Countering Block Withholding Attack Efficiently

Suhyeon Lee[1,2] and Seungjoo Kim[1]

[1]CIST(Center for Information Security Technologies),
Korea University
[1]{*orion-alpha, skim71*}*@korea.ac.kr*
[2]Agency for Defense Development
[2]*korea@add.re.kr*

October 2018

Abstract

Bitcoin, well-known cryptocurrency, selected Poof-of-Work (PoW) for its security. PoW mechanism incentivizes participants and deters attacks on the network. Bitcoin seems to have operated the stable distributed network with PoW until now. Researchers found however some vulnerabilities in PoW such as selfish mining, block withholding attack, and so on. Especially, after Rosenfeld suggested block withholding attack and Eyal made this attack practical, many variants and countermeasures have been proposed. However, most countermeasures were accompanied by changes in the mining algorithm to make the attack impossible, which lowered the practical adaptability. In this paper, we propose a countermeasure to prevent block withholding attack effectively. Mining pools can adapt our method without changing their mining environment.

1 Introduction

In Bitcoin [2] and Ethereum [10], PoW is adapted to make its system secure, especially against Sybil attack. In PoW, people make mining pools and join in mining pools to mine blocks efficiently and share rewards. In Bitcoin protocol, mining difficulty is automatically set by calculating interval time of block generation. As a result of control of difficulty, bitcoin network generates a block every 15 minutes, and a miner who generates the block gets a reward. To get this reward efficiently, miners cooperate with each other. We call it a mining pool.

Researchers have shown several vulnerabilities in PoW. At first, selfish mining makes incentive incompatible with the blockchain reward design. If a miner has enough power, he can keep a block he mined and mining the next block in secret. By releasing more than one blocks when other miners generate a block,

he can make other miners waste their power. Hence, he earns more reward than he mines honestly. Furthermore, Eyal[16] proved that his selfish mining is useful even if a miner has over 25% of total mining power.

Second, block withholding attack is related to competition between mining pools. Mining Pools compete with each other to get Bitcoin reward. The concept 'difficulty', numerical value, in Bitcoin protocol means that only blocks which have a hash value less than difficulty are credited as legal blocks. Because, in most time, miners cannot find legal blocks(full Proof-of-Work; fPoW), they can prove they are mining with submitting partial blocks(partial Proof-of-Work; pPoW) that have a bigger hash value than the difficulty in Bitcoin protocol. Eyal[14] proposed the effective algorithm to implement block withholding attack. In his paper, one mining pool can attack other mining pools to get more reward than its mining power by submitting only pPoW but keeping fPoW in other mining pools.

Contributions Summarizing this paper, our contributions are as follows:

- *Method to detect block withholding attack.* If there is no bad intent, there is no reason to infiltrate and mine in other pools. Thus, for preventing block withholding attack, our method tries to detect the infiltration of mining pool first. Existing research papers do not treat methods to detect the attack or refer that it is hard to detect the attack [17]. If a pool is under block withholding attack, the pool can check whether other pools are attacking it by infiltration to other pools. Because block withholding attack shares task from the victim pool to miners, it is inevitable to expose where attacks.

- *Compatible method to counter attack.* Our method consists of two phases to detect and punish the attack. The punishment phase is the process of eliminating the damage to the attack by reducing the stake taken by the attacker. Our method does not need to alter the algorithm of mining, and it can be applied not only to Bitcoin but also to other cryptocurrencies using PoW. Furthermore, our countermeasure works well against other variants of block withholding attack such as FAW attack [17].

2 Preliminary

2.1 Proof-of-Work in Bitcoin

Bitcoin uses Proof-of-work(PoW) for its security property. PoW is a sort of data which verifies resource, for example, time-consuming work spent on data. It needs to be verified much easily by others. On the contrary, PoW is a mathematical puzzle challenging to produce. Blockchain network gives revenue(generally, a type of coin) to miners who find PoW to motivate them to generate blocks. In the blockchain system uses PoW, to profit specific miner, the block should depend on the miner's unique value.

