If You Can't Beat Them, Pay Them: Bitcoin Protection Racket is Profitable

Zheng Yang
Southwest University
Chongqing, China
youngzheng@swu.edu.cn

Chao Yin
Vrije University
Amsterdam, Netherlands
c.yin@vu.nl

Junming Ke
University of Tartu
Tartu, Estonia
junmingke1994@gmail.com

Tien Tuan Anh Dinh
Singapore University of Technology and Design
Singapore
dinhtta@sutd.edu.sg

Jianying Zhou
Singapore University of Technology and Design
Singapore
jianying_zhou@sutd.edu.sg

ABSTRACT

Pooled mining has become the most popular mining approach in the Bitcoin system, which can effectively reduce the variance of the block generation reward of participants. The security of pooled mining depends on whether it is incentive compatible, that is, an honest participant will get a reward proportional to his work. Recent attacks on mining pools, for example, Block Withholding, Fork After Withholding, and Power Adjusting Withholding (PAW) attacks, show that malicious participants may undermine the revenue of the honest pools and receive an unfair share of the mining reward. This paper shows that the security of Bitcoin is even worse than what the recent attacks demonstrated. We describe an attack called Fork Withholding Attack under a Protection Racket (FWAP), in which the mining pool pays the attacker for withholding a fork. Our insight is that the mining pools under forking attacks have incentives to pay in exchange for not being forked. The attacker and the paying pool negotiate how much to be paid, and we show that it is possible for both the attacker and the paying pool to earn higher rewards at the expense of the other pools. In particular, our formal analysis and simulation demonstrate that the payer and the FWAP attacker can get up to 1.8× and 3.8× of extra reward as in PAW, respectively. Furthermore, FWAP can escape from the "miners' dilemma" when two FWAP attackers attack each other under some circumstances. We also propose simple approaches that serve as the first step towards preventing the FWAP attack.

CCS CONCEPTS

• **Security and privacy → Distributed systems security**.

*Zheng Yang and Chao Yin contribute equally and share the first authorship. Chao Yin is the corresponding author.

ACSAC '22, December 5–9, 2022, Austin, TX, USA

ACM ISBN 978-1-4503-9759-9/22/12…$15.00
https://doi.org/10.1145/3564625.3567983

KEYWORDS

Bitcoin, mining attack, block withholding attack, fork after withholding, power adjusting, protection racket.

ACM Reference Format:
Zheng Yang, Chao Yin, Junming Ke, Tien Tuan Anh Dinh, and Jianying Zhou. 2022. If You Can't Beat Them, Pay Them: Bitcoin Protection Racket is Profitable. In *Annual Computer Security Applications Conference (ACSAC '22), December 5–9, 2022, Austin, TX, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3564625.3567983

1 INTRODUCTION

The success of Bitcoin helps jump-start a new era of decentralized computing [29]. Bitcoin and some other open blockchain systems adopt proof-of-work (PoW) protocols for ensuring security in the decentralized settings [7, 22, 23, 35, 39]. Participants (or nodes) in the blockchain maintain a ledger data structure recording cryptocurrency transactions. PoW allows honest nodes to keep a consistent ledger, despite Byzantine behavior of a minority of nodes. To propose a block, a node must solve an expensive computation puzzle regarding PoW, a process called *mining*. If a miner solves the puzzle, and gets its block included in the blockchain, it earns a reward (for example, 6.25 Bitcoins as of June 2022). The security of PoW rests on the assumption that a majority of nodes are honest, which implies that the blockchain must incentivize nodes to behave honestly. The current incentive mechanism is to reward nodes with native cryptocurrencies when they find a new block.

As the network grows, for example, due to the popularity of Bitcoin, the mining difficulty increases, making it harder for an individual (or solo) miner to find a new block. To maintain steady rewards, multiple miners form mining pools in which they solve the same puzzle and get rewards proportional to their contributions to the pool. In fact, mining pools account for up to 90% of the total network mining power [17, 18, 34]. Most mining pools [17, 18, 34] are open, which allow any miner to join and leave. Such pools can be infiltrated by dishonest miners who join in order to undermine the pool's operations with dishonest mining strategies. Due to the large number of the miners in a pool, it is challenging for the pool manager to identify and eliminate such infiltrator [16, 24].

Blockchain nodes can be rational rather than purely honest. In particular, they expect to get rewards proportional with their effort, otherwise they might leave the network or join the attacker. As a

consequence, the pooled mining must be *incentive compatible* in order to maintain a majority of honest nodes. However, there are attacks demonstrating that pool mining is not incentive compatible [6, 15, 16, 19, 24, 32], in which the attacker exploits mining pools to get more rewards than honest nodes who spend the same effort.

One example of attacking the mining pool is the Block Withholding (BWH) attack [19, 32], in which the attacker infiltrates a victim pool and only produces incomplete solutions to the puzzles (or Partial Proofs of Work, PPoW) for that pool. When it finds full solutions to the puzzles (or Full Proofs of Work, FPoW) for the victim pool, it withholds them. This attack reduces the victim pool's probability of finding a block, thereby increasing the winning probability for the attacker. In 2014, this attack was launched against the mining pool Eligius [37] resulting in a loss of 300 Bitcoins of honest miners. Another attack example is selfish mining (SM) [6, 15], in which the attacker keeps mining on the blocks she discovers, rather than discarding them as in BWH. A powerful SM attacker might have a private chain formed by the withheld blocks which will be selectively propagated to cause a fork when other miners find a block.

Fork After Withholding (FAW) [24] attacks extends BWH, in which the attacker only broadcasts the FPoW she discovers when pools other than the victim pool find a block. The attacker's goal in FAW is to generate fork to compete with miners outside of the victim pool. FAW allows the attacker to earn more rewards than BHW, while avoiding the "miner's dilemma" that arises when two attackers launch FAW against each other. Two recent attacks, namely Power Adjusting Withholding (PAW) and Bribery Selfish Mining (BSM) [16] increases the attacker's reward even further. In PAW, the attacker can reallocate its capacity between infiltration mining and innocent mining. In BSM, she bribes other miners to select its branch as the main blockchain.

In this paper, we propose a new attack against Bitcoin pooled mining that gives the attacker higher rewards than in PAW. Our insight is that a pool has incentives to pay the attacker to be spared from forking attacks. We refer to such payment as *protection money* (PM), and the paying pool as the *colluding pool*. We note that the protection money is different to bribery in BSM, in which the attacker pays the other miners. Our attack, called Fork Withholding Attack With Protection (FWAP), extends PAW with a new strategy based on protection money. The intuition behind FWAP is that the colluding pool always gains "extra reward" (called colluding reward) when the attacker stops creating forks against it and keeps attacking other pools. In FWAP, the attacker first demonstrates its ability to create forks against a colluding pool. This step is done out-of-band without revealing the attacker's identity. Next, the colluding pool and attacker negotiate a payment ratio μ of the colluding reward. Once the payment is received, the attacker discards the fork. We show that with properly chosen values of μ, it is possible to have a win-win situation, i.e., both the attacker and the colluding pool gains more rewards than in PAW.[1]

Contributions. We summarize the contributions of this paper as follows:

- We present a new attack called Fork Withholding Attack With Protection (FWAP), in which a colluding pool pays money to the attacker with then discards the forks created against the pool. That the attacker can gain more rewards than in the state-of-the-art attack, i.e., PAW, gives another evidence of the weakness of Bitcoin's pooled mining.
- We analyze FWAP formally and using simulation. Our analysis takes into account both the number of victim pools and the game between multiple FWAP attackers. It focuses on the protection money's lower bound, μ, that results in win-win situations. We show that the colluding pool and the FWAP attacker can earn up to 1.8× and 3.8× more extra rewards than in PAW (based on our pricing function of μ).
- We introduce an end-to-end instantiation of the FWAP attack to show how the attacker establishes the protection racket with a colluding pool. We also discuss and propose practical countermeasures that serve as first steps towards preventing the FWAP attack (including our instantiation of protection racket).

Organization. Section 2 describes the preliminaries and related work. Section 3 introduces the threat model and attack assumptions. Section 4 presents an overview of FWAP, followed by more details in Section 5. Section 6 discusses the pricing scheme. Section 7 discusses an attack game to analyze the "miner's delimma". Section E describes how to instantiate FWAP. Section 8 discusses several mitigations against FWAP before Section 9 concludes.

2 BACKGROUND & RELATED WORK

In this section, we review the basic concepts of Bitcoin and discuss related attacks on pooled mining protocols.

2.1 Bitcoin Mining

Mining. Mining is a process of finding a valid block to be added to the blockchain, which involves solving a hard computational puzzle. Specifically, each block in the blockchain contains a header with the Merkle root [27] of the transactions in the block, the hash of the previous block, a timestamp, and a nonce. The nonce is the solution to the computation puzzle, which is chosen such that the hash of the block's header blkH is lower than or equal to a target difficulty d, i.e., $\mathsf{SHA256}(\mathsf{SHA256}(\mathsf{blkH})) < d$. The nonce serves as the proof-of-work (PoW). d is adjusted dynamically to ensure that on average a block is generated every ten minutes. When a miner finds a valid nonce, it gets a reward, for instance 6.25 Bitcoins.

Mining Pools. As Bitcoin becomes popular, the number of miners increases, leads to higher d and therefore making it more difficult for a solo miner to find an FPoW. To maintain steady reward, multiple miners form a mining pool [9, 26], in which they all find blocks on behalf of the pool. The pool rewards are then divided among the miners according their contribution. The mining pool is operated by a pool manager who assigns work to the miners at the start of each round. Each miner shows its contribution in the form of PPoW or FPoW. The former are easier than the latter. Once the manager receives an FPoW, it broadcasts the corresponding block to the network to get a reward which it shares with contributors.

Forks. When multiple miners broadcast valid blocks at roughly the same time, the blockchain is forked. In particular, there are

[1] The extra reward helps deter colluding pool from exposing the attacks.

multiple branches extending from the same block. Although all the branches are valid, only the longest branch will be selected as the main chain, and other branches are discarded. Only miners whose blocks are in the main chain would get the rewards. This mechanism can be exploited to give an attacker more rewards than honest miners [2, 15, 20, 24, 28].

2.2 Related Work

BWH Attack. Rosenfeld [32] proposes an attack, called Block Withholding (BWH), in which the attacker joins (or infiltrates) a victim pool to sabotage its chance of getting rewards by submitting only PPoW to the pool. The attacker discards any FPoW it finds for the pool, thereby reducing the pool's block generation rate. The attacker can split its mining power between infiltration mining in the victim pool to share the reward of the pool, and solo mining to generate the blocks for itself. This strategy is highly effective as it earns the attacker more rewards [13]. A real-world BWH attack was mounted against "Eligius" mining pool, causing it to lose 300 BTC [37]. The BWH attack is difficult to detect if the attacker changes its account frequently. Luu et al. [25] propose a game-theoretic approach for splitting the mining power when targeting one or multiple pools that results in an improved reward for the attacker. When BWH attackers target each other, Eyal [14] demonstrates the "miner's dilemma" in which the attackers both suffer a loss.

The fact that the BWH attacks can help increasing reward probability of non-victim pools is exploited in *Sponsored Block Withholding* (S-BWH) attack [4], in which a sponsoring pool colludes with the attacker to spend a fraction of its computing power to target a pool of its choice. This strategy earns the attacker extra rewards from the sponsorship which is proportional to the damage inflicted to the victim pool. Our work differs from S-BWH in that we consider forks, and our threat model is different. In particular, the attacker withholds attacks against the pools that pay protection money.

FAW Attack. This attack extends BWH with strategies using forks [6, 15, 30, 33]. In an FAW attack, the infiltration miner would adaptively decide to withhold or submit an FFoW. That is, an FFoW of the infiltration miner will be submitted the pool manger to cause a fork if and only if the other miners (outside the victim pool) find a valid block. Unlike selfish mining, which may not always be profitable, the FAW attacker earns more rewards than the BWH. In particular, there is a 56% higher reward for the attacker. At the same time, there is no "miner's dilemma", since one pool with a larger size can consistently win in the two-pool attack game against another pool.

PAW and B-PAW Attacks. Gao et al. [16] propose two extensions to previous attacks that increase the attacker's reward even further. The first extension is to combine a power adjusting strategy with FAW, which leads to a new attack called Power Adjusting Withholding (PAW). We review the high-level idea of PAW in Appendix A. PAW achieves 2.5× higher reward for the attacker than FAW, without suffering from the "miner's dilemma". The second extension is a bribery strategy that is combined with selfish mining, which results in a new attack called Bribery Selfish Mining (BSM). BSM gives the attacker get 10% more rewards than selfish mining.

3 SECURITY MODEL

This section outlines the attack model and the assumptions made in the rest of the paper.

Attack Model. The attacker can be a solo miner or a mining pool manager. The attacker can create many identities and Bitcoin accounts, using them to join multiple open mining pools. We assume that private pools cannot be infiltrated because they require private membership. The attacker has limited computational power, but it can dynamically split this power into *innocent mining* and *infiltration mining*. The former is for finding the block as a normal miner, the latter is for joining other pools. When the manager of an open pool is an attacker, her infiltration mining power is the "loyal mining power" [14] which is a secret. The attacker can plant Sybil nodes [3] in the Bitcoin network to track the propagation of valid blocks, and to speed up the propagation of its own blocks. These Sybil nodes do not consume mining power [24]. The attacker can bribe other miners or pools as in Bribery attack [10]. It can accept payments from others and stop attacking them. We call a pool that makes such payment as a *colluding pool*.

Assumptions. We made the following assumptions to simplify our analysis. We note that all except for the last assumption are with those in existing mining attacks [5, 14, 16, 24].

- The total mining power of the system is normalized to 1. To avoid "51% attack" [12], a miner or a pool's mining power must be less than 0.5.
- There are only intentional forks in the system. This is reasonable because the fork rates are negligible (the stale block rate is about 0.41% [14, 16, 24]). As a result, a miner's expected reward is equal to its probability of finding a valid block. The time for a miner to find a block follows an exponential distribution with mean inversely to his computational power. Therefore, a miner's probability of finding a block is equal to his mining power.
- The reward of a valid block is normalized to 1 BTC instead of 6.25 BTC (for simplicity), and the block reward for each round is computed as a probabilistic expectation [16, 24].
- When a pool miner finds an FPoW, the manager propagates the corresponding block, then splits the reward based on the PPoW submitted by each pool miner.
- We assume that the attacker only launches the attack proposed in this work, that is, it does not perform other attacks.
- Except the attacker, other miners are honest but rational. That is, they do not directly attack each other, but will follow the most profitable mining strategy [14].
- Some miners will pay an attacker to avoid losing rewards due to forks.

4 OVERVIEW

We propose a new attack strategy based on *protection money*. In particular, a mining pool pays an attacker for withholding forks against its chain. We combine this strategy with PAW attack to derive a new attack called FWAP.

