac{R^{[2]}_2(\text{Case 2})}{1-Q(f^{[2]}_1, \bar{f}^{[1]}_{1,2})} + \frac{R^{[2]}_2(\text{Case 3})+R^{[2]}_2(\text{Case 6})}{1-Q(\bar{f}^{[2]}_{1,2}, f^{[1]}_1)}\\ &+ \frac{R^{[2]}_2(\text{Case 4})+R^{[2]}_2(\text{Case 7})}{1-Q(\bar{f}^{[2]}_{1,1,2}, \bar{f}^{[1]}_{1,2,2})} + \frac{R^{[2]}_2(\text{Case 5})+R^{[2]}_2(\text{Case 8})}{1-Q(\bar{f}^{[2]}_{1,2,2}, \bar{f}^{[1]}_{1,1,2})}.\end{aligned}$$

Notice that the calculation of shared reward of Pool$_1$ when Pool$_2$ first finds an FPoW is exactly same as the calculation of shared reward of Pool$_2$ when Pool$_1$ first finds an FPoW ($\sum R^{[2]}_n$). We can derive the reward of Pool$_i$ in a two-pool PAW game:

$$R^{[i]} = \sum_{n\in\mathbb{N}^*} R^{[i]}_n = \sum_{\substack{n=2k+1\\k\in\mathbb{N}}} R^{[i]}_n + \sum_{\substack{n=2k\\k\in\mathbb{N}^*}} R^{[i]}_n.$$

$\bar{f}^{[i]}_{1,2}$, $\bar{f}^{[i]}_{1,1,2}$, and $\bar{f}^{[i]}_{1,2,2}$ can be calculated as:

$$\begin{aligned}\bar{f}^{[i]}_{1,2} &= \frac{f^{[i]}_1 + f^{[i]}_2 - f^{[i]}_1 f^{[i]}_2}{2 - f^{[i]}_2};\\ \bar{f}^{[i]}_{1,1,2} &= \frac{2f^{[i]}_1(1 - f^{[1]}_2 - f^{[2]}_2) + f^{[i]}_2}{3 - 2(f^{[1]}_2 + f^{[2]}_2)};\\ \bar{f}^{[i]}_{1,2,2} &= \frac{(f^{[i]}_1 + f^{[i]}_2)(1 - f^{[1]}_2 - f^{[2]}_2) + f^{[i]}_2}{3 - 2(f^{[1]}_2 + f^{[2]}_2)}.\end{aligned}$$

Detailed proof can refer to Theorem 5.1 and Appendix-A.

E PROOF OF THE NASH EQUILIBRIUM (THEOREM 5.3)

Proof. To prove that there exists a Nash equilibrium, it suffices to show $\nabla_{\boldsymbol{f}^{[1]}}(\nabla_{\boldsymbol{f}^{[1]}} R^{[1]}) < 0$ and $\nabla_{\boldsymbol{f}^{[2]}}(\nabla_{\boldsymbol{f}^{[2]}} R^{[2]}) < 0$ under the following conditions:

$$\begin{aligned}&0 \leqslant f^{[1]}_1, f^{[1]}_2 \leqslant \alpha^{[1]} \leqslant 1;\\ &0 \leqslant f^{[2]}_1, f^{[2]}_2 \leqslant \alpha^{[2]} \leqslant 1;\\ &\alpha^{[1]} + \alpha^{[2]} \leqslant 1;\\ &0 \leqslant c^{[1]}_1, c^{[2]}_1 \leqslant 1;\\ &0 \leqslant c^{[1]}_2 + c^{[2]}_2 \leqslant 1.\end{aligned} \tag{21}$$

Therefore, there exists a unique Nash equilibrium point since $R^{[1]}$ and $R^{[2]}$ are convex functions with $\boldsymbol{f}^{[1]}$ and $\boldsymbol{f}^{[2]}$ respectively [36].

Furthermore, we use Best-response dynamics to find the Nash equilibrium point. Specifically, we let Pool$_1$ and Pool$_2$ start at $(\boldsymbol{f}^{[1]}_0, \boldsymbol{f}^{[2]}_0) = ((0,0),(0,0))$, and adjust $\boldsymbol{f}^{[1]}$ and $\boldsymbol{f}^{[2]}$ to maximize $R^{[1]}$ and $R^{[2]}$ respectively, till $\boldsymbol{f}^{[1]}$ and $\boldsymbol{f}^{[2]}$ converge. For instance, we first update $\boldsymbol{f}^{[1]}_1$ to maximize $R^{[1]}$, and then update $\boldsymbol{f}^{[2]}_1$ to maximize $R^{[2]}$. After that, we repeat the procedures to maximize $R^{[1]}$ with $\boldsymbol{f}^{[1]}_2$ and $R^{[2]}$ with $\boldsymbol{f}^{[2]}_2$, and so on. At k-th step, $\boldsymbol{f}^{[1]}_k$ and $\boldsymbol{f}^{[2]}_k$ can be represented by:

$$\boldsymbol{f}^{[1]}_k = \underset{\boldsymbol{f}^{[1]}}{\arg\max}\ R^{[1]}(\boldsymbol{f}^{[1]}, \boldsymbol{f}^{[2]}_{k-1}), \quad \boldsymbol{f}^{[2]}_k = \underset{\boldsymbol{f}^{[2]}}{\arg\max}\ R^{[2]}(\boldsymbol{f}^{[1]}_k, \boldsymbol{f}^{[2]}),$$

which is under the constraints in Equation (21).

The Nash equilibrium point will be found when $\boldsymbol{f}^{[1]}$ and $\boldsymbol{f}^{[2]}$ converge as k approaches infinity, which either satisfies $\nabla_{\boldsymbol{f}^{[1]}} R^{[1]} = 0$ and $\nabla_{\boldsymbol{f}^{[2]}} R^{[2]} = 0$, or a point on a borderline in Equation (21) which maximizes $R^{[1]}$ with $\boldsymbol{f}^{[1]}$ and $R^{[2]}$ with $\boldsymbol{f}^{[2]}$. $\square$

F BRIBERY RACING GAME

In our previous analysis, we regard the attacker races with an "honest" opponent. Now we consider the opponent also carry out bribery racing to avoid a loss. Suppose two miners a_1 and a_2 fall into the "0-lead" racing situation. When a_2 is aware of bribes ε_1, he will broadcast a transaction $T^{A_2}_B$ which transfer ε_2 from his mining reward address A_2 to an "anyone-can-claim" address B as bribes. In such scenarios, bribery racing will become a bribery racing game, and targets (venal miners) will choose to work on a more profitable branch. Thus, the probability of all other miners (including targets) follows a_i's branch γ_i will be an increasing function with ε_i (more bribes will bring more targets).

Now we analyze the Nash equilibrium of a bribery racing game. Since we consider common "0-lead" racing scenarios, we use the reward of the "forked" block as our objective function (different from BSM or Bribery PAW which should use the attacker's reward as the objective function). The probability of extending a_i's branch is $\alpha_i + \gamma_i(1 - \alpha_1 - \alpha_2)$. Therefore, a_i's reward is:

$$R_{a_i} = (1 - \varepsilon_i)(\alpha_i + \gamma_i(1 - \alpha_1 - \alpha_2)), \quad i \in \{1,2\}.$$

We then use Best-response dynamics to find the Nash equilibrium point. Let a_1 and a_2 start at $(\varepsilon_{1,0}, \varepsilon_{1,0}) = (0,0)$. a_i will adjust ε_i to maximize R_{a_i} as the best response to the opponent, till R_{a_1} and R_{a_2} converge. For example, we first update $\varepsilon_{1,1}$ to maximize R_{a_1}, and then update $\varepsilon_{2,1}$ to maximize R_{a_2}. After that, we repeat the procedures to maximize R_{a_1} with $\varepsilon_{1,2}$ and R_{a_2} with $\varepsilon_{2,2}$, and so on. At k-th step, $\varepsilon_{1,k}$ and $\varepsilon_{2,k}$ can be represented by:

$$\varepsilon_{1,k} = \underset{\varepsilon_1}{\arg\max}\ R_{a_1}(\varepsilon_1, \varepsilon_{2,k-1}), \quad \varepsilon_{2,k} = \underset{\varepsilon_2}{\arg\max}\ R_{a_2}(\varepsilon_{1,k}, \varepsilon_2).$$

Moreover, a bribery racing game can become more interesting when also considering a_i as venal miners: a_i can even be rewarded when working on the opponent's branch. For instance, when a_1 and a_2 pay too much for bribes, they will consider working on the opponent's branch when claiming the bribes becomes more profitable. We can use similar methods above to analysis this situation and get the same conclusion in most scenarios.

G BRIBERY FAW/PAW

We discuss strategies to combine bribery racing with FAW/-PAW attacks.

Out-of-band payment. An attacker first launches FAW/-PAW against victim pools. When "0-lead" racing occurs (case 4-2 in Section 5.1), the attacker directly pays the owners of mining capacity (other miners or "cloud mining" services [25, 28, 32]) to work on the attacker's branch. The payment can be made in Bitcoin or any outside (state) currency. Furthermore, bribing through smart contracts [17] can also be applied to make bribery less visible and difficult to be detected (out-of-band payment can also be used to bribe pool managers in in-band payment, which will be discussed latter).