In Bitcoin, its difficulty is a specific numeric value. It means the difficulty of this mathematical puzzle. Miners who try to generate a block should find nonce which makes a hash value of a block less than the difficulty in current Bitcoin protocol. In Figure 1, miners try to find a nonce value that the hash value of 6 header values(Version, Time, Bits, Nonce, hashPrevBlock, hashMerkleRoot) is less than difficulty. This random process to find nonce is very more exhausting work when the difficulty is smaller. The hash value is calculated by double iteration of the SHA256 hash algorithm.

To be concrete, Bitcoin block structure [3] is as Figure 1. Take note of the values, 'merkleRootHash' and 'coinbase.' The value 'merkleRootHash' depends on all transactions in the block. The first transaction in the block must be 'coinbase' transaction. 'Coinbase' transaction is a special transaction which generates bitcoin, that is, this transaction creates a certain amount of Bitcoin. As it is a creation, this transaction does not have a sender. 'Coinbase' transaction is a reward for who creates a block in PoW. Miner should make a block include 'coinbase' transaction of him as we referred. It is directly related to miners' reward. Even, some miners record their signature in arbitrary data space in 'coinbase' transaction [4]. So, 'coinbase' transaction is the basis for identifying who has mined a block.

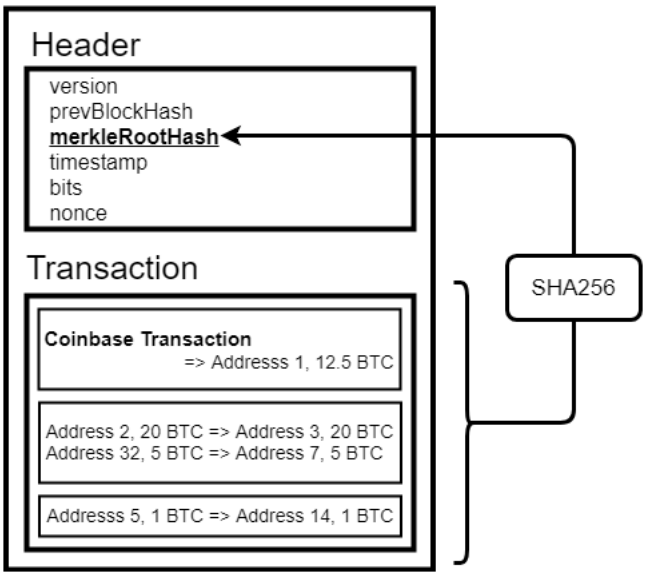

Figure 1: Bitcoin block structure

2.2 Mining Pool

Since a PoW requires a very big hashrate, which individual miners rarely find a block, a pooled mining is the dominant approach to generate a block [6]. In the mining pool, multiple client miners contribute to generate a block and share their reward as much as they contribute. This method makes the reward of block generation spread to multiple miners. Also, miners can reduce their risk of absent mining.

As we mentioned above, it needs high hashrate to mine a block. Therefore, there is a possibility that small miners cannot find any block in whole time. To prove that they are working, they submit pPoW(partial PoW) to the mining pool. pPoW is a block that meets the smaller value than the protocol specified in the current protocol. fPoW is a valid block that meets protocol difficulty. Even though pPoW does not help in obtaining Bitcoin reward for mining pools, it is useful as a basis for sharing the total rewards within a mining pool. Based on pPoW, each mining pools have a slightly different sharing algorithm for each mining pool in detail [5, 12].