Fig. 1 illustrates general idea of the FWAP attack. The attacker infiltrates a *victim pool*. When it finds an FPoW inside this victim pool, it waits until another pool, called *colluding pool*, finds a block (① in the figure). Then, instead of broadcasting the FPoW immediately to create a fork, the attacker sends a request for protection

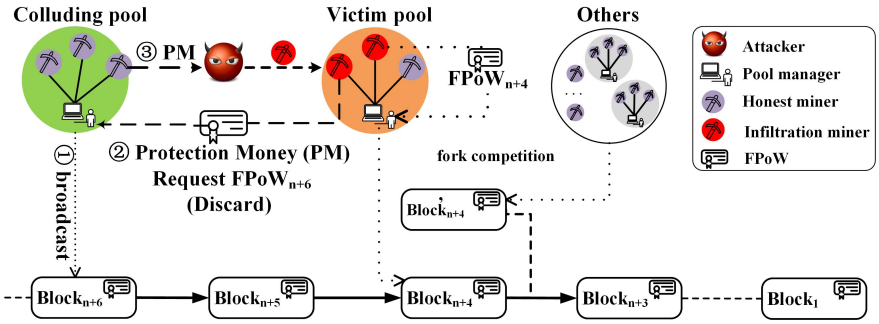


Figure 1: Overview of FWAP.

money to the colluding pool (②). This request proves the attacker's ability to create a fork. The colluding pool verifies the request and pays the requested money (③). Once payment is received, the attacker discards the FPoW. We discuss in Section E how the attacker establishes an agreement with the colluding pool that will pay protection money. Meanwhile, other honest miners would still be the forking targets of the attacker. Even so, other miners' reward will be improved due to FWAP, so they may also expect the FWAP attack to someone else.

To determine the source of protection money, we define *colluding reward* as the difference between the colluding pool's reward under the FWAP attack and that under the PAW attack. Our first observation is that the colluding reward is always non-zero because the attacker does not generate forks against it (while she keeps attacking other honest mining pools). In other words, it can be profitable for the colluding pool to pay protection money to earn the colluding reward. Our second observation is that the reward of an infiltration miner gained by creating and winning a fork against a pool has to be shared with other miners in the victim pool, which can be less than what the colluding pool offers to pay. This means it is also profitable for the attacker to engage in the protection racket. Since both the attacker and the colluding pool can profit, a rational pool is incentivized to collude and join the protection racket.

Our main contribution is to derive a range of the fraction μ of the colluding reward such that the attacker ends up with a higher reward than simply creating forks against the colluding pool. We obtain the lower bound of μ such that both the attacker and the colluding pool have higher rewards than in the PAW attack.

One Victim Pool. The FWAP attacker splits its mining power into innocent mining (i.e., as a solo miner) and infiltration mining in one victim pool. The attack is similar to PAW, except when the attacker finds an FPoW in the victim pool. In the FWAP, the colluding pool pays protection money so that the attacker withholds this FPoW. We show how the attack parameters can be chosen to ensure a non-zero colluding reward. When $\mu < 1$, the colluding pool's reward is improved by $(1-\mu)$ fraction of the colluding reward. In the next section, we derive the lower bound of μ to obtain a win-win situation, that is, both the attacker and the colluding pool get more rewards than in PAW.

Multiple Victim Pools. The FWAP attacker can split her mining power to infiltrate multiple victim pools to maximize her reward. If lucky enough, it can find multiple FPoWs in a given round, one for each victim pool. This increases the attacker's probability of winning the forks against other honest mining pools. We will derive the lower bound for μ with n victim pools.

Attack Game. To earn more rewards, FWAP attackers may execute the attack against each other, i.e., by distributing infiltration miners to her opponent, leading to an attack game similar to previous mining attacks [16, 24]. Our analysis with two attacking pools shows that FWAP does not suffer from "miner's dilemma" [14], and that the game's outcome depends on the pools' sizes.

5 FWAP ATTACK

In this section, we elaborate on the new FWAP attack in different aforementioned scenarios.

5.1 One Victim Pool

Theoretical Analysis. We formally analyze the rewards of the FWAP attacker and the colluding pool in one victim pool case. Some attack parameters and notations are summarized as follows:

- α: Computational power of the attacker;
- β: Computational power of the victim pool;
- η: Computational power of the colluding pool;
- ξ: Computational power of other miners which are none of attacker, colluding pool, and victim pool;
- τ_1: Attacker's original infiltration mining power as a proportion of α before power adjusting;
- τ_2: Attacker's reallocated infiltration mining power as a proportion of α after power adjusting;
- $\bar{\tau}$: Attacker's average portion of computational power allocated to infiltration mining in a mining process;
- c: Probability of the attacker's FPoW that is selected as the main chain in a fork;
- R_{cp}^{P}: Colluding pool's rewards in PAW;
- R_{cp}^{PM}: Colluding pool's rewards in FWAP;
- R_{cp}^{Df}: Colluding reward $R_{cp}^{Df} = R_{cp}^{PM} - R_{cp}^{P}$;
- R_m: Colluding pool's protection money;
- μ: Protection money ratio, i.e., $R_m = \mu \cdot R_{cp}^{Df}$.

Comparing to PAW, we here have a few more parameters. We use η to denote the computational power of the colluding pool, and let μ denote the protection money ratio that is defined as a proportion of the extra reward of the colluding pool. Unlike in the previous attacks, the computational power of other miners is $\xi = 1-\alpha-\beta-\eta$. Namely, it additionally excludes the computational power of the colluding pool. Note that the computational power of the victim pool β does not include the computational power of infiltration miners. Moreover, we highlight that the network capability c of an attacker can be determined following the approach in [24, §9],

which is lower bounded by $\alpha + \beta$ with rational pool managers and can reach a value over 0.9 in practice.

As in the PAW attack, we consider an FWAP attacker who can adjust the infiltration power when her infiltrating miner finds an FPoW in a given round. The difference between FWAP attack and PAW attack is that the fork target of FWAP attack does not include the colluding pool. Specifically, we analyze the attack results with the following cases:

Case 1: The attacker finds an FPoW and earns a legitimate reward via innocent mining.

Case 2: Other miners find an FPoW. The attacker will accept the block and keep mining. So she cannot earn any reward in this case.

Case 3: Colluding pool finds an FPoW. This case is similar to Case 2, in which the attacker earns nothing but continues mining the next block.

Case 4: Honest miners in the victim pool find an FPoW. The attacker gets a shared reward from the pool in terms of her infiltration mining power.

Case 5: Infiltration miner finds an FPoW. The attacker withholds the block and adjusts the infiltration power. In the meantime, we consider the following sub-cases (in the given round) regarding the situations that another FPoW is found:

Case 5-1: By innocent mining. The attacker discards the withheld block and propagates her new block.

Case 5-2: By other miners. The attacker immediately submits the withheld FPoW to the victim pool manager. A fork will be generated if the manager decides to propagate her FPoW.

Case 5-3: By the colluding pool. The attacker discards the withheld block and accepts the protection money.

Case 5-4: By honest miners in the victim pool. The attacker discards the withheld block and shares the reward of the victim pool.

The above cases are adapted from the PAW attack by incorporating two new cases, i.e., Case 3 and Case 5-3. The Case 3 would not affect the attacker's reward. But the attacker can receive protection money from the colluding pool in Case 5-3. We will show the protection money would incentive the attacker to adjust her attack parameters to get more reward. The other cases are similar in PAW. Before power adjusting, the Case 1 occurs with probability $(1-\tau_1)\alpha$. The occurrence probabilities of Case 2 and Case 3 are ξ and η, respectively. The attacker would fall in Case 4 with probability β, and into Case 5 with probability $\tau_1\alpha$. Note that, after adjusting the computational power in Case 5, the total mining power will become $(1-\tau_2)\alpha$ because the infiltration mining will not mine FPoWs in that round. As a result, the occurrence probabilities of Case 5-1, 5-2, 5-3 and 5-4 are $\tau_1\alpha \cdot \frac{(1-\tau_2)\alpha}{1-\tau_2\alpha}$, $\tau_1\alpha \cdot \frac{\xi}{1-\tau_2\alpha}$, $\tau_1\alpha \cdot \frac{\eta}{1-\tau_2\alpha}$, $\tau_1\alpha \cdot \frac{\beta}{1-\tau_2\alpha}$, respectively. Nevertheless, the sum of the probability in Case 5-1, 5-2, 5-3 and 5-4 is $\tau_1\alpha$.

Protection Money. We assume that the FWAP attacker and the colluding pool have a secret pact with the protection money ratio μ to execute the FWAP attack. Here we will focus on studying the constraints of μ to make the FWAP attack effective. In Section 6, we will discuss how to price the protection money in detail. We let $\mathrm{R}_{\mathrm{cp}}^{\mathrm{PM}}$ be the colluding pool's reward in FWAP before she pays protection money, and let $\widetilde{\mathrm{R}_{\mathrm{cp}}^{\mathrm{PM}}}$ be the colluding pool's reward after she pays protection money. With the same parameters (i.e., α, β, and c), we can obtain the reward of the colluding pool in the PAW attack denoted by $\mathrm{R}_{\mathrm{cp}}^{\mathrm{P}}$ which will be used as our reward baseline.

Generally speaking, the colluding pool will take μ fraction of her colluding reward $\mathrm{R}_{\mathrm{cp}}^{\mathrm{Df}}$ as protection money. The minimum protection money must be sufficient to guarantee the reward of the attacker in the FWAP attack is greater than that in the PAW attack.

To calculate the colluding reward $\mathrm{R}_{\mathrm{cp}}^{\mathrm{Df}}$, we should get the colluding pool's reward $\mathrm{R}_{\mathrm{cp}}^{\mathrm{P}}$ in the PAW attack and her reward $\mathrm{R}_{\mathrm{cp}}^{\mathrm{PM}}$ in the FWAP attack. Note that the other miners' computational power is $1-\alpha-\beta$ in the PAW attack but $1-\alpha-\beta-\eta$ in the FWAP attack. This is because the other miners' mining power in an FWAP attack does not include the colluding pool's mining power. Namely, other miners' computing power is reduced in the FWAP attack, so the FWAP attacker should figure out optimal infiltration mining power to maximize her reward. We let τ_1 and τ_2 be the optimal infiltration mining power before and after power adjusting in the FWAP attack, respectively. And we let τ_1' and τ_2' be the optimal infiltration mining power before and after power adjusting in the PAW attack, respectively. Then, we can get the colluding pool's reward $\mathrm{R}_{\mathrm{cp}}^{\mathrm{P}}$ in the PAW attack as $\mathrm{R}_{\mathrm{cp}}^{\mathrm{P}} = \eta + (1-\mathrm{c})\tau_1'\alpha\frac{\eta}{1-\tau_2'\alpha}$, and the colluding pool's reward $\mathrm{R}_{\mathrm{cp}}^{\mathrm{PM}}$ in the FWAP attack as

$$\mathrm{R}_{\mathrm{cp}}^{\mathrm{PM}} = \eta + \tau_1\alpha\frac{\eta}{1-\tau_2\alpha}.$$

Then, the colluding reward can be represented as

$$\mathrm{R}_{\mathrm{cp}}^{\mathrm{Df}} = \mathrm{R}_{\mathrm{cp}}^{\mathrm{PM}} - \mathrm{R}_{\mathrm{cp}}^{\mathrm{P}} = \tau_1\alpha\frac{\eta}{1-\tau_2\alpha} - (1-\mathrm{c})\tau_1'\alpha\frac{\eta}{1-\tau_2'\alpha}.$$

Now, we can calculate protection money as a portion of the colluding reward $\mathrm{R}_{\mathrm{cp}}^{\mathrm{Df}}$ as

$$\mathrm{R}_{\mathrm{m}} = \mu\cdot\mathrm{R}_{\mathrm{cp}}^{\mathrm{Df}} = \mu(\mathrm{R}_{\mathrm{cp}}^{\mathrm{PM}} - \mathrm{R}_{\mathrm{cp}}^{\mathrm{P}}) = \mu(\tau_1\alpha\frac{\eta}{1-\tau_2\alpha} - (1-\mathrm{c})\tau_1'\alpha\frac{\eta}{1-\tau_2'\alpha}).$$

We can further derive the reward $\mathrm{R}_{\mathrm{a}}^{\mathrm{PM}}(\tau_1,\tau_2)$ of an FWAP attacker from the protection money and the rewards in all other cases as follows:

$$\begin{aligned}\mathrm{R}_{\mathrm{a}}^{\mathrm{PM}}(\tau_1,\tau_2) = {}& \mu(\tau_1\alpha\frac{\eta}{1-\tau_2\alpha} - (1-\mathrm{c})\tau_1'\alpha\frac{\eta}{1-\tau_2'\alpha})(1-\tau_1)\alpha + \\ & + \beta\frac{\tau_1\alpha}{\beta+\tau_1\alpha} + \tau_1\alpha\left(\frac{(1-\tau_2)\alpha}{1-\tau_2\alpha} + (\frac{\beta}{1-\tau_2\alpha} + c\frac{\xi}{1-\tau_2\alpha})\frac{\bar{\tau}\alpha}{\beta+\bar{\tau}\alpha}\right).\end{aligned} \tag{1}$$

Since the μ is pre-defined and the R_{m} in Eq. (1) can be regarded as a function related to τ_1 and τ_2, we can calculate the optimal infiltration mining power τ_1, τ_2, and $\bar{\tau}$ respectively. Hence, with known μ and other parameters, the attacker can determine the optimal infiltration power to maximum her rewards. The reward $\mathrm{R}_{\mathrm{a}}^{\mathrm{PM}}(\tau_1,\tau_2)$ of an FWAP attacker (defined by Eq. (1)) consists of three parts, i.e., the innocent mining rewards in Cases 1 and 5-1, the infiltration mining rewards in Cases 4, 5-2 and 5-4, and protection money received in Case 5-3. Specifically, we have the innocent mining reward $(1-\tau_1)\alpha + \tau_1\alpha\cdot\frac{(1-\tau_2)\alpha}{1-\tau_2\alpha}$. Since the attacker will share the profit with honest miners in the victim pool, the reward for infiltration mining is the sum the rewards in the corresponding cases, i.e., $\beta\cdot\frac{\tau_1\alpha}{\beta+\tau_1\alpha} + \mathrm{c}\tau_1\alpha\frac{\xi}{1-\tau_2\alpha}\frac{\bar{\tau}\alpha}{\beta+\bar{\tau}\alpha} + \tau_1\alpha\frac{\beta}{1-\tau_2\alpha}\frac{\bar{\tau}\alpha}{\beta+\bar{\tau}\alpha}$.

When the colluding pool can not afford enough protection money to incentivize the attacker for running FWAP attack, the attacker will run PAW attack instead. In the meantime, the colluding pool's reward should be improved in the FWAP attack as well after paying protection money. Otherwise, the colluding pool prefers to be forked as in the PAW attack. In the following, we are going to show these requirements can be satisfied in the FWAP attack. Our strategy is first to present that the colluding pool can have a non-zero colluding reward when $\mathrm{c} > 0$, so it has protection money to pay

the FWAP attacker via Lemma 5.1. On the second, we will show that an FWAP attacker can earn more rewards than honest mining via Theorem 5.2, so the attacker may choose to execute the attack. Then, we will show that FWAP is better than PAW via Theorem 5.3 under certain circumstances. That is, we will prove that the FWAP attacker can always enable the colluding pool to have enough colluding reward and set a proper μ to increase both the reward of herself and the colluding pool comparing to the PAW attack.