Negative-fee mining pool. Negative-fee mining pools provide pool miners higher rewards than honest mining to lure miners (bribes) join in the pool [2]. When combined with FAW attacks, an attacker works as a negative-fee mining pool manager and announces a higher reward to encourage miners to join in (i.e., bribing other miners). Meanwhile, the attacker uses her loyal mining power [7] as infiltration mining in a victim pool (different from the negative-fee mining pool). When "0-lead" racing occurs (case 4-2 in Section 5.1), the attacker can force negative-fee pool miners working on her branch since she is the pool manager.

In-band payment. We discuss two strategies: bribing pool manager and Eclipse attacks [11] to launch B-FAW/B-PAW attacks via in-band payment.

First, an attacker can bribe the pool manager to choose T_B^A. When the manager is profit-driven (only cares about the profit of himself), he may accept the bribes and make all pool miners encapsulate T_B^A. Since the bribery can be done via out-of-band payment, bribing manager can be hardly detected. However, victim pool miners can listen to the transactions and count the number of T_B^A when both T_B^A and $T_{A'}^A$ are received to identify the corrupted manager. Moreover, for an honest manager, they can reject the bribes and expel the attacker.

Second, an attacker can launch Eclipse attacks to block the global view of the victim pool manager [11]. With Eclipse attacks, the attacker can filter out $T_{A'}^A$ to ensure the pool manager only sees T_B^A. However, successfully launching Eclipse attacks against the victim pool requires a non-trivial cost.

Suppose the victim pool manager chooses to record T_B^A instead of $T_{A'}^A$ in B-FAW/B-PAW. An attacker will firstly encapsulate the bribery transaction T_B^A when calculating shares in her infiltration mining power since "0-lead" racing can only be caused by the withheld FPoWs. Second, after discovering an FPoW, the attacker withholds the FPoW, broadcasts $T_{A'}^A$ in the network, and sends T_B^A to the (corrupted/eclipsed) victim pool manager. Third, when other miners find a block, the attacker immediately submits the withheld FPoW to the pool manager to cause a fork. Notice that when the next FPoW is also found by the infiltration mining, the attacker should submit immediately to the manager immediately to win in forks as with selfish mining. Therefore, the second round of infiltration mining does not need to contain bribes.

Let's first consider "0-lead" racing in B-PAW. We first consider the attacker races with other miners. When the target accepts the bribes, the attacker's branch can be extended by herself, victim pool miners, the target, or γ portion of other miners. Second, when the attacker races with the target, her branch can be extended by herself, victim pool miners, or γ portion of other miners. In either scenario, the attacker needs to pay one block of bribes. For other cases (not "0-lead" racing), the attacker's reward is the same as the PAW reward. Therefore, we can derive the attacker's reward in "0-lead" racing.

We compare the attacker's reward with FAW, PAW, and B-PAW (target accepting bribes) in a specific case, where the attacker, victim pool, and target with computational power 0.1, 0.2, 0.2 respectively. The attacker will set $\varepsilon = 0.02$ as bribes. We show attacker's RERs in terms of γ in Fig. 16. When the attacker chooses a proper ε, the B-PAW reward will be higher than both PAW and FAW reward.

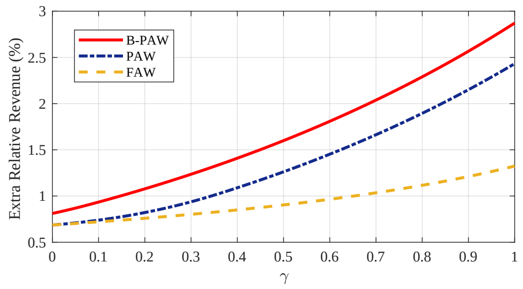


Figure 16: Attacker's RERs against one pool with FAW, PAW, and B-PAW, according to γ when the attacker, victim pool, and target with computational power 0.1, 0.2, 0.2 respectively.

H ORIENTED BRIBERY ATTACKS

The original bribery attacks adopt transactions with scripts allowing anyone to claim the bribes. It may result in many targets race to claim the funds. A large pool may not willing to compete with other solo miners for the bribes. Besides, original bribery attacks cannot control the size of target pools. To overcome these limitations, we suggest that the attacker create transactions ($T_{B_i}^A$) to specific targets (B_i) in her private chain and spend the bribes on the public chain. Since creating a transaction only needs the public key of the receiver (i.e., hash of B_i's public key), $T_{B_i}^A$ can be created without the confirmation of B_i. The targets will get the bribes automatically when the attacker's branch is selected as the main chain (without competing with other miners). To obtain the public key, the attacker can join in the target pool or just check out the previously mined blocks by the target (the hash of the public key is included in the transaction to claim the system reward, and is accessible for anyone). Attackers can also use the oriented bribery attacks to lure better targets with higher bribes.