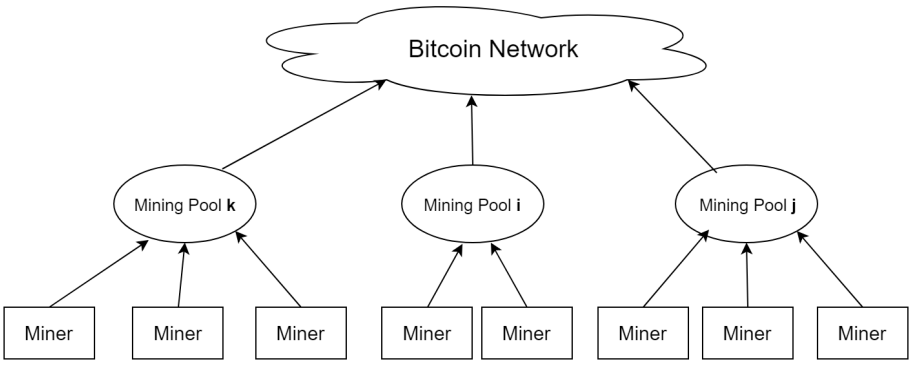

Figure 2: Bitcoin Mining Pool

In this paper, we treat a mining pool model which has a central manager. Because it is easy to manage, most of the mining pools use central management to share their revenue [5]. On the other hand, *P2P Pool* [11] is working with decentralized structure. *P2P Pool* needs its own blockchain. This blockchain helps to check and share their revenue. Even though its structure is different with centralized mining pools, this design is also vulnerable to block withholding attack. *Smart Pool* [19] is another design of decentralized mining pool. It is not vulnerable to block withholding attack because it assumes only one pool in the network. This design however is not adaptable in the Bitcoin network as it does not use *Smart Contract*.

3 Related Works

3.1 Block Withholding Attack

Rosenfeld [20] showed the classical forms of block withholding attack. Two forms are about block withholding by a miner in a pool. The first form is *Sabotage*. An Attacker in a mining pool withholds blocks and can harm to his pool. However, it is not profitable to attackers. The second form is *Lie in wait*. An attacker postponed submitting blocks to increase his reward. Contrary to *Sabotage*, it is profitable to the attacker.

In [14], Rosenfeld's idea is more evolved. Unlike [20], [14] developed it as an attack between a mining pool and a mining pool rather than an individual. As a manager of the mining pool share reward by pPoW share, miners can cheat their contribution. The pool can get the reward of block generation on by publication of fPoW. If miners do not submit fPoW on purpose, they can get share though they do not contribute to the pool in real. To implement this attack, the manager of the block withholding pool works as a proxy to infiltration miners. Concrete algorithm of the attack is 3 in Section 4. In this paper, we suggest a countermeasure against this kind of block withholding attack.

In block withholding attack, the reward of the attacker is as follows when we assume the total mining power of the network is 1.

α = computational power of the block withholding mining pool A
β = computational power of the victim mining pool B
τ = a proportion of the attacker pool A's infiltration mining power

Theorem 3.1. *The block withholding attacker pool's reward*

$$R_\beta(\tau) = \frac{(1-\tau)\alpha}{1-\tau\alpha} + \frac{\beta}{1-\tau\alpha}\frac{\tau\alpha}{\beta+\tau\alpha}$$

The reward is maximized when $\boldsymbol{\tau}$ *is*

$$\frac{\beta - \alpha\beta - \sqrt{\beta^2 - \alpha\beta^2 - \alpha\beta^3}}{-\alpha + \alpha^2 + \alpha\beta}$$

Kwon [17] showed that the attacker can earn more revenue if he releases fPoW of infiltration in specific situations. When the infiltration miners have already mined fPoW, and a third party unrelated to the attack publishes a valid block, the attacker can invoke a fork situation by submitting the pending fPoW. Then the attacker can get the reward of publishing a block conditional in competition with the valid block of others. This method does not increase the total release of valid blocks in the long chain so that it does not affect to total mining power.

3.2 Countermeasures

So far, researchers have suggested several countermeasures. Rosenfeld [20] and Eyal [15] proposed the first countermeasures to prevent the block withholding attack. Their methods are based on the idea that the attack is possible because miners can distinguish between pPoW and fPoW. Both of two methods commonly change PoW into two steps of PoW. So it is called as 'two-phase PoW'. At first, Rosenfeld[20] modified the original PoW process with the secret value so that the miners might not check whether the found block was a valid fPoW or not. This method needs blocks to have three additional fields as follows.