Lemma 5.1. *The colluding reward is always greater than zero when coefficient $c > 0$.*

Proof. We show that colluding pool can always increase her reward by paying protection money when c $>$ 0 (implying the probability on winning in forks). We consider that the FWAP attacker set $(\tau_1, \tau_2) = (\tau_1', \tau_2')$ first, since a FWAP attacker was a PAW attacker before collecting protection money. The protection money could bring more rewards to the attacker, she will find the optimal (τ_1, τ_2) based on the reward with (τ_1', τ_2') (i.e., the optimal (τ_1, τ_2) will maximize the attacker's reward $\mathrm{R}_{\mathrm{a}}^{\mathrm{PM}}$).

With (τ_1', τ_2'), the colluding pool can get reward

$$\mathrm{R}_{\mathrm{cp}}^{\mathrm{PM}}(\tau_1', \tau_2') = \eta + \tau_1'\alpha\frac{\eta}{1-\tau_2'\alpha}$$

in the FWAP attack, and can earn the reward in the PAW attack as

$$\mathrm{R}_{\mathrm{cp}}^{\mathrm{P}}(\tau_1', \tau_2') = \eta + (1-\mathrm{c})\cdot\tau_1'\alpha\frac{\eta}{1-\tau_2'\alpha}.$$

As well, we can calculate the colluding reward with (τ_1', τ_2') as

$$\mathrm{R}_{\mathrm{cp}}^{\mathrm{Df}}(\tau_1', \tau_2') = \mathrm{R}_{\mathrm{cp}}^{\mathrm{PM}}(\tau_1', \tau_2') - \mathrm{R}_{\mathrm{cp}}^{\mathrm{P}}(\tau_1', \tau_2') = \mathrm{c}\cdot\tau_1'\alpha\frac{\eta}{1-\tau_2'\alpha} > 0. \quad (2)$$

Since $\mathrm{R}_{\mathrm{cp}}^{\mathrm{Df}}(\tau_1', \tau_2')$ is non-zero, this proves this lemma. □

Theorem 5.2. *An FWAP attacker can always earn more rewards than honest mining, and the reward of an FWAP attacker has a lower bound defined by the reward from an BWH attack.*

Theorem 5.3. *For $\mu \in (\frac{\bar{\tau}'\alpha}{\beta+\bar{\tau}'\alpha}, 1)$, an FWAP attacker can earn more rewards than an PAW attacker when $c > 0$.*

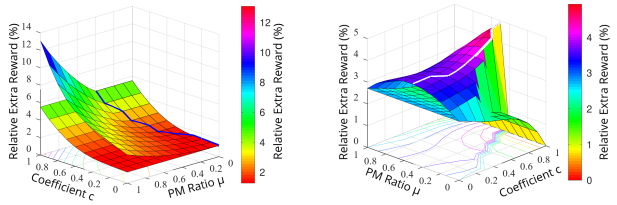
We present the proofs of Theorem 5.2 and Theorem 5.3 in Appendix B and Appendix C, respectively.

Theorem 5.4. *For arbitrary $\mu \in (0, 1)$, the colluding pool can always get more rewards in the FWAP attack than that in the PAW attack.*

Proof. Since the fork-wining probability c of an attacker is assumed to be non-zero, the colluding pool may suffer from some loss of reward in the PAW attack. On the other side, if the attacker holds the block without creating any fork against the colluding pool in the FWAP attack, such a reward loss would become $\mathrm{R}_{\mathrm{cp}}^{\mathrm{Df}}$ for $\mu = 0$. Hence, the colluding reward can be non-zero. Even if the colluding pool pays $\mu \cdot \mathrm{R}_{\mathrm{cp}}^{\mathrm{Df}}$ (for $\mu < 1$) as protection money to compensate the attacker, it can still have an extra reward $(1-\mu)\mathrm{R}_{\mathrm{cp}}^{\mathrm{Df}}$ comparing to its reward in the PAW attack. In a nutshell, the colluding pool will have greater rewards in FWAP than in PAW for any $\mu \in (\frac{\bar{\tau}'\alpha}{\beta+\bar{\tau}'\alpha}, 1)$. Since μ defines the fraction of colluding reward being paid as protection money, which is smaller than 1, so we can have that $(1-\mu)\mathrm{R}_{\mathrm{cp}}^{\mathrm{Df}} > 0$. □

The result of Theorem 5.4 guarantees that a rational colluding pool is willing to join the FWAP attack. Once the attacker chooses to execute the FWAP attack with a pre-determined μ, she has motivation to adjust its attack parameters (i.e., τ_1 and τ_2) to increase the colluding reward to gain more rewards. The attacker could use a similar approach in PAW to determine the optimal τ_1 and τ_2.

Quantitative Analysis. We use specific attack parameters to show the effectiveness of FWAP attack. Following the approach in [16, 24], we define the expected relative extra reward (RER) $\mathrm{RER}_x^S = \frac{\mathrm{R}_x^S - \mathrm{R}_x^H}{\mathrm{R}_x^H}$ for comparison, where S represents different mining strategies that can be honest mining (H), FWAP (PM), and PAW (P), x denotes an entity that can be attacker (a), victim pool (b) and colluding pool (cp), and R_x^S represents the reward of x with strategy S. Initially, we have $\mathrm{R}_a^H = \alpha$, $\mathrm{R}_b^H = \beta$, $\mathrm{R}_{cp}^H = \eta$ with honest mining strategy. In the following analysis, we consider a specific attacker with computational power $\alpha = 0.2$, who executes the attacks against a victim pool with computational power $\beta = 0.2$ and conspires with a colluding pool with computational power $\eta = 0.2$.



(a) $\mathrm{RER}_a^{\mathrm{PM}}$ and $\mathrm{RER}_a^{\mathrm{P}}$ **(b)** $\mathrm{RER}_{cp}^{\mathrm{PM}}$ and $\mathrm{RER}_{cp}^{\mathrm{P}}$

Figure 2: Quantitative analysis results against one pool. (a) and (b) show the RERs of the FWAP attacker and the colluding pool, respectively, with varying c and μ, and constant computational power of attacker $\alpha = 0.2$, victim pool $\beta = 0.2$, and colluding pool $\eta = 0.2$.

Fig. 2(a) shows the attacker's $\mathrm{RER}_a^{\mathrm{PM}}$ and $\mathrm{RER}_a^{\mathrm{P}}$ in FWAP and PAW, respectively. The upper surface is $\mathrm{RER}_a^{\mathrm{PM}}$, and the lower surface represents $\mathrm{RER}_a^{\mathrm{P}}$. The RERs of FWAP are proportional to μ and c. The larger μ and c are, the higher reward of attacker is. So the attacker can earn more rewards in FWAP than in PAW with the μ divided by the *blue-lined boundary* shown in Fig. 2(a). For $0.99 \le \mu < 1$ and $c = 1$, the RER of the FWAP attacker is about 2.3× as in PAW.

Fig. 2(b) shows the $\mathrm{RER}_{cp}^{\mathrm{PM}}$ and $\mathrm{RER}_{cp}^{\mathrm{PM}}$ of the colluding pool in two kinds of attacks. The upper surface represents $\mathrm{RER}_{cp}^{\mathrm{PM}}$, and the lower surface is $\mathrm{RER}_{cp}^{\mathrm{P}}$. Note that Fig. 2(b) only shows the $\mathrm{RER}_{cp}^{\mathrm{PM}}$ with valid range of μ since the attacker would not run the FWAP attack with the invalid μ. The optimal μ (that maximizes the colluding pool's extra reward) falls in the range between 0.4 to 0.6, which is highlighted with a *white line*. For $\mu = 0.5$ and $c = 0.6$, the colluding pool's RER in FWAP is about 2.5× higher than that in PAW. Nevertheless, the colluding pool can always gain an extra reward if μ is within the valid range defined by Theorem 5.3. Even if we use parameters $\mu = 0.95$ (in attacker's interests) and $c = 1$, the FWAP attack still gives the colluding pool an extra reward of 50% more than the PAW attack.

In Table 1, we show a few lower bounds of the protection money ratio μ and the optimal ones to the colluding pool. The lower bound of μ is inversely proportional to the size of the victim pool, since

the infiltration mining would share less in a bigger pool (note that the protection money only needs to be more than the attacker's shared reward in the victim pool). And the lower bound of μ in all cases is not big, which makes the FWAP attack easy to implement.

Table 1: Examples of protection money ratio μ. The values x(y) indicate the lower bound (x) and optimal value (y) of μ in a single-pool attack scenario, respectively.

β	c = 0	c = 0.25	c = 0.5	c = 0.75	c = 1
0.1	0 (0.54)	0.27 (0.50)	0.49 (0.49)	0.56 (0.52)	0.58 (0.61)
0.2	0 (0.56)	0.07 (0.48)	0.25 (0.42)	0.40 (0.39)	0.44 (0.42)
0.3	0 (0.59)	0.08 (0.49)	0.09 (0.41)	0.21 (0.35)	0.33 (0.36)

5.2 Multiple Victim Pools

Theoretically Analysis. Here we will analyze the attack scenario that the attacker distributes infiltration mining power into n (for $n \geq 2$) distinct victim pools $(p_1, p_2, \ldots, p_n)$ to maximize its reward. In this analysis, we shall use the following additional parameters:

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

We denote with $[n] = \{1, \ldots, n\} \subset \mathbb{N}$ the set of integers between 1 and n. Recall that the attacker's reward comprises of the reward of innocent mining, the shares from victim pools, the reward of creating branches, and the protection money from the colluding pool. Before calculating these rewards, we first consider two cases regarding infiltration mining:

Case 1: No infiltration miner finds FPoW;

Case 2: i infiltration miners find FPoWs for $i \in [n]$.

In Case 1, the total infiltration mining power is $\sum_{k=1}^{n} \tau_1^{(p_k)}\alpha$, and the total infiltration mining power become $\sum_{k=1}^{n} \tau_{\mathrm{i+1}}^{(p_k)}\alpha$ in Case 2 since the attacker would adjust infiltration mining power whenever the infiltration mining finds an FPoW, where $\tau_{\mathrm{i+1}}^{(p_k)}\alpha$ denotes the power adjusting result after i FPoWs were found in the pool p_k.

Income from innocent mining. From innocent mining, the attacker can get a reward $(1 - \sum_{k=1}^{n} \tau_1^{(p_k)})\alpha$ in Case 1. In Case 2, the innocent mining power is $(1 - \sum_{k=1}^{n} \tau_{\mathrm{1+i}}^{(p_k)})\alpha$. We consider the infiltration mining in pools p_1, p_2, …, and p_i finds FPoWs sequentially (in order) before the innocent mining discovers a block. The innocent mining can get a reward

$$(1 - \sum_{k=1}^{n} \tau_{\mathrm{i+1}}^{(p_k)})\alpha \cdot \prod_{j=1}^{i} \frac{\tau_{\mathrm{j}}^{(p_j)}\alpha}{1 - \sum_{k=1}^{j} \tau_{\mathrm{j+1}}^{(p_k)}\alpha}. \tag{3}$$

Since each of the time orders of these found FPoWs has a different probability and causes different rewards to the attacker, we need to consider all the possible numbers and orders of FPoWs found before the innocent mining. By summing up the innocent mining reward in all cases. Based on Eq. (3), we get the total reward R_{inno} of innocent mining as

$$\mathrm{R}_{inno} = (1 - \sum_{k=1}^{n} \tau_1^{(p_k)})\alpha + \sum_{i=1}^{n} \sum_{\mathbf{p}_i \in \mathcal{P}^i} (1 - \sum_{k=1}^{n} \tau_{\mathrm{i+1}}^{(p_k)})\alpha \cdot \mathrm{PO}_{\mathbf{p}_i}, \tag{4}$$

where $\mathcal{P}^i$ denotes all possible FPoW-found orders among i victim pools $\hat{p_1}, \hat{p_2}, \ldots, \hat{p_i}$ (where each $\hat{p_j} \in (p_1, p_2, \ldots, p_n$ and $j \in [i])$, and $\mathbf{p}_i$ is one kind of orders in $\mathcal{P}^i$, and $\mathrm{PO}_{\mathbf{p}_i} = \prod_{j=1}^{i} \frac{\tau_{\mathrm{j}}^{(p_j)}\alpha}{1-\sum_{k=1}^{j} \tau_{\mathrm{j+1}}^{(p_k)}\alpha}$ which is relevant to $\mathbf{p}_i$ and will appear in many other places later.

Share from victim pools. Similarly, we calculate the shares from victim pools in two cases, respectively. In Case 1, the victim pool p_i can get a reward β_i when an honest miner in p_i finds an FPoW before any infiltration miners did. Let $\mathrm{SH}_0^{(p_k)} = \frac{\tau_1^{(p_k)}\alpha}{\tau_1^{(p_k)}\alpha+\beta_k}$ be the proportion of infiltration mining power in the victim pool p_k in Case 1. Therefore, infiltration miners in all victim pools can get a shared reward $\sum_{k=1}^{n} \beta^{(p_k)}\mathrm{SH}_0^{(p_k)} = \sum_{k=1}^{n} \beta^{(p_k)} \frac{\tau_1^{(p_k)}\alpha}{\tau_1^{(p_k)}\alpha+\beta_k}$. When an honest miner in victim pool p_k finds an FPoW in Case 2, the attacker can share $\mathrm{SH}_i^{(p_k)} = \frac{\bar{\tau}_{1,\ldots,i+1}^{(p_k)}\alpha}{\beta^{(p_k)}+\bar{\tau}_{1,\ldots,i+1}^{(p_k)}\alpha}$ fraction of the victim pool p_k's reward. Considering all possible k and i, the attacker can share a reward $\sum_{i=1}^{n} \sum_{\mathbf{p}_i \in \mathcal{P}^i} \left(\sum_{k=1}^{i} \mathrm{SH}_i^{((p_k))} \beta^{(p_k)} \mathrm{PO}_{\mathbf{p}_i} \right)$. To sum up the share in each case, we have the following total reward R_{share} from sharing the victim pools' reward

$$\mathrm{R}_{share} = \sum_{k=1}^{n} \beta^{(p_k)}\mathrm{SH}_0^{(p_k)} + \sum_{i=1}^{n} \sum_{\mathbf{p}_i \in \mathcal{P}^i} \left(\sum_{k=1}^{i} \mathrm{SH}_i^{(p_k)} \beta^{(p_k)} \mathrm{PO}_{\mathbf{p}_i} \right) \tag{5}$$

Income from Creating branches. Since there is no fork in Case 1, we here focus on the rewards of infiltration miners in Case 2. As there are at most i infiltration miners can find FPoWs in Case 2 (by assumption), the attacker can get a share from wining in the fork as long as one of the blocks submitted by the infiltration miners is selected as the main chain. The total reward R_{fork} of creating branches is the sum of the rewards from all the cases of generating a fork, which is calculated as follows

$$\mathrm{R}_{fork} = \sum_{i=1}^{n} \sum_{\mathbf{p}_i \in \mathcal{P}^i} \left(\sum_{k=1}^{i} \mathrm{SH}_i^{(p_k)} \mathrm{c}_i^{(p_k)} \xi \cdot \mathrm{PO}_{\mathbf{p}_i} \right). \tag{6}$$

Income from Protection money. The protection money obtained in the multiple-pool scenario is analyzed by the following lemma.