- *SecretSeed* : The secret value of the mining pool manager
- *ExtraHash* : *SHA256(SecretSeed)*
- *SecretHash* : The secret difficulty value of the mining pool manager

ExtraHash is the hash value of *SecretSeed*. *SecretHash* is the hash value of the concatenation of the block hash and *SecretSeed*. The pool manager only provides *ExtraHash* to miners and require the hash value of blocks less then *SecretSeed*. Instead of requiring that the block hash is less than specific difficulty, it is required that the block hash is less than difficulty and that *SecretHash* is less than some difficulty. Miners submit PoW without knowing *SecretSeed*. They cannot know their block is valid. The pool operator calculates SecretHash and checks if blocks submitted by miners are valid.

Second, another two-phase PoW by Eyal and Sirer [15] proceeds according to the following process.

- The first condition for the block to be valid, the double hash of the header *SHA256(SHA256(header))* is smaller than a difficulty parameter X.
- The second condition for the block to be valid, the header is signed with the coinbase transaction's private key, and the hash *SHA256(SIG(header, privkey))* of that signature is smaller than a second difficulty parameter Y.

Different from Rosenfeld's method, it needs the private key and its signature to divide the PoW process into the two-phase. Miners submit PoW without knowing blocks satisfy the second condition. They cannot know it because they do not know the coinbase transaction's private key. A similar intention, but using different technique is Bag[13]. He proposed a scheme that a mining pool has own secret *commitment* value to prevent miners to distinguish pPoW and fPoW. The most defection of their methods is that the original PoW process should be modified and the common ASIC hardware which miners are using cannot be applicable anymore.

In [18], Luu analyzed the changing payoff scheme of mining pools. He suggested that the pool gives a direct reward to the miners who submit fPoW. His

method however is not proper for small miners who rarely find fPoW. Eyal[14] suggested *pool fees* to make pools less attractive to block withholding attack. Fees add a friction element to the flow of revenue among infiltrated and infiltrating pools. However, this method makes pools not flexible for miners since miners do not want to pay fees of course.

3.3 Necessary Conditions for BWH Countermeasure

Countermeasures against the attack should be practical. Luu[18] discussed seven desired properties to review countermeasures. Kwon[17] also discussed drawbacks of existing countermeasures. Through the comprehensive analysis of [18] and [17], we will suggest below three necessary conditions for the countermeasure against block withholding attack. Also, we use these criteria for comparing all the proposed methods including ours at Section 6.2.

- **No Loss**: The countermeasure is a defense method against attacks and action to reduce loss due to attacks. It should not do any damage to the revenue of honest pools or honest miners as well as the pool who uses the method.
- **Compatibility**: A countermeasure should be adaptable in existing blockchain environment. This condition can be satisfied by minimizing the change of the entire blockchain mechanism. It means not only about the protocol but also about hardware compatibility. We conjecture a lot of big Bitcoin miners use ASIC hardware as ASIC structure is very efficient in specific calculation [9]. So a countermeasure should be adaptable in existing ASIC miners. If a method does not have it, big Bitcoin miners who have a big amount of ASIC mining machine will not adopt this method.
- **Fairness**: A pool should treat miners only by their contribution, not by their scale or other perspectives when the pool adopts a method. For example, if a pool can give a better share to miners who generate more valid blocks(fPoW), this is a disadvantage for small miners who rarely produce a valid block because of their mining power. If a method does not have a fair policy, some miners do not want to join the pool because of unfair treatment.

4 Model of Mining Pool

We modeled the elements of the mining pool by referring to the model of Eyal [14]. The format and some of the contents of the algorithms have been modified to make it easier to understand.

4.1 Assumption

We assumed two items in our model.

1. Public Pools: At least two pools exist in the network. All mining pools are public pools that every miner can join and leave to any pools freely. In real, some mining pools can be private. Even public pools can have their royal miners who do not leave.

2. Task with *Coinbase* Transaction: The mining pool sends PoW task containing *coinbase* to the miners. To solve the PoW problem, at least the information we need to know is the header value. Even if a miner does not know the transaction information, he can solve the PoW task if he knows only the *merkleRootHash* value. It is assumed here that the transaction information is also passed to miners to clarify the algorithm.

4.2 Elements of Mining Pool

4.2.1 Honest Miner

Algorithm 1 shows how **Honest Miner** M_i works. Miners get a new task from their mining pool. They work for finding PoW which is less then difficulty of the mining pool. They send both pPoW and fPoW if they find. They get revenue from the mining pool based on their share of PoW.

Algorithm 1 Honest Miner M_i

1: **procedure** MINING ▷ start mining
2: $tast \leftarrow newTask(w)$
3: $(pPoW, fPoW) \leftarrow work(task)$
4: $send(i, (pPoW, fPoW))$
5: $revenue \leftarrow revenue + recv(i)$

4.2.2 Honest Pool

Our pool model is the simple model which has a centralized pool manager. Algorithm 2 shows how **Honest Pool** P_i operates with miners. It has the list of miner(*workers*) and the list of their amount pPoW share(*jotRate*). The manager of P_i gives a task to miners. When miners submit their PoW, the manager checks validity and records their amount of PoW. The manager publishes all fPoW to the Bitcoin network and gets a reward from the network. He shares this reward based on each miner's job list(*jobRate*).

4.3 Elements of Block Withholding Pool

4.3.1 Honest Miner

In [14], they modeled an honest miner and a block withholding miner respectively. To perform the block withholding attack proposed initially by Rosenfeld [20], we need a model of the block withholding miner. However, as we mentioned in section 3.1, this paper deals with the block withholding attack between pools

Algorithm 2 Honest Pool P_i

```
1: variables
2:     List workers                                  ▷ registered miners
3:     List jobRate                                  ▷ amount of submitted PPoW
4:     Float revenue                                 ▷ total reward of mining pool
5: end variables
6:
7: procedure MANAGE MINING
8:     revenue ← 0
9:     for each a ∈ Workers do
10:        task(a) ← newTask(i)
11:        send(a, tasks(a))
12:    for each a ∈ Workers do                       ▷ get PoW from workers
13:        (pPoW, fPoW) ← recv(a)
14:        revenue ← revenue + publish(tasks(a), fPoW)
15:        jobRate(a) ← pPoW
16:    totalPoW ← each jobRate(a)
17:    for each a ∈ Workers do                       ▷ do payment
18:        pay(a, revenue × jobRate(a)/totalPoW)
```

proposed by Eyal [14]. We assume miners are honest in the block withholding pool as they are in the honest pool.

4.3.2 Block Withholding Pool

Figure 3 shows this attack in visual way. The mining pool P_k attacks mining pool P_i. The manager of mining pool P_k joins to Mining Pool P_i as a miner of miners. Mining pool P_i gives PoW task to mining pool P_k. Then, the manager of mining pool P_k tosses to its miner who is in the gray box in Figure 2. These miners work as infiltration miners for block withholding attack though they work honestly. Even though they normally work for a given task, the manager of mining pool P_k doesn't submit fPoW to mining pool P_i.

Algorithm 3 shows the block withholding attack controlled by a mining pool manager.

5 Our Method against Block Withholding Attack

Our proposal consists of two phases: detecting an infiltration and punishing an attacker.