Lemma 5.5. *When the attacker executes the FWAP attacks against n victim pools, she can get protection money*

$$R_m = \mu\eta \sum_{i=0}^{n} \sum_{\mathbf{p}_i \in \mathcal{P}^i} \left(PO_{\mathbf{p}_i} - (1 - \sum_{k=1}^{i} \mathrm{c}_i^{(p_k)}) \widehat{PO_{\mathbf{p}_i}} \right), \tag{7}$$

where $\widehat{PO_{\mathbf{p}_i}} = \prod_{j=1}^{i} \frac{\widehat{\tau_j^{(p_j)}}\alpha}{1-\sum_{k=1}^{j} \widehat{\tau_{j+1}^{(p_k)}}\alpha}$.

Proof. Note that the protection money is μ fraction of the colluding reward which is the difference between the colluding pool's rewards in the FWAP attack and in the PAW attack. In a PAW attack, the colluding pool can get rewards from honest mining and winning a fork. Since the colluding pool has probability $1 - \sum_{k=1}^{i} \mathrm{c}_i^{p(k)}$ to win in a fork with $i+1$ branches, it can earn a reward $\eta + \sum_{i=0}^{n} \sum_{\mathbf{p}_i \in \mathcal{P}^i} \left((1 - \sum_{k=1}^{i} \mathrm{c}_i^{(p_k)})\eta \cdot \widehat{\mathrm{PO}_{\mathbf{p}_i}} \right)$.

In an FWAP attack, the colluding pool can get the following reward from honest mining $\eta + \sum_{i=0}^{n} \sum_{\mathbf{p}_i \in \mathcal{P}^i} \left(\eta \cdot \mathrm{PO}_{\mathbf{p}_i} \right)$. Then we

can derive the equation of colluding reward as

$$\mathrm{R}_{\mathrm{cp}}^{\mathrm{Df}} = \eta \sum_{i=0}^{n} \sum_{\mathbf{p}_i \in \mathcal{P}^i} \left(\mathrm{PO}_{\mathbf{p}_i} - (1 - \sum_{k=1}^{i} \mathrm{c}_i^{(p_k)}) \cdot \widehat{\mathrm{PO}_{\mathbf{p}_i}} \right). \tag{8}$$

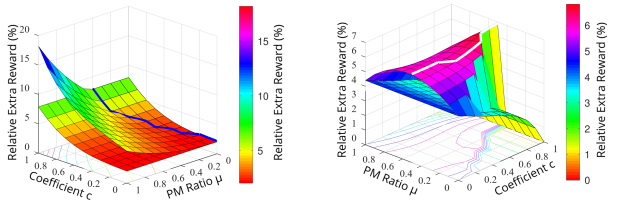
From Eq. (8), we can derive protection money $\mathrm{R_m}$ in Eq. (7). □

Theorem 5.6. *For $\mu \in (\rho, 1)$, an FWAP attacker can earn more rewards than an PAW attacker when attacking n victim pools, where* $\rho = \frac{\sum_{i=1}^{n} \sum_{\mathbf{p}_i \in \mathcal{P}^i} \left\{ \sum_{k=1}^{i} \widehat{SH_i^{(p_k)}}\, c_i^{(p_k)}\, \widehat{PO_{\mathbf{p}_i}} \right\}}{\sum_{i=1}^{n} \sum_{\mathbf{p}_i \in \mathcal{P}^i} \left\{ \sum_{k=1}^{i} c_i^{(p_k)}\, \widehat{PO_{\mathbf{p}_i}} \right\}}$, *and* $\widehat{SH_i^{(p_k)}} = \frac{\widehat{\bar{\tau}_{1,\ldots,i+1}^{(p_k)}}\alpha}{\beta^{(p_k)} + \widehat{\bar{\tau}_{1,\ldots,i+1}^{(p_k)}}\alpha}$.

The proof of this theorem is given in Appendix D. Eventually, the total reward of an FWAP attacker against n pools can be obtained by summing up the rewards in Eq. (4), (5), (6), and (7):

$$\mathrm{R}_{\mathrm{a}}^{\mathrm{PM}} = \mathrm{R}_{inno} + \mathrm{R}_{share} + \mathrm{R}_{fork} + \mathrm{R_m} \tag{9}$$

Quantitative Analysis. We consider a special case with two victim pools (two-pool) to show the RER (%) of the attacker in an FWAP attack. In our analysis, we use the similar parameters as in the single victim pool scenario, i.e., $\alpha = 0.2$ and $\eta = 0.2$ but we set the computational power of two victim pools as $\beta_1 = 0.2$ and $\beta_2 = 0.1$.



(a) $\mathrm{RER}_a^{\mathrm{PM}}$ and $\mathrm{RER}_a^{\mathrm{P}}$ **(b)** $\mathrm{RER}_{cp}^{\mathrm{PM}}$ and $\mathrm{RER}_{cp}^{\mathrm{P}}$

Figure 3: Quantitative analysis results against two pools. (a) and (b) show the RERs of the FWAP attacker and the colluding pool, respectively, with $\alpha = 0.2$ and $\eta = 0.2$, and $(\beta_1, \beta_2) = (0.2, 0.1)$.

By comparing Fig. 3 and Fig. 2, we can find that the overall shapes of these sub-figures do not change much but the quantities become larger. Fig. 3(a) shows that FWAP attacker can still gain more rewards than the PAW attacker in the two-pool scenario, and the gap between $\mathrm{RER}_a^{\mathrm{PM}}$ and $\mathrm{RER}_a^{\mathrm{P}}$ is not widened. Fig. 3(b) shows that the two-pool attack scenario would also indirectly increase the reward of the colluding pool since the attacker may distribute more infiltration mining power to victim pools.

In Table 2, we give typical examples regarding lower bounds of protection money ratio in a two-pool attack scenario. From the results, we can see that the number of victim pools has little impact on the lower-bound comparing to Table 1.

Table 2: Examples of protection money ratio μ. The values x(y) indicate the lower bound (x) and optimal value (y) of μ in a two-pool attack scenario, respectively.

(β_1, β_2)	c = 0	c = 0.25	c = 0.5	c = 0.75	c = 1
(0.1, 0.1)	0 (0.56)	0.23 (0.49)	0.44 (0.44)	0.56 (0.52)	0.58 (0.60)
(0.2, 0.1)	0 (0.54)	0.11 (0.49)	0.27 (0.44)	0.42 (0.40)	0.47 (0.50)
(0.3, 0.1)	0 (0.56)	0.09 (0.50)	0.15 (0.43)	0.24 (0.35)	0.36 (0.37)

6 PROTECTION MONEY PRICING

For an FWAP attack, the attacker and the colluding pool need to pre-negotiate a protection money ratio μ in advance. Although Eq. 13 defines the lower bound of μ, the attacker is very likely to ask more than that, e.g., a value close to 1. Although the colluding pool can always be more profitable for any $\mu \in (0, 1)$, it still wants to pay with its optimal μ'. Let ρ be the value of the lower bound of μ, and $\epsilon \in (0, 1)$ be a small constant that is used to guarantee the minimum colluding reward reserved for the colluding pool, e.g., $\epsilon = 0.01$. Note that the ϵ could be set as a public parameter. The problem then becomes how to share the rest fraction $1 - \mu - \epsilon$ of colluding reward that the attacker can get from the colluding pool. That is, we can first model μ as a pricing function $\mu(\psi) = \rho + \psi \cdot (1 - \rho - \epsilon)$, where $\psi \in (0, 1)$ is an uncertain coefficient for now. To determine ψ, we propose to use the network capability c, since it reflects the capability of attacker on winning in the forks. Eventually, we can set $\psi = c$, and re-write the pricing function as

$$\mu = \rho + c \cdot (1 - \rho - \epsilon), \tag{10}$$

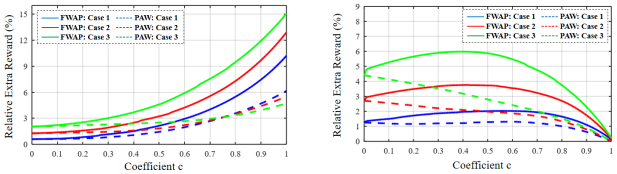
where ϵ satisfies that $\epsilon \in (0, 1)$ and $1 - \rho - \epsilon > 0$. We can also see a similar pattern from the quantitative analysis (i.e, Table 1) in Section 5. Since τ_1, α, and η are known to the attacker in advance, we could reduce the pricing problem to the one regarding determining the network capability c. The c can be computed following the approach in [24, §9], with a lower-bound $c = \alpha + \beta$. Since the c may change due to many reasons (e.g., Bitcoin network changing [21, 36] or Sybil nodes shifting [3]), the attacker and colluding pool can re-negotiate μ after an agreed period of time (e.g., every week). After the μ is settled, the colluding pool can pay the protection money whenever the attacker shows an FPoW that can be used to generate the fork to her block.

Quantitative Analysis and Simulation. We quantitatively analyze the RERs of an attacker and a colluding pool in the FWAP attack to show the effectiveness of our pricing function. We also compare FWAP with PAW under the new pricing function. In our analysis, we set $\alpha = 0.2$ for attacker and $\eta = 0.2$ for colluding pool. In single-pool scenario, we use different victim pool sizes $\beta \in \{0.1, 0.2, 0.3\}$. And we choose tree cases of computational power of victim pools (β_1, β_2), i.e., $(0.1, 0.1)$, $(0.2, 0.1)$, and $(0.3, 0.1)$, in the two-pool scenario. In addition, we set $\mathrm{c}_k^{(p_i)}/k$ (for $k \in \{1, 2\}$ when k infiltration miners find k distinct FPoWs in the two-pool scenario.

The RERs for an attacker based on the pricing function Eq. (10) are depicted in Fig. 4(a) and Fig. 5(a), respectively. Considering the two-pool scenario and the Case 3 ($\beta = 0.3$) in Fig. 4(a), the PAW attack gives the attacker an RER of 6.17% but she can gain a maximum RER of 20.26% in an FWAP attack. Therefore, the RER of an FWAP attacker is 3.8× in PAW (when $c = 1$ in the Case 3 of two-pool scenario). The RER of the attacker obtained based on Eq. (10) is near optimal comparing to the theoretical results shown in Section 5.

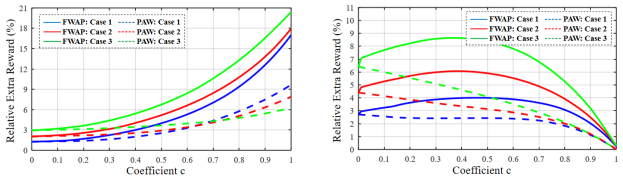
Fig. 4(b) and Fig. 5(b) depict the colluding pool's RER in the FWAP attack with a μ defined by Eq. (10). The results show that the FWAP attack can give the colluding pool a substantial extra reward compared to the PAW attack. E.g., the FWAP attack allows the colluding pool to earn up to 1.8× RER as PAW (when $c = 0.48$ in the Case 3 of two-pool scenario).

Moreover, we realize two Monte Carlo simulators with Matlab to verify the accuracy of our theoretical analysis of FWAP attacks in single-pool and two-pool scenarios, respectively. Besides, we run



(a) RER_a^{PM} and RER_a^{P} **(b)** RER_c^{PM} and RER_c^{P}

Figure 4: RER_a^{PM} and RER_c^{PM} against one pool with μ defined by Eq. (10). (a) and (b) show the RERs of FWAP attacker and the colluding pool, respectively, with $\alpha = 0.2$ and $\eta = 0.2$ and $\beta \in \{0, 1, 0.2, 0, 3\}$.



(a) RER_a^{PM} and RER_a^{P} **(b)** RER_c^{PM} and RER_c^{P}

Figure 5: RER_a^{PM} and RER_c^{PM} against two pools with μ defined by Eq. (10). (a) and (b) show the RERs of FWAP attacker and the colluding pool, respectively, with $\alpha = 0.2$ and $\eta = 0.2$, and $(\beta_1, \beta_2) \in \{(0.1, 0.1), (0.2, 0.1), (0.3, 0.1)\}$.

both simulators over 10^9 rounds. In our experiments, we consider the real-world mining pools taking an approximate computational power distribution from [1], which encompasses AntPool with computational power 21.27%, F2Pool with 16.24%, ViaBTC with 14.71%, BTC.com with 12.24%, Poolin with 13.52%, and SlushPool with 5%. We let AntPool play the attacker and let Binance be the colluding pool. The other pools remain in honest mining, which might be considered as potential victim pools.

Table 3: $\text{RER}_a^{\text{PM}}(\%)$ against one pool (1P) and two pools (2P). The value x(y) indicate RER_a^{PM} in the simulation (x) and theoretical (y) analysis, respectively.

	β	c = 0	c = 0.25	c = 0.5	c = 0.75	c = 1
1P	F2Pool	1.08 (1.07)	1.30 (1.29)	2.14(2.13)	4.19 (4.20)	8.37 (8.38)
	BTC.com	0.79(0.80)	1.00 (1.00)	1.86 (1.88)	3.93 (3.92)	8.09 (8.10)
	SlushPool	0.31 (0.33)	0.53 (0.53)	1.22 (1.21)	2.85(2.85)	6.44 (6.43)
	(β_1, β_2)	c = 0	c = 0.25	c = 0.5	c = 0.75	c = 1
2P	(F2Pool, BTC.com)	2.05 (2.05)	2.45 (2.44)	3.68(3.69)	6.36 (6.33)	11.85 (11.88)
	(Poolin, BTC.com)	1.82(1.83)	2.20 (2.20)	3.47 (3.45)	6.27 (6.28)	12.09 (12.10)
	(SlushPool, BTC.com)	1.15 (1.13)	1.52 (1.52)	2.82 (2.86)	5.75 (5.74)	11.91 (11.91)

The RERs shown in Table 3 are almost identical in simulation and theoretical analysis in both scenarios.

7 ANALYSIS OF MINER'S DELIMMA

In this section, we study an attack game between two pools p_1 and p_2 that execute the FWAP attacks against each other. We will show that the game has a unique Nash equilibrium, so we prove that FWAP can avoid the "miner's dilemma" [14]. That is, the game has a single winner, which is determined by the pool size (i.e., the larger pool can win the game). Similar to FAW and PAW, we here focus on a two-pool attack game for simplicity. We leave a detailed analysis of the generalized game with n pools for future work. To analyze the two-pool FWAP attack game, we define the winning condition as earning an extra reward.