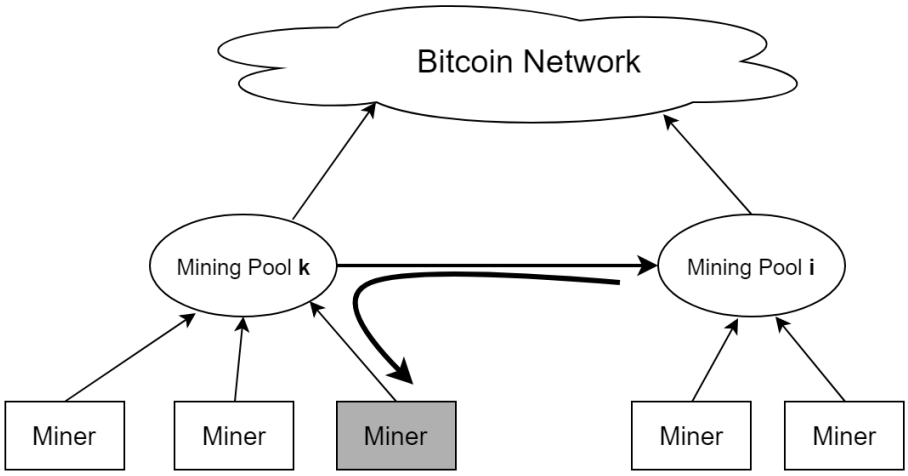

Figure 3: Block Withholding Attack

5.1 Phase 1: Detection of Infiltration

We took note of the structure of block withholding attack. The manager of the block withholding pool joins to the victim pool as a miner or miners. The miners under the attacker pool work for infiltration. In Figure 4, the miner in the grey box work as infiltration power. The role of the attack manager is a sort of proxy of PoW task. Then, if the victim pool joins to the block withholding pool, in the same way, the victim pool can investigate what PoW task the attacker pool send to miners. In figure 4, the miner with dotted line joins the attacker pool and works as a sensor of the attack. Since this join is for detection, this infiltration mining power honestly works. That is, it sends both pPoW and fPoW. We can call this infiltration mining power as sensor mining power. Consequently, two pools' task is shared with each other. By investigating PoW task in the attacker pool, the detective pool can find its task in the middle of the attacker's task. In Figure 4, PoW task is circulating in mining pools which are infiltrating to each other.

We can detect the infiltration in a way similar to a block withholding attack. A detective pool puts sensor miners in other pools. If the detective pool finds task including detective pool's *coinbase* transaction in infiltration miners, it can determine that the mining pool in which the task is found is infiltrating to the pool which uses the sensor. Because the infiltration creates a severe burden on the attacker, if there is no evil intent, there is no reason to infiltrate and mine in other pools. More specifically, the block withholding attack can be implemented by the proxy mining structure as Alogirhtm 3 and Figure 2. This structure is accompanied with overhead to toss PoW task, and also the infiltrator(i.e., manager) must give up the ancillary fee(Transaction fee, PPS fee, etc.)[5] that he can obtain by honest mining in his pool. Thus we choose to

Algorithm 3 Block Withholding Pool P_k against Mining Pool P_i

variables
  List $workers$ ▷ registered miners
  List $isInfilt$ ▷ registered miners for infiltration mining
  List $infiltID$ ▷ registered miners' ID in target pool
  List $jobRate$ ▷ amount of submitted PPoW
  Float $revenue$ ▷ total reward of mining pool
  Float $infiltRevenue$ ▷ total reward of target mining pool
end variables

procedure MANAGE MINING
  for each $a \in \mathcal{W}orkers$ **do**
    if $isInfilt(a)$ **then**
      $task(a) \leftarrow taskFromPool(infiltID(a), t)$
    else
      $task(a) \leftarrow newTask(i)$
    $send(a, tasks(a))$

  $totalPoW \leftarrow 0$
  $revenue \leftarrow 0$
  $infiltRevenue \leftarrow 0$

  for each $a \in \mathcal{W}orkers$ **do** ▷ get PoW from workers
    $(pPoW, fPoW) \leftarrow recv(a)$
    $jobRate(a) \leftarrow pPoW$
    if $isInfilt(a)$ **then**
      $SendPoWtoPool(infiltID(a), pPoW, Null)$
    else
      $revenue \leftarrow revenue + publish(tasks(a), fPoW)$
  for each $a \in \mathcal{W}orkers$ **do** ▷ do payment
    if $isInfilt(a)$ **then**
      $infiltRevenue \leftarrow infiltRevenue + recv(a)$
  $revenue \leftarrow revenue + infiltRevenue$
  $totalPoW \leftarrow$ each $jobRate(a)$
  for each $a \in \mathcal{W}orkers$ **do**
    $pay(a, revenue \times jobRate(a)/totalPoW)$

detect the infiltration of mining pool to prevent block withholding attack.

Our detection method consumes little resource because a pool does not need to put a lot of mining power to sensors.

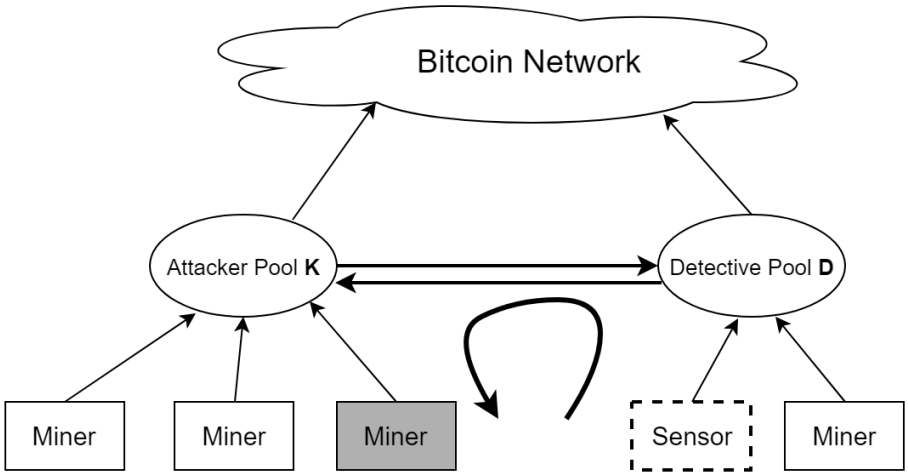

Figure 4: Sensing of Block Withholding Attack

5.2 Phase 2: Punishment on Infiltration

In this subsection, we propose the action after detection. Despite the detection of the block withholding attack, the detection itself does not change the fact that the attack continues to cause damage to the mining pool. For this reason, we need to deal with the attacker we detected.

Since the detective pool knows which mining pool is attacking it, it can react by adjusting the compensation he sends to the pool. That is, a pool can modify share of block withholding attack miners to punish the attacker. We set this punishment coefficient value as h. The loss from block withholding attack is zero when h value is as below Theorem 5.1. The reason why the detective pool does not need to disconnect the attack miners is about **Public Pools** assumption in Section 4.1. With this assumption, the attackers can join again to other pools whenever they want. Hence, disconnection is not a proper solution. Instead of disconnection, we propose a method to modify share of attacker miners.

Algorithm 5 Punishment function

```
1: procedure PUNISHRATE(workers, totalPoW, jobRate)
2:     for each a ∈ Workers do
3:         if isAttacker(a) then
4:             jobRate(a) ← h × jobRate(a)
5:     totalPoW ← eachjobRate(a)
```

Theorem 5.1. *When the detecting pool finds the attack, the pool should modify*

Hashrate of the Attacker	Hashrate of the Detective	Punishment Range
0.2	0.3	h $\leq$ 0.33614
0.2	0.2	h $\leq$ 0.223927
0.1	0.2	h $\leq$ 0.210881
0.1	0.1	h $\leq$ 0.105425

Table 1: Punishment Coefficient Range of Several Context

the reward of suspicious miners less than the proportional value below.

$$h = -\frac{(1-\alpha\tau)(\beta+\alpha\tau)(1-\frac{1}{\alpha\tau})}{\alpha\tau}$$

Proof. h value can be solved by below equation. In this calculation, we assumed sensor mining power is zero.

$$\beta = \frac{\beta}{1-\alpha\tau}(1-h\frac{\alpha\tau}{\beta+\alpha\tau})$$

□

When the attacker's mining power is 0.2, proportional to total mining power 1, the modifying border value h varies by the mining power of the detecting pool. Additionally, we assumed the attacker uses optimal infiltration mining power to maximize revenue. Table 5.2 shows several sample values.