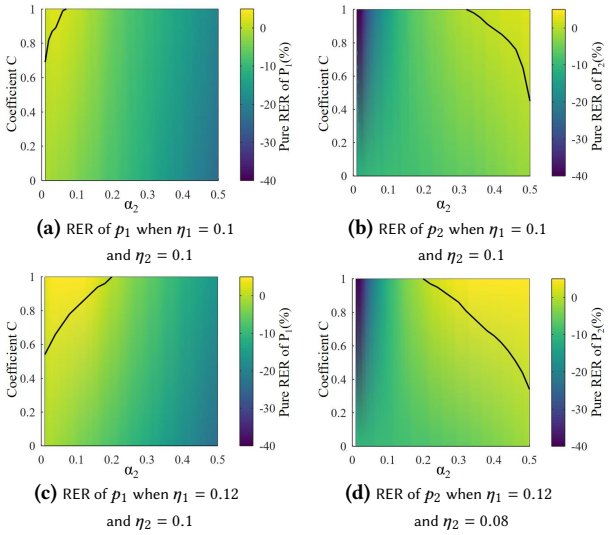
Theoretical Analysis. We consider a game involving two pools p_1 and p_2 with computational power α_i ($i \in \{1, 2\}$). The pool p_i will distribute infiltration mining power either $f_1^{(p_i)} = \tau_1^{(p_i)}\alpha_i$ or $f_2^{(p_i)} = \tau_2^{(p_i)}\alpha_i$ (after power adjusting) in its opponent pool $p_{\neg i}$, where $\neg i = 3 - i$ and $i \in \{1, 2\}$. We assume there are two colluding pools cp_1 and cp_2 which with computational power η_1 and η_2 and respectively provide the protection money to p_1 and p_2. We list the additional parameters to analyze a two-pool FWAP game as follows:

- $f_1^{(p_i)}$: Pool p_i's original infiltration mining power;
- $f_2^{(p_i)}$: Pool p_i's reallocated infiltration mining power after its infiltration mining finds an FPoW;
- $c_1^{(p_i)}$: Probability of the p_i's withheld FPoW is selected as the main chain in two-branch cases;
- $c_2^{(p_i)}$: Probability of the p_i's withheld FPoW is selected as the main chain in three-branch cases;

The pool p_i's reward $R_a^{(p_i)}$ in a two-pool FWAP game is calculated in Appendix F. Here we present the analysis result concerning the Nash equilibrium in the game via the following theorem.

Theorem 7.1. *The two-pool FWAP attack game has a unique Nash equilibrium and the equilibrium point is either a point satisfying $\nabla_{\mathbf{f}^{(p_1)}} R_a^{(p_1)} = 0$, $\nabla_{\mathbf{f}^{(p_2)}} R_a^{(p_2)} = 0$; or a point on the borderline which maximizes $R_a^{(p_1)}$ with $\mathbf{f}^{(p_1)}$ and $R_a^{(p_2)}$ with $\mathbf{f}^{(p_2)}$, where $\mathbf{f}^{(p_1)} = (f_1^{(p_1)}, f_2^{(p_1)})$ and $\mathbf{f}^{(p_2)} = (f_1^{(p_2)}, f_2^{(p_2)})$.*

The proof of this theorem is given in Appendix G.



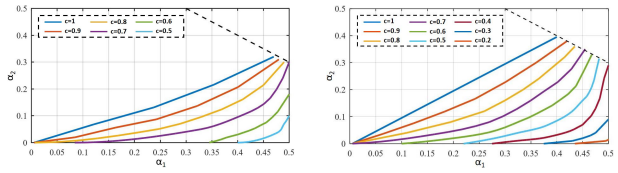
(a) RER of p_1 when $\eta_1 = 0.1$ and $\eta_2 = 0.1$ **(b)** RER of p_2 when $\eta_1 = 0.1$ and $\eta_2 = 0.1$

(c) RER of p_1 when $\eta_1 = 0.12$ and $\eta_2 = 0.1$ **(d)** RER of p_2 when $\eta_1 = 0.12$ and $\eta_2 = 0.08$

Figure 6: Quantitative analysis results of a two-pool FWAP game according to pool p_2's size α_2 and coefficient c ($c_1^{(p_1)} = c_1^{(p_2)}$, $c_2^{(p_1)} = c_2^{(p_2)} = c/2$) when $\alpha_1 = 0.2$.

Winning Conditions. In our analysis, the honest mining reward of p_i is represented by α_i, which means the total reward of p_i in a case that all of the miners in the Bitcoin system are honest (i.e., there is no attacker in the system). Hence, we say that there is a winner in the two-pool FWAP game if the reward of any pool p_i (for $i \in [2]$) is more than the pure reward of p_i. To quantitatively analyze the reward in a two-pool FWAP game under the Nash equilibrium point,

we first consider symmetric coefficients such that $c_1^{(p_1)} = c_1^{(p_2)} = c$, $c_2^{(p_1)} = c_2^{(p_2)} = c/2$. In particular, we consider two specific settings concerning the sizes of colluding pools, i.e., i) $\eta_1 = \eta_2 = 0.1$; and ii) $\eta_1 = 0.12$ and $\eta_2 = 0.08$. Fig. 6 shows the RERs of p_1 and p_2 with varying α_2 and c, a constant $\alpha_1 = 0.2$, and two colluding-pool settings, where the black line represents the $RER = 0$ (the same as honest mining). Each pool can earn the extra reward above the lines, so the "prisoner's dilemma" [14] does not hold. Meanwhile, we use a protection money ratio μ obtained from the pricing function in Eq. (10). It is not hard to see, the pool p_2 may earn more reward in the game if $\alpha_2 > 0.2$, and suffer a loss otherwise. By comparing the results from the two colluding-pool settings, we can see that the attacker with a bigger colluding pool is harder to defeat.



(a) $\eta_1 = 0.1$ and $\eta_2 = 0.1$ **(b)** $\eta_1 = 0.12$ and $\eta_2 = 0.08$

Figure 7: Winning conditions of pool p_1. The right side of each line depicts the winning range of the pool p_1 with the corresponding c. The dotted line shows the constraint such that $\alpha_1 + \alpha_2 = 0.8$.

In Fig. 7, we further demonstrate the winning conditions of pool p_1 for different coefficients c, under the above two colluding-pool settings, respectively. Each solid line in Fig. 7 represents a borderline at which the reward of the pool p_1 is identical to the honest mining (with the same parameters). Since FWAP involves colluding pools, the computational power of two attackers should satisfy the constraint $\alpha_1 + \alpha_2 \leq 1 - \eta_1 - \eta_2$. The area on the right side of each line represents the winning range of p_1 under the corresponding c. The winner in a two-pool FWAP game is determined by its pool size. Namely, the winning condition is related to the pool sizes α_1 and α_2, and the coefficient c. The results, as shown in the second colluding-pool setting, also imply that the pool p_1 with a bigger colluding pool is easier to win the game. In a nutshell, the larger pool can earn an extra reward depending on c, while the smaller pool will always suffer a loss despite c. In this sense, the "miner's dilemma" can be avoided in the two-pool FWAP game.

8 COUNTERMEASURES

Defending Infiltration Mining. Previous works on block withholding attacks (such as [16, 24]) introduce several countermeasures for reducing the infiltration miners' reward, which can also be applied to mitigate the FWAP attack (in which the infiltration mining still plays a fundamental role). Kwon et al. [24] propose to use a reward strategy that increases the relative value of FPoWs compared to PPoWs. They show that the infiltration miners' shares obtained from the victim pool would be much smaller if most of their contributions are PPoWs. This effectively reduces the incentives for infiltration mining. We can apply the same strategy to mitigate the FWAP attack, but we leave the detailed analysis to future work. In [16], Gao et al. propose to identify and expel infiltration miners by finding stale FPoWs, that is, to discard FPoWs submitted after the pool has received a new block from external miners for a pre-defined time threshold T_t (e.g., $\mathrm{T}_t = 6$ seconds [16]).

An Improved Method on Detecting Stale FPoWs. We first revisit the Gao et al.'s scheme on detecting stale FPoWs, they define a *submission rule* that an FPoW submitted by each miner is accepted as valid if and only if it contains a timestamp T_a within a range $[\max(\mathrm{T}_c - \mathrm{T}_t, 0), \mathrm{T}_c + \mathrm{T}_t]$, where T_t is a time threshold defined by the pool manager and T_c is current time. A submitted FPoW is considered as a stale FPoW when the pool manager receives a block from the others at time T_r, if T_r is not within the range $[\mathrm{T}_a - \mathrm{T}_t, \mathrm{T}_a + \mathrm{T}_t]$. However, we notice that the attacker might evade this kind of detection with a significant probability. Since Gao et al.'s scheme allows the attacker to specify the timestamp for mining an FPoW, so the attacker can always choose a future timestamp T_a such that T_a is close to the block generation time T_r of the others. Note that T_r might be predictable with a non-small successful probability following the approach in [11].

Here we introduce an improved scheme for detecting stale FPoWs to avoid the above timestamp manipulation issue. Our goal is to make the attacker have a negligible probability of evading the detection. Specifically, we let the pool manager refresh the Merkle root that in the block header every T_t seconds for the miners. In this sense, a consensus round with mining time length TL is divided into TL/T_t time slots. The concrete Merkle root of the i-th time slot is randomly generated and serves as a unique identifier for the corresponding time slot. Each new Merkle root will be released to miners at the end of the $i-1$-th time slot. Then the pool manager would only accept an FPoW at a time T_c if it contains a Merkle root that is associated with the time slot covering T_c. Since each Merkle root is assigned by the pool manager on the fly, the attackers cannot predicate it (with a non-negligible probability) and mine for a future time slot. So the attacker should submit an FPoW once it is found, otherwise the found FPoW will be treated as a stale FPoW. In a nutshell, our approach can effectively reduce the intentional fork caused by infiltration mining. If all mining pools apply our above patch and refuse to broadcast stale FPoWs, the colluding pool will not establish a protection racket with the attacker since no intentional forks will be created against her.

However, if a rational pool manager chooses to distribute stale blocks and just evict the miner submitting the corresponding stale FPoW, then the attacker can still exploit the victim pool by continuously sending infiltration miners to the pool with fresh addresses. We also discussed other countermeasures on detecting the colluding pool in Appendix I.

9 CONCLUSION

In this paper, we presented a new attack, called FWAP, that extends existing mining attacks with a strategy involved protection money. Our attack increases the rewards for both the attacker and the colluding pool. Although our work focuses on Bitcoin, the attack also works with other PoW-based coin systems, such as Litecoin and ZCash. We have formalized the attack and modeled the pricing function. However, questions such as finding the optimal pricing function are left as future work. In Appendix I, we give the discussion on the practicability of FWAP attack in the real world. Finally, we proposed a new countermeasure to mitigate the FWAP attack by introducing an improved method to detect the stale FPoWs and a phishing approach to prevent the colluding pool.

ACKNOWLEDGMENTS

We would like to thank our shepherd and anonymous reviewers for their invaluable comments and suggestions. This work is supported by the Natural Science Foundation of China (Grant No. 61872051), the Natural Science Foundation of Chongqing (Grant No. CSTB2022NSCQ-MSX0437), and the Fundamental Research Funds for the Central Universities (Grant No. SWU-KR22003). Junming Ke was partly supported by the 2022 Fund of XRP Ledger Trust Scholarship from University of Tartu. The work of Chao Yin is supppprted by China Scholarship Council and the Dutch Sectorplan.

REFERENCES

[1] 2021. Bitcoin mining pools. *http://history.btc126.com/pools/[Online; accessed 19-Aug-2021]* (2021).
[2] Maria Apostolaki, Aviv Zohar, and Laurent Vanbever. 2017. Hijacking Bitcoin: Routing Attacks on Cryptocurrencies. In *IEEE Symposium on Security and Privacy*. IEEE, 375–392.
[3] Moshe Babaioff, Shahar Dobzinski, Sigal Oren, and Aviv Zohar. 2012. On bitcoin and red balloons. In *EC*. ACM, 56–73.
[4] Samiran Bag, Sushmita Ruj, and Kouichi Sakurai. 2017. Bitcoin Block Withholding Attack: Analysis and Mitigation. *IEEE Trans. Inf. Forensics Secur.* 12, 8 (2017), 1967–1978.
[5] S. Bag, S. Ruj, and K. Sakurai. 2017. Bitcoin Block Withholding Attack: Analysis and Mitigation. *IEEE Trans. Inf. Forensics Secur.* 12, 8 (2017), 1967–1978.
[6] Lear Bahack. 2013. Theoretical Bitcoin Attacks with less than Half of the Computational Power (draft). *CoRR* abs/1312.7013 (2013).
[7] George Bissias and Brian Neil Levine. 2020. Bobtail: Improved Blockchain Security with Low-Variance Mining. In *NDSS*. The Internet Society.
[8] Bitcoin Project 2020. Bitcoin developer guide. https://bitcoin.org/en/developer-guide.
[9] bitcoin.it. 2020. Pooled Mining. https://en.bitcoin.it/wiki/Pooled_mining.
[10] Joseph Bonneau. 2016. Why Buy When You Can Rent? - Bribery Attacks on Bitcoin-Style Consensus. In *FC (Lecture Notes in Computer Science, Vol. 9604)*. Springer, 19–26.
[11] R. Bowden, H. P. Keeler, A. E. Krzesinski, and P. G. Taylor. 2020. Modeling and analysis of block arrival times in the Bitcoin blockchain. *Stochastic Models* 36, 4 (2020), 602–637. https://doi.org/10.1080/15326349.2020.1786404.
[12] Danny Bradbury. 2013. The problem with Bitcoin. *Computer Fraud & Security* 2013, 11 (2013), 5 – 8.
[13] Nicolas T. Courtois and Lear Bahack. 2014. On Subversive Miner Strategies and Block Withholding Attack in Bitcoin Digital Currency. *CoRR* abs/1402.1718 (2014).
[14] Ittay Eyal. 2015. The Miner's Dilemma. In *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 89–103.
[15] Ittay Eyal and Emin Gün Sirer. 2014. Majority Is Not Enough: Bitcoin Mining Is Vulnerable. In *FC*. Springer, 436–454.
[16] Shang Gao, Zecheng Li, Zhe Peng, and Bin Xiao. 2019. Power Adjusting and Bribery Racing: Novel Mining Attacks in the Bitcoin System. In *CCS*. ACM, 833–850.
[17] Peter Gazi, Aggelos Kiayias, and Alexander Russell. 2020. Tight Consistency Bounds for Bitcoin. In *CCS*. ACM, 819–838.
[18] Adem Efe Gencer, Soumya Basu, Ittay Eyal, Robbert van Renesse, and Emin Gün Sirer. 2018. Decentralization in Bitcoin and Ethereum Networks. In *Financial Cryptography (Lecture Notes in Computer Science, Vol. 10957)*. Springer, 439–457.
[19] Alireza Toroghi Haghighat and Mehdi Shajari. 2019. Block withholding game among bitcoin mining pools. *Future Generation Computer Systems* 97 (2019), 482–491.
[20] Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. 2015. Eclipse Attacks on Bitcoin's Peer-to-Peer Network. In *USENIX Security Symposium*. USENIX Association, 129–144.
[21] Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. 2015. Eclipse attacks on bitcoin's peer-to-peer network. In *Security*. USENIX, 129–144.
[22] Harry Kalodner, Malte Möser, Kevin Lee, Steven Goldfeder, Martin Plattner, Alishah Chator, and Arvind Narayanan. 2020. BlockSci: Design and applications of a blockchain analysis platform. In *Security*. USENIX, 2721–2738.
[23] Lucianna Kiffer, Rajmohan Rajaraman, and Abhi Shelat. 2018. A better method to analyze blockchain consistency. In *CCS*. 729–744.
[24] Yujin Kwon, Dohyun Kim, Yunmok Son, Eugene Y. Vasserman, and Yongdae Kim. 2017. Be Selfish and Avoid Dilemmas: Fork After Withholding (FAW) Attacks on Bitcoin. In *CCS*. ACM, 195–209.
[25] 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 *CSF*. IEEE, 397–411.
[26] Loi Luu, Yaron Velner, Jason Teutsch, and Prateek Saxena. 2017. SmartPool: Practical Decentralized Pooled Mining. In *USENIX Security Symposium*. USENIX Association, 1409–1426.
[27] Ralph C. Merkle. 1987. A Digital Signature Based on a Conventional Encryption Function. In *CRYPTO (Lecture Notes in Computer Science, Vol. 293)*. Springer, 369–378.
[28] Jelena V. Misic, Vojislav B. Misic, Xiaolin Chang, Saeideh Gholamrezazadeh Motlagh, and M. Zulfiker Ali. 2020. Modeling of Bitcoin's Blockchain Delivery Network. *IEEE Trans. Netw. Sci. Eng.* 7, 3 (2020), 1368–1381.
[29] Satoshi Nakamoto. 2019. *Bitcoin: A peer-to-peer electronic cash system.* Technical Report. Manubot.
[30] Kartik Nayak, Srijan Kumar, Andrew Miller, and Elaine Shi. 2016. Stubborn Mining: Generalizing Selfish Mining and Combining with an Eclipse Attack. In *EuroS&P*. IEEE, 305–320.
[31] J Ben Rosen. 1965. Existence and uniqueness of equilibrium points for concave n-person games. *Econometrica: Journal of the Econometric Society* (1965), 520–534.
[32] Meni Rosenfeld. 2011. Analysis of Bitcoin Pooled Mining Reward Systems. *CoRR* abs/1112.4980 (2011).
[33] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. 2016. Optimal Selfish Mining Strategies in Bitcoin. In *Financial Cryptography (Lecture Notes in Computer Science, Vol. 9603)*. Springer, 515–532.
[34] Clara Schneidewind, Ilya Grishchenko, Markus Scherer, and Matteo Maffei. 2020. eThor: Practical and Provably Sound Static Analysis of Ethereum Smart Contracts. In *CCS*. ACM, 621–640.
[35] Alin Tomescu and Srinivas Devadas. 2017. Catena: Efficient non-equivocation via bitcoin. In *S&P*. IEEE, 393–409.
[36] Muoi Tran, Inho Choi, Gi Jun Moon, Anh V. Vu, and Min Suk Kang. 2020. A Stealthier Partitioning Attack against Bitcoin Peer-to-Peer Network. In *S&P*. IEEE, 894–909.
[37] wizkid057. 2014. BWH attacks against Eligius. https://bitcointalk.org/?topic=441465.msg7282674.
[38] Karl Wüst, Loris Diana, Kari Kostiainen, Ghassan Karame, Sinisa Matetic, and Srdjan Capkun. 2019. Bitcontracts: Supporting Smart Contracts in Legacy Blockchains. Cryptology ePrint Archive, Paper 2019/857. https://doi.org/10.14722/ndss.2021.24294. https://eprint.iacr.org/2019/857.
[39] Mengya Zhang, Xiaokuan Zhang, Yinqian Zhang, and Zhiqiang Lin. 2020. TXSPECTOR: Uncovering Attacks in Ethereum from Transactions. In *Security*. USENIX, 2775–2792.

A PRELIMINARY ON POWER ADJUSTING

Power adjusting is a mining strategy proposed to optimize the reward of PAW attackers [16]. That is, the attacker dynamically adjusts the mining power between innocent and infiltration mining in a given round. As a result, the attacker can always improve her reward by allocating more power to the more profitable reward (either a whole block reward or a shared reward). In a single-pool attack scenario, the attacker will distribute τ_i (for $i \in \{1, 2\}$) fraction of her computational power for infiltration mining before (with τ_1) or after (with τ_2) power adjusting. The rest of $1 - \tau_i$ fraction of her computational power is used for innocent mining. The main goal of power adjusting is to determine the optimal portion of the attacker's computational power to maximize the expected reward in a given round. Specifically, in each round, the attacker pre-computes the optimal infiltration mining power τ_1 and τ_2 to maximize her the expected reward (e.g., defined by Eq. (1)). Meanwhile, the average infiltration mining power used in the calculation of the expected reward is computed following [16, Theorem 5.1]. So the attacker starts the attack with the infiltration mining power τ_1 and adjusts it to be τ_2 when her infiltration mining finds an FFoW before her innocent mining and the others. The above approach can be generalized to fit in the scenario when infiltrating multiple victim pools. Our new attack will adopt the identical power adjusting strategy of PAW, but leverage on the FFoWs found by infiltration mining to realize the Bitcoin protection racket.

B PROOF OF THEOREM 5.2

We show that the miner who executes the FWAP attack can get more rewards comparing to honest mining. Suppose the optimal τ for an BWH attack. When c = 0, $\tau_1 = \tau_2 = \tau$, and $\mu = 0$, the reward of the FWAP attacker $\mathrm{R}_{\mathrm{a}}^{\mathrm{PM}}(\tau_1, \tau_2)$ is equal to the reward R_a^{B} of an BWH attacker since the FWAP attacker cannot get reward from generating forks with $c = 0$ (that is, the infiltration miners do nothing but withhold their FPoWs as BWH attackers). Previous work by Luu et al. [25, Theorem IV-B.1] proved that an BWH attacker can always get more reward than the honest mining by choosing a proper τ, so an FWAP attacker with the above-assumed parameters can always get an extra reward as in the BWH attack regardless of the attacker's computing power.

Besides, the attacker will get more rewards than in BWH with non-zero coefficient $\mathrm{c} > 0$ and the protection money $\mathrm{R_m}$. That is, we have $\mathrm{R}_{\mathrm{a}}^{\mathrm{PM}}(\tau_1, \tau_2) > \mathrm{R}_a^{\mathrm{B}} + \mathrm{R_m}$. From Lemma 5.1, we can have that the colluding reward $\mathrm{R}_{\mathrm{cp}}^{\mathrm{Df}}(\tau_1, \tau_2)$ is always greater than zero, under the assumed optimal τ in the BWH attack (i.e., $\tau_1 = \tau_2 = \tau$). This results in the protection money such that $\mathrm{R_m} = \mu \cdot \mathrm{R}_{\mathrm{cp}}^{\mathrm{Df}}(\tau_1, \tau_2) > 0$ for $\mu > 0$. Furthermore, if we fix μ to be an arbitrary positive number, $\mathrm{R}_{\mathrm{a}}^{\mathrm{PM}}(\tau_1, \tau_2)$ is an increasing function of c. Therefore, the extra reward for the FWAP attacker is always lower bounded by that for the BWH attacker.

C PROOF OF THEOREM 5.3

Recall that if an PAW attacker's innocent mining fails to generate a block, but her infiltration mining wins the fork against the colluding pool, she is supposed to get a shared reward from the victim pool. However, an FWAP attacker gives up such a forking opportunity to provide the protection for the colluding pool, so the attacker would suffer a *reward loss* if she gets no protection money from the colluding pool. A pre-requisite of FWAP attack is that the colluding reward should be greater than the reward loss, so the attacker can gain more reward by taking the protection money from the colluding reward. We first show that this is always held when $\mathrm{c} > 0$. Since the infiltration miner has $\tau_1'\alpha \cdot \frac{\eta}{1-\tau_2'\alpha}$ probability to generate a fork against colluding pool, and has c probability to win in the fork, then the victim pool manager will share $\frac{\bar{\tau}'\alpha}{\beta+\bar{\tau}'\alpha}$ fraction of a block reward with the infiltration miner. By multiplying these two probabilities, we can have the reward loss

$$\mathrm{c} \cdot \tau_1'\alpha \cdot \frac{\eta}{1-\tau_2'\alpha} \cdot \frac{\bar{\tau}'\alpha}{\beta+\bar{\tau}'\alpha}. \tag{11}$$

For comparison, we still use the same infiltration mining power in both PAW and FWAP. With (τ_1', τ_2') in PAW, we have a colluding reward defined by the Eq. (2). By Lemma 5.1, we know that colluding reward is always greater than zero when $c > 0$, which implies the victim pool's reward (from generating forks against the colluding pool) becomes the colluding reward.

Now we can divide Eq. (11) by Eq. (2) to get $\frac{\bar{\tau}'\alpha}{\beta+\bar{\tau}'\alpha}$. Since the victim pool's computational power $\beta > 0$, the resultant quotient is always smaller than 1, i.e., $\frac{\bar{\tau}'\alpha}{\beta+\bar{\tau}'\alpha} < 1$. This implies that the forking reward is smaller than the colluding reward as well. Therefore, we can always find a range of protection money ratio μ that can meet the FWAP's pre-requisite. The protection money is μ fraction of colluding reward,

$$\mu \cdot \mathrm{c} \cdot \tau_1'\alpha \cdot \frac{\eta}{1-\tau_2'\alpha}. \tag{12}$$

The protection money shown by Eq. (12) should be greater than the reward loss defined by Eq. (11), i.e.,

$$\begin{aligned} & \mathrm{c} \cdot \tau_1'\alpha \cdot \frac{\eta}{1-\tau_2'\alpha} \cdot \frac{\bar{\tau}'\alpha}{\beta+\bar{\tau}'\alpha} < \mu \cdot \mathrm{c} \cdot \tau_1'\alpha \cdot \frac{\eta}{1-\tau_2'\alpha} \\ \iff & \frac{\bar{\tau}'\alpha}{\beta+\bar{\tau}'\alpha} < \mu. \end{aligned} \tag{13}$$

That is, with a μ given by Eq. (13), the FWAP attacker can always earn more rewards than PAW attackers since the protection money can be more than the reward from winning the fork.

D PROOF OF THEOREM 5.6

We first review the *FWAP attack pre-requisite* that the protection money obtained by Lemma 5.5 should be greater than the attacker's loss due to giving up the forking chance against the colluding pool. Analogously, to determine the lower bound of μ, we let $\tau_j^{(p_i)} = \tau_j^{(p_i)'}$ (for all $j \in [n]$), where the latter one is the optimal infiltration mining power in the PAW attack. The attacker's loss caused by protecting the colluding pool is

$$\sum_{i=1}^{n} \sum_{\mathbf{p}_i \in \mathcal{P}^i} \left(\sum_{k=1}^{i} \widehat{\mathrm{SH}_i^{(p_k)}} \eta \mathrm{c}_i^{(p_k)} \cdot \widehat{\mathrm{PO}_{\mathbf{p}_i}} \right).$$

And the colluding reward is $\sum_{i=1}^{n} \sum_{\mathbf{p}_i \in \mathcal{P}^i} \left(\sum_{k=1}^{i} \eta \mathrm{c}_i^{(p_k)} \widehat{\mathrm{PO}_{\mathbf{p}_i}} \right)$.

From the following inequality (representing the FWAP attack pre-requisite), we can obtain the lower bound of μ

$$\begin{aligned} & \mu \sum_{i=1}^{n} \sum_{\mathbf{p}_i \in \mathcal{P}^i} \left(\sum_{k=1}^{i} \eta \mathrm{c}_i^{(p_k)} \widehat{\mathrm{PO}_{\mathbf{p}_i}} \right) > \sum_{i=1}^{n} \sum_{\mathbf{p}_i \in \mathcal{P}^i} \left(\sum_{k=1}^{i} \widehat{\mathrm{SH}_i^{(p_k)}} \eta \mathrm{c}_i^{(p_k)} \widehat{\mathrm{PO}_{\mathbf{p}_i}} \right) \\ \iff & \mu > \frac{\sum_{i=1}^{n} \sum_{\mathbf{p}_i \in \mathcal{P}^i} \left\{ \sum_{k=1}^{i} \widehat{\mathrm{SH}_i^{(p_k)}} \mathrm{c}_i^{(p_k)} \widehat{\mathrm{PO}_{\mathbf{p}_i}} \right\}}{\sum_{i=1}^{n} \sum_{\mathbf{p}_i \in \mathcal{P}^i} \left\{ \sum_{k=1}^{i} \mathrm{c}_i^{(p_k)} \widehat{\mathrm{PO}_{\mathbf{p}_i}} \right\}}. \end{aligned}$$

E PROTECTION RACKET

In this section, we describe an end-to-end instantiation of the FWAP attack. This includes how the attacker establishes the protection racket with a pool that is willing to pay. The main procedures of the protection racket is also shown in Figure 8. In the following, we assume communication between the attacker and any other party is secure and does not reveal the attacker's true identity.

Initial Contact. The FWAP attacker contacts a potential colluding pool cp, advertising that it has the ability to generate forks against cp's blocks. If the pool responds positively, both the pool and the attacker move to the next phase. Otherwise, the attacker looks for another potential colluding pool.

Proving Forking Ability. The attacker needs to convince cp that it can fork. To do so, once it finds a valid FPoW through the infiltration miners, it withholds the FPoW. But when it sees that cp broadcasts a new block in the same round, it immediately submits the FPoW to the victim pool's manager and cp simultaneously. If cp sees that the block released by the victim pool contains the received FPoW, then it believes in the attacker's forking ability.

Agreement and Preparation. The FWAP attacker and cp then enters into an agreement on their collusion. We identify four conditions for the agreement as follows:

- **C1:** If the attacker finds a valid FPoW and cp also finds a block but the other miners have not found a block yet, the attacker can send the valid FPoW to cp to ask for the immediate payment.

- **C2:** If cp receives a valid FPoW which can be used to generate a fork against her, it immediately pays the PM to the attacker. We assume that the attacker has sufficient communication bandwidth and latency so that it can minimize the impact of waiting for PM.
- **C3:** If either cp fails to pay or the attacker sees that cp is forked by other blocks, the attacker will submit this FPoW to create a fork instead of demanding PM.
- **C4:** If cp is forked by any other blocks after paying the PM in this round, cp can get a refund through the attacker's *deposit* (discussed later).

We note that **C2**, **C3**, and **C4** ensure that the FWAP attack can downgrade to the PAW attack when necessary, so the attacker's profit is not harmed. Once cp and the attacker agree on all the above conditions, they can determine the PM ratio μ based on Eq. (10). For simplicity, we assume that a colluding pool will only enter into such agreement with one FWAP attacker. Also, this can reduce the risk of the disclosure of the colluding pool.

Attacker's Deposit. As both the attacker and the colluding pool can gain more reward (as analyzed in Section 5 and 5.2), they are motivated to keep performing the attack for a long period. However, a dishonest attacker may break the agreement at some point (e.g., when it wants to terminate the attack). Since the colluding pool is required to pay the PM immediately upon receiving the FPoW from the attacker, the latter can submit the FPoW to the victim pool after receiving the payment, which causes a loss to cp due to the fork and the paid money. Furthermore, if the attacker, having been paid, fails to protect the colluding pool cp from other forks, the colluding pool should be able to get a refund (**C4**).

Here we describe a deposit mechanism to protect the colluding pool's interests. Specifically, the attacker puts some money into a deposit account, and the colluding pool can claim the amount if the following *refund condition* is held:

- Two valid blocks are broadcast in a given round after the colluding pool has paid the PM in that round, in which one of the blocks is from the colluding pool.

The attacker can re-claim this deposit after a pre-determined duration has passed, during which refund condition is not triggered.

The FWAP attacker needs to protect itself from a dishonest colluding pool who claims the deposit without paying PM. That is, the colluding pool must provide a proof that it has paid the PM. To realize this, we leverage a hash based commitment scheme to generate payment tokens for creating both deposit and PM (PM) transactions. Paying the PM is done by opening the committed value. In the following, we describe a high-level idea of the deposit in conjunction with the payment procedure.

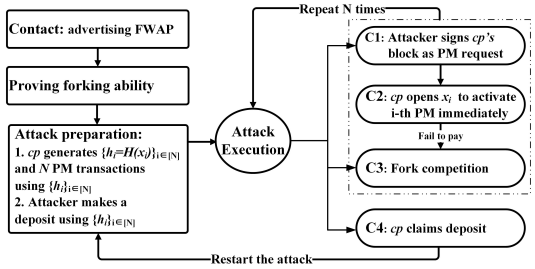


Figure 8: Main Procedures of Protection Racket.

Payment. Let $H : \{0,1\}^* \to \{0,1\}^\kappa$ be collision-resistant hash function, where κ is a security parameter. Let N be the number of potential protections for a PM ratio. To ensure a fast payment process during the FWAP attack, the attacker and the colluding pool can pre-create N deposit and PM transactions, respectively. First, the deposit as follows:

- The colluding pool generates N random values $\{x_i\}_{i\in[N]}$ and compute $h_i := H(x_i)$ for $i \in [N]$, where each x_i has κ bits.
- The set of hash values $\{h_i\}_{i\in[N]}$ is sent to the FWAP attacker.
- The attacker can realize the above refund condition by transforming it into the following logically equivalent sub-conditions: i) two valid blocks should have the same previous hash prev_hash of the last accepted block; ii) the i-th block of the colluding pool should contain a transaction involving h_i (for $i \in [N]$), and it has been digitally signed by the attacker using the secret key of one of her minions; iii) the colluding pool can provide the valid pre-image x_i of h_i.

Next, the colluding pool can create N PM transactions based on Bitcoin script [8] with constraints: i) the PM in each transaction will be returned to the colluding pool after a specified expiry time; ii) the i-th PM can be used by the attacker if x_i is presented.

For the subsequent executions of the FWAP attack, when the colluding pool cp generates the i-th block blk_i with a valid h_i, the attacker can send the signed blk_i and a valid FPoW (that can be used to create a fork against blk_i) to the colluding pool as the i-th payment request. Then, the colluding pool opens the x_i to the attacker to claim the i-th payment.

FWAP attackers and colluding pools can tailor smart contracts based on [38] (which supports the secure and efficient execution of generic smart contracts for legacy cryptocurrencies) to realize forking ability proving and payment. We leave the concrete implementation as future work.

F CALCULATION OF THE REWARD OF AN ATTACKER IN A TWO-POOL FWAP GAME

We calculate the reward $\text{R}_a^{(p_i)}$ of a pool p_i under the following cases (which are parameterized by p_i):

Case 1 (p_i): p_i's innocent mining finds an FPoW;
Case 2 (p_i): p_i's infiltration mining (in $p_{\neg i}$) finds an FPoW first and then her innocent mining finds another FPoW;
Case 3 (p_i): $p_{\neg i}$'s infiltration mining (in p_i) first finds an FPoW and then p_i's innocent mining finds another FPoW;
Case 4 (p_i): p_1's infiltration mining (in p_2), p_2's infiltration mining (in p_1), and p_i's innocent mining find three FPoWs in order;
Case 5 (p_i): p_2's infiltration mining (in p_1), p_1's infiltration mining, and p_i's innocent mining find three FPoWs in order.
Case 6 (p_i): $p_{\neg i}$'s infiltration mining (in p_i) finds an FPoW and then other miners (except for $cp_{\neg i}$) find an FPoW;
Case 7 (p_i): p_1's infiltration mining (in p_2), p_2's infiltration mining (in p_1), and other miners find three FPoWs in order;
Case 8 (p_i): p_2's infiltration mining (in p_1), p_1's infiltration mining (in p_2), and other miners find three FPoWs in order.

In a two-pool FWAP game, the reward $\text{R}_a^{(p_i)}$ of the pool p_i consists of the reward $\text{R}_{inno}^{(p_i)}$ of innocent mining, the reward $\text{R}_{fork}^{(p_i)}$ of creating a fork, the share $\text{R}_{share}^{(p_i)}$ obtained by infiltration mining, and protection money $\text{R}_{\text{m}}^{(p_i)}$ from its colluding pool. In the following, we specifically calculate the four reward sources of $\text{R}_a^{(p_i)}$, respectively.

First, the pool p_i can get rewards in Cases 1~5 (p_i) by innocent mining, which are: $\mathrm{R}^{(p_i)}_{\mathrm{C}_1(p_i)} = \alpha_i - f_1^{(p_i)}$; $\mathrm{R}^{(p_i)}_{\mathrm{C}_2(p_i)} = f_1^{(p_i)} \frac{\alpha_i - f_2^{(p_i)}}{1-f_2^{(p_i)}}$; $\mathrm{R}^{(p_i)}_{\mathrm{C}_3(p_i)} = f_1^{(p_{\neg i})} \frac{\alpha_i - f_1^{(p_i)}}{1-f_2^{(p_{\neg i})}}$; $\mathrm{R}^{(p_i)}_{\mathrm{C}_4(p_i)} = f_1^{(p_1)} \frac{f_1^{(p_2)}}{1-f_2^{(p_1)}} \frac{\alpha_i - f_2^{(p_i)}}{1-f_2^{(p_1)}-f_2^{(p_2)}}$; $\mathrm{R}^{(p_i)}_{\mathrm{C}_5(p_i)} = f_1^{(p_2)} \frac{f_1^{(p_1)}}{1-f_2^{(p_2)}} \frac{\alpha_i - f_2^{(p_i)}}{1-f_2^{(p_1)}-f_2^{(p_2)}}$. Then we can derive the reward of p_i from innocent mining as $\mathrm{R}^{(p_i)}_{inno} = \sum_{k=1}^{5} \mathrm{R}^{(p_i)}_{\mathrm{C}_k(p_i)}$.

Second, the pool p_i can get reward from causing a fork in Cases 6~8 (p_i) when any FPoW from p_i (including the one submitted by the infiltration miners of $p_{\neg i}$) in these cases is selected as the main chain. The reward of p_i in the these cases are $\mathrm{R}^{(p_i)}_{\mathrm{C}_6(p_i)} = \mathrm{c}_1^{(p_{\neg i})} f_1^{(p_{\neg i})} \frac{1-\alpha_1-\alpha_2-\eta_{\neg i}}{1-f_2^{(p_{\neg i})}}$, $\mathrm{R}^{(p_i)}_{\mathrm{C}_7(p_i)} = \mathrm{c}_2^{(p_{\neg i})} f_1^{(p_1)} \frac{f_1^{(p_2)}}{1-f_2^{(p_1)}} \frac{1-\alpha_1-\alpha_2}{1-f_2^{(p_1)}-f_2^{(p_2)}}$, and $\mathrm{R}^{(p_i)}_{\mathrm{C}_8(p_i)} = \mathrm{c}_2^{(p_{\neg i})} f_1^{(p_2)} \frac{f_1^{(p_1)}}{1-f_2^{(p_2)}} \frac{1-\alpha_1-\alpha_2}{1-f_2^{(p_1)}-f_2^{(p_2)}}$, respectively. So the reward of p_i from creating a fork is $\mathrm{R}^{(p_i)}_{fork} = \sum_{k=6}^{8} \mathrm{R}^{(p_i)}_{\mathrm{C}_k(p_i)}$.

Third, if the pool $p_{\neg i}$ gets a block reward, p_i can obtain shares of the reward depending on her average infiltration mining power in $p_{\neg i}$. We use $\overline{f}^{(p_i)}_{\mathrm{C}_k(p_j)}$ to generically denote the average infiltration mining power in the corresponding Case k (p_j) (where $\mathrm{C}_k(p_j)$ is an abbreviation for Case k (p_j)). The concrete value of $\overline{f}^{(p_i)}_{\mathrm{C}_k(p_j)}$ in each case is given in Table 4 (e.g., the value of $\overline{f}^{(p_1)}_{\mathrm{C}_7(p_1)}$ in Case 7 (p_1) is $\overline{f}^{(p_1)}_{1,2,2}$). The cells, which are associated with the cases involving power adjusting, are calculated as $\overline{f}^{(p_i)}_{1,2} = \frac{f_1^{(p_i)}+f_2^{(p_i)}-f_1^{(p_i)}f_2^{(p_i)}}{2-f_2^{(p_i)}}$, $\overline{f}^{(p_i)}_{1,1,2} = \frac{2f_1^{(p_i)}(1-f_2^{(p_1)}-f_2^{(p_2)})+f_2^{(p_i)}}{3-2(f_2^{(p_1)}+f_2^{(p_2)})}$, and $\overline{f}^{(p_i)}_{1,2,2} = \frac{(f_1^{(p_i)}+f_2^{(p_i)})(1-f_2^{(p_1)}-f_2^{(p_2)})+f_2^{(p_i)}}{3-2(f_2^{(p_1)}+f_2^{(p_2)})}$, respectively.

Table 4: Average infiltration power.

	$\overline{f}^{(p_1)}_{\mathrm{C}_k(p_1)}$	$\overline{f}^{(p_2)}_{\mathrm{C}_k(p_1)}$		$\overline{f}^{(p_1)}_{\mathrm{C}_k(p_2)}$	$\overline{f}^{(p_2)}_{\mathrm{C}_k(p_2)}$
Case 1 (p_1)	$f_1^{(p_1)}$	$f_1^{(p_2)}$	Case 1 (p_2)	$f_1^{(p_1)}$	$f_1^{(p_2)}$
Case 2 (p_1)	$\overline{f}^{(p_1)}_{1,2}$	$f_1^{(p_2)}$	Case 2 (p_2)	$f_1^{(p_1)}$	$\overline{f}^{(p_2)}_{1,2}$
Case 3 (p_1)	$f_1^{(p_1)}$	$\overline{f}^{(p_2)}_{1,2}$	Case 3 (p_2)	$\overline{f}^{(p_1)}_{1,2}$	$f_1^{(p_2)}$
Case 4 (p_1)	$\overline{f}^{(p_1)}_{1,2,2}$	$\overline{f}^{(p_2)}_{1,1,2}$	Case 4 (p_2)	$\overline{f}^{(p_1)}_{1,2,2}$	$\overline{f}^{(p_2)}_{1,1,2}$
Case 5 (p_1)	$\overline{f}^{(p_1)}_{1,1,2}$	$\overline{f}^{(p_2)}_{1,2,2}$	Case 5 (p_2)	$\overline{f}^{(p_1)}_{1,1,2}$	$\overline{f}^{(p_2)}_{1,2,2}$
Case 6 (p_1)	$f_1^{(p_1)}$	$\overline{f}^{(p_2)}_{1,2}$	Case 6 (p_2)	$\overline{f}^{(p_1)}_{1,2}$	$f_1^{(p_2)}$
Case 7 (p_1)	$\overline{f}^{(p_1)}_{1,2,2}$	$\overline{f}^{(p_2)}_{1,1,2}$	Case 7 (p_2)	$\overline{f}^{(p_1)}_{1,2,2}$	$\overline{f}^{(p_2)}_{1,1,2}$
Case 8 (p_1)	$\overline{f}^{(p_1)}_{1,1,2}$	$\overline{f}^{(p_2)}_{1,2,2}$	Case 8 (p_2)	$\overline{f}^{(p_1)}_{1,1,2}$	$\overline{f}^{(p_2)}_{1,2,2}$

When the pool p_i gets a block reward or a share (obtained by her infiltration mining) in each Case k (p_i), p_i shares such an income with her opponent pool $p_{\neg i}$ proportional to the infiltration mining power $\overline{f}^{(p_{\neg i})}_{\mathrm{C}_k(p_i)}$. In particular, the two attacking pools will share each other's prior shares back and forth in many rounds which we call *sharing rounds*. That is, in the n-th sharing round, the pool p_i can receive the shares that are determined by the shares of $p_{\neg i}$ obtained in her $n-1$-th sharing round. We let $\mathrm{R}^{(p_i)}_{S_n,\mathrm{C}_k(p_{\neg i})}$ be the shared reward of p_i in the n-th round under the Case k ($p_{\neg i}$). We can derive the following generic expressions of the shared reward in the Case k (p_i) for the n-th sharing round as $\mathrm{R}^{(p_i)}_{S_{n+2},\mathrm{C}_k(p_{\neg i})} = \mathrm{R}^{(p_i)}_{S_n,\mathrm{C}_k(p_{\neg i})} \frac{\overline{f}^{(p_{\neg i})}_{\mathrm{C}_k(p_{\neg i})}}{\alpha_i+\overline{f}^{(p_{\neg i})}_{\mathrm{C}_k(p_{\neg i})}} \frac{\overline{f}^{(p_i)}_{\mathrm{C}_k(p_{\neg i})}}{\alpha_2+\overline{f}^{(p_i)}_{\mathrm{C}_k(p_{\neg i})}}$ (for odd n), and $\mathrm{R}^{(p_i)}_{S_{n+2},\mathrm{C}_k(p_i)} = \mathrm{R}^{(p_i)}_{S_n,\mathrm{C}_k(p_i)} \frac{\overline{f}^{(p_{\neg i})}_{\mathrm{C}_k(p_i)}}{\alpha_i+\overline{f}^{(p_{\neg i})}_{\mathrm{C}_k(p_i)}} \frac{\overline{f}^{(p_i)}_{\mathrm{C}_k(p_i)}}{\alpha_2+\overline{f}^{(p_i)}_{\mathrm{C}_k(p_i)}}$ (for even n), where the shares in first two sharing rounds are $\mathrm{R}^{(p_i)}_{S_1,\mathrm{C}_k(p_{\neg i})} = \mathrm{R}^{(p_{\neg i})}_{\mathrm{C}_k(p_{\neg i})} \frac{\overline{f}^{(p_i)}_{\mathrm{C}_k(p_{\neg i})}}{\alpha_{\neg i}+\overline{f}^{(p_i)}_{\mathrm{C}_k(p_{\neg i})}}$ and $\mathrm{R}^{(p_i)}_{S_2,\mathrm{C}_k(p_i)} = \mathrm{R}^{(p_{\neg i})}_{S_1,\mathrm{C}_k(p_i)} \frac{\overline{f}^{(p_i)}_{\mathrm{C}_k(p_i)}}{\alpha_{\neg i}+\overline{f}^{(p_i)}_{\mathrm{C}_k(p_i)}}$, respectively. Note that the sharing rounds with the same parity have a common source, i.e., $\mathrm{R}^{(p_i)}_{\mathrm{C}_k(p_{\neg i})}$ in the odd sharing rounds and $\mathrm{R}^{(p_i)}_{\mathrm{C}_k(p_i)}$ in the even sharing rounds.

To simplify our following expressions, we introduce a function $F(a,b) = 1 + \frac{ab}{\alpha_i\alpha_{\neg i}+\alpha_{\neg i}b+a\alpha_i}$. When n approaches infinity, the pool shared reward of the pool p_i in the last sharing round can be negligible. The sums of shared reward in the even and the odd rounds can be therefore derived as $\sum_{\substack{n=2k+1 \\ k\in\mathbb{N}}} \mathrm{R}^{(p_i)}_{S_n,\mathrm{C}_k(p_{\neg i})} = \mathrm{R}^{(p_i)}_{S_1,\mathrm{C}_k(p_{\neg i})} \cdot F(\overline{f}^{(p_i)}_{\mathrm{C}_k(p_{\neg i})}, \overline{f}^{(p_{\neg i})}_{\mathrm{C}_k(p_{\neg i})})$ and $\sum_{\substack{n=2k \\ k\in\mathbb{N}^*}} \mathrm{R}^{(p_i)}_{S_n,\mathrm{C}_k(p_i)} = \mathrm{R}^{(p_i)}_{S_2,\mathrm{C}_k(p_i)} \cdot F(\overline{f}^{(p_i)}_{\mathrm{C}_k(p_i)}, \overline{f}^{(p_{\neg i})}_{\mathrm{C}_k(p_i)})$, respectively. The shared reward of p_i in a two-pool FWAP game is $\mathrm{R}^{(p_i)}_{share} = \sum_{k=1}^{8} \left\{ \sum_{\substack{n=2k+1 \\ k\in\mathbb{N}}} \mathrm{R}^{(p_i)}_{S_n,\mathrm{C}_k(p_{\neg i})} + \sum_{\substack{n=2k \\ k\in\mathbb{N}^*}} \mathrm{R}^{(p_i)}_{S_n,\mathrm{C}_k(p_i)} \right\}$.

Forth, we consider the PM of the colluding pool cp_i in the exchange of protection. Similarly, we start with the calculation of the lower bound of the PM ratio μ that cp_i can afford. We let $\mathrm{R}^{\mathrm{P}}_{cp_i}$ and $\mathrm{R}^{\mathrm{PM}}_{cp_i}$ be cp_i's reward in PAW and FWAP, respectively. And let $\mathrm{R}^{\mathrm{Df}}_{cp_i}$ denote the colluding reward of cp_i, and $\mathrm{R}^{(p_i)}_{loss}$ be the p_i's reward loss while offering the protection to cp_i without PM. Following the approach in Section 5, we can first evaluate the lower bound of PM ratio using the optimal infiltration power $(f_1^{(p_i)'}, f_2^{(p_i)'})$ in PAW, i.e., we set $(f_1^{(p_i)}, f_2^{(p_i)}) = (f_1^{(p_i)'}, f_2^{(p_i)'})$ in the following calculations. Since the PM should be larger than the attacker's reward loss because of withholding the FPoW to protect the colluding pool for free, so the lower bound of ρ_i should satisfy that $\rho_i > \frac{\mathrm{R}^{(p_i)}_{loss}}{\mathrm{R}^{\mathrm{PM}}_{cp_i} - \mathrm{R}^{\mathrm{P}}_{cp_i}}$.

The pool p_i will offer protection to her colluding pool cp_i if and only if her infiltration mining in $p_{\neg i}$ finds an FPoW before cp_i. For other cases, p_i distributes the FPoW found within her pool. Similar to the single-pool FWAP attack, the colluding reward of the colluding pool while enjoying the protection can be derived as $\mathrm{R}^{\mathrm{Df}}_{cp_i} = \mathrm{R}^{\mathrm{PM}}_{cp_i} - \mathrm{R}^{\mathrm{P}}_{cp_i} = \mathrm{c}_1^{(p_i)} f_1^{(p_i)} \frac{\eta_i}{1-f_2^{(p_i)}}$. The reward loss of p_i for protecting cp_i is $\mathrm{R}^{(p_i)}_{loss} = \mathrm{c}_1^{(p_i)} f_1^{(p_i)} \frac{\eta_i}{1-f_2^{(p_i)}} \frac{\overline{f}^{(p_i)}_{1,2}}{\alpha_{\neg i}+\overline{f}^{(p_i)}_{1,2}}$. By dividing the reward loss by the colluding reward, we can obtain the estimated PM ratio's lower bound $\rho_i = \frac{\mathrm{R}^{(p_i)}_{loss}}{\mathrm{R}^{\mathrm{PM}}_{cp_i} - \mathrm{R}^{\mathrm{P}}_{cp_i}} = \frac{\mathrm{R}^{(p_i)}_{loss}}{\mathrm{R}^{\mathrm{Df}}_{cp_i}}$. Based on Eq. (10), we can obtain an effective PM ratio of cp_i in a two-pool FWAP game $\mu_i = \rho_i + \mathrm{c}_1^{(p_i)} \cdot (1 - \rho_i - \epsilon)$.Once the the PM ratio μ_i is agreed, the pool p_i can re-calculate the optimal infiltration mining power $(f_1^{(p_i)}, f_2^{(p_i)})$ in FWAP that can result in an optimal colluding reward $(\mathrm{R}^{\mathrm{PM}}_{cp_i} - \mathrm{R}^{\mathrm{P}}_{cp_i})$ in a two-pool FWAP game. Eventually, the attacker p_i can receive a protection money $\mathrm{R}^{(p_i)}_{\mathrm{m}} = \mu_i \cdot \mathrm{R}^{\mathrm{Df}}_{cp_i}$.

Therefore, the reward $R_a^{(p_i)}$ of the pool p_i in a two-pool FWAP game can be expressed as $R_a^{(p_i)} = R_{inno}^{(p_i)} + R_{fork}^{(p_i)} + R_{share}^{(p_i)} + R_m^{(p_i)}$.

G PROOF OF THEOREM 7.1

Obviously, a two-pool FWAP game is a constrained 2-person game, in which the constrains for each pool p_i and her reward depends on the strategy (i.e., the distribution of infiltration mining power) of its opponent pool $p_{\neg i}$. In [31, Theorem 1], Rosen proves that an equilibrium point exists for every concave n-person game. Hence, to prove the existence of Nash equilibrium exists in a two-pool FWAP game, it suffices to show that the second partial derivatives of $R_a^{(p_1)}$ and $R_a^{(p_2)}$) for $\mathbf{f}^{(p_1)}$ and $\mathbf{f}^{(p_2)}$, respectively, are always negative (i.e., $\nabla_{\mathbf{f}^{(p_1)}}(\nabla_{\mathbf{f}^{(p_1)}} R_a^{(p_1)}) < 0$ and $\nabla_{\mathbf{f}^{(p_2)}}(\nabla_{\mathbf{f}^{(p_2)}} R_a^{(p_2)}) < 0$) under the following conditions:

$$\begin{aligned} &0 \le f_1^{(p_1)}, f_1^{(p_2)} \le \alpha_1 \le 1, 0 \le f_2^{(p_1)}, f_2^{(p_2)} \le \alpha_2 \le 1 \\ &\alpha_1 + \alpha_2 \le 1, 0 \le c_1^{(p_1)}, c_2^{(p_1)} \le 1, 0 \le c_2^{(p_1)} + c_2^{(p_2)} \le 1; \\ &0 \le \eta_1, \eta_2 \le 1, \eta_1 + \eta_2 \le 1, 0 \le \alpha_1 + \alpha_2 + \eta_1 + \eta_2 \le 1, 0 \le \mu \le 1. \end{aligned} \tag{14}$$

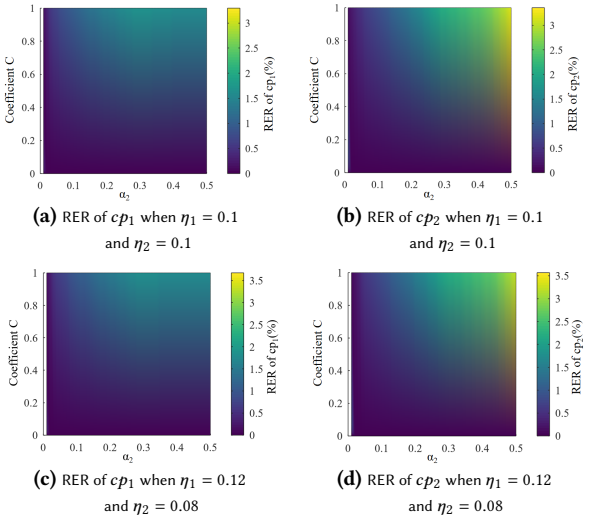
Moreover, we find the equilibrium point using the Best-response dynamics method. We initiate the game by setting the infiltration mining power of the two pools to be $(\mathbf{f}_0^{(p_1)}, \mathbf{f}_0^{(p_2)}) = ((0,0),(0,0))$. Next, the pools will alternately adjust their infiltration mining power to maximize $R_a^{(p_1)}$ and $R_a^{(p_2)}$. Namely, the pool p_1 first adjusts her infiltration power to be $\mathbf{f}_1^{(p_1)}$ to get the maximized $R_a^{(p_1)}$, and then the pool p_2 updates her infiltration power $\mathbf{f}_1^{(p_2)}$ to maximize $R_a^{(p_2)}$ based on $\mathbf{f}_1^{(p_1)}$. Both pools will repeat this procedure under the constraints in Eq. (14) to maximize their reward till $\mathbf{f}^{(p_1)}$ and $\mathbf{f}^{(p_2)}$ converge. In the k-th iteration, $\mathbf{f}_k^{(p_1)}$ and $\mathbf{f}_k^{(p_2)}$ can be derived as $\mathbf{f}_k^{(p_1)} = \arg\max_{\mathbf{f}^{(p_1)}} R_a^{(p_1)}(\mathbf{f}^{(p_1)}, \mathbf{f}_{k-1}^{(p_2)})$ and $\mathbf{f}_k^{(p_2)} = \arg\max_{\mathbf{f}^{(p_2)}} R_a^{(p_2)}(\mathbf{f}_k^{(p_1)}, \mathbf{f}^{(p_2)})$. We can find the Nash equilibrium point in the two-pool FWAP game when k approaches infinity and $\mathbf{f}^{(p_1)}$ and $\mathbf{f}^{(p_2)}$ converge. That is, the Nash equilibrium $(\mathbf{f}^{(p_1)}, \mathbf{f}^{(p_2)})$ is either a point satisfying $\nabla_{\mathbf{f}^{(p_1)}} R_a^{(p_1)} = 0$ and $\nabla_{\mathbf{f}^{(p_1)}} R_a^{(p_1)} = 0$, or a point on a borderline in Eq. (14) which maximizes $(R_a^{(p_1)}, R_a^{(p_2)})$ with $(\mathbf{f}^{(p_1)}, \mathbf{f}^{(p_2)})$.

H COLLUDING POOL IN GAME

Fig. 9 shows colluding pools cp_1 and cp_2's relative extra rewards in a two-pool FWAP game in the comparison with their rewards in a two-pool PAW game (with the similar setting in Fig. 6). That is, colluding pools' rewards are always more than that in the PAW attack game, because as in other FWAP attack scenarios, p_1 and p_2 will try to increase PM by adjusting infiltration mining power and increasing colluding pools' reward. Hence, this further confirms that colluding pools would join the FWAP attack.

I MISCELLANEOUS

Practicability Discussion. For the initial contact phase, the attackers first wrap the FWAP and advertise it as any other investment advertisements (e.g., via email, short message, or leaflet) to the potential colluding pool. Meanwhile, the FWAP attacker may try to create some intentional forks against the potential colluding pool



(a) RER of cp_1 when $\eta_1 = 0.1$ and $\eta_2 = 0.1$

(b) RER of cp_2 when $\eta_1 = 0.1$ and $\eta_2 = 0.1$

(c) RER of cp_1 when $\eta_1 = 0.12$ and $\eta_2 = 0.08$

(d) RER of cp_2 when $\eta_1 = 0.12$ and $\eta_2 = 0.08$

Figure 9: Quantitative analysis results for colluding pools in a two-pool FWAP game according to pool p_2's size α_2 and coefficient c ($c_1^{(p_1)} = c_1^{(p_2)}$, $c_2^{(p_1)} = c_2^{(p_2)} = c/2$) when $\alpha_1 = 0.2$.

to raise its attention. A rational pool manager (for gaining higher rewards) may choose to join the FWAP attack, no matter what its pool size is. Our result of Theorem 5.4 shows that a colluding pool can always gain more rewards in FWAP than in honest mining and in PAW, in spite of the size of the colluding pool. That is, a big colluding pool may also lose the fork competition with some probability, and then suffer a loss. Meanwhile, it might be impossible that all honest pools can resist such a temptation of the extra rewards brought by FWAP. For executing the FWAP attack, we do not require parties in FWAP to be near each other but need them to create high-speed connection channels during the attack (to ensure a fast FPoW exchange). Since the attacker and the colluding pool will first establish a secret pact before trading subsequent forking FPoWs, so they can pre-establish some kind of fast one-to-one communication channel (such as AT&T Dedicated Internet) to reduce latency. In addition, our tailored protection racket in Appendix E can ensure the interests of the colluding pools in any protection failures. In a nutshell, the whole life cycle of FWAP can be practical.

Other Countermeasures. As a colluding pool is complicit in the FWAP attack, it is important to detect and punish such a pool. Suppose the potential colluding pool uses her public contact methods to negotiate with the attacker. Honest miners can also masquerade as an attacker to test whether a pool is willing to become a colluding pool. However, this approach may not work if the negotiation takes place out-of-band, since one may not know all the contact methods used by the attacker and the colluding pool. Also, it is an open question to punish a colluding pool (without modifying the Bitcoin protocol) once it is identified.

Finally, the colluding pool can be identified based on the condition of the deposit if the method used to realize the deposit is known. This approach works because the spending condition of the deposit records the event that the colluding pool's block is forked and the protection money is paid. This is a current limitation of our work. However, we note that the attacker can obfuscate the deposit mechanism to avoid being detected. We leave the design of such deposit mechanisms as future work.