6 Analysis

To demonstrate the validity of our method, at first, we investigate security analysis. Secondly, We compare our method with other countermeasures. In order to compare, we use three viewpoints for PoW ecosystem.

6.1 Security Analysis

We should consider the possibility that Attackers can detour or exploit our method. This subsection checks the various attack scenarios to defeat our method. If need, we consider attackers even cross our assumptions(Section 4.1) for rigorous analysis.

Task without *Coinbase* Transaction To detour our method by hiding *Coinbase* transaction information, we can consider *Task with Coinbase Transaction* in Section 4.1 is broken. The attacker can set all transactions in the block to mine and calculate all header value except nonce in Figure 1. Then he sends PoW task to his miners without exposing *coinbase* transaction. In this case, Algorithm 4 cannot check the attack. However, *hashMerkleRoot* must be exposed to PoW in the header. This value is dependent on transactions in the

block. Thus, even if we check *hashMerkleRoot* or all transaction in PoW, this variation is also detectable.

Anonymized Infiltration Miners Even though the attacker is detected, loss by attack continues without punishment or proper measures. That is why we suggested punishment share strategy in Section 5.2. The theorem 5.1 assumed we could control all share of infiltration miners when we detected. The attacker can however hide information of the infiltration miners into a lot of miner IDs and many source addresses. Then, we cannot control all share of infiltration IDs even though we detect the block withholding attack. This strategy occurs quite significant overhead to anonymize and manage the source of infiltration miners. For this reason, we assume that the attacker is less likely to try anonymization of infiltration miners.

Attack with Private Infiltration In practice, unlike public pool assumptions in Sections 4.1, closed pools exist in the real world. The closed pool means the pools mine with their private mining power and does not allow random join of miners. For this reason, the block withholding attack is not possible to them since infiltration to the closed pools is impossible. For example, BTC.TOP and Bitfury are known as pools cannot be joined.[1]. Between the concepts of the public pool and the closed pool, we can conjecture that the existence of an intermediate pool with a big proportion of private mining powers in its pool. Here, we can consider an attacker who cannot be infiltrated to detect. If the closed pool tries block withholding attack, our method is not valid as the detective pool cannot join to the closed pool. Also, if the intermediate form of the pool performs this attack and distributes the task of victim pool only to its mining power, our method is not valid because the infiltration miners of the detective pool cannot get block withholding task. We leave it as an open problem that when some pools have enough private mining power, they can act asymmetric capability in the mining game.

Misunderstanding as an Attack If two pools try to detect the block withholding attack against each other, they share their tasks and decide that their opponent is attacking each pool. Our method involves a punishment process. However, it does not operate the block withholding attack. Then we only reduce the share to each other, so there is no dilemma situation that Eyal [14] showed. To be more specific, as they infiltrate each other and work honestly, the total hashrate is conserved. Then the effect of reducing each other's interests is negligible.

6.2 Comparison with other methods

Based on the six conditions in Section 3.3, we evaluate our method and compare other countermeasures as below Table 6.2.

The two-phase method by Rosenfeld [20] does not meet Condition 2 *Compaibility* because they changed the mining process and need additional fields.

Properties	Our Method	[20]	[15]	[13]	[14]	[18]
1. No Loss	O	O	O	O	X	O
2. Compatibility	O	X	X	X	O	O
5. Fairness	O	O	O	O	O	X

Also, another two-phase method by Eyal [15] is not proper with Condition 2 *Compaibility* since miners should solve additional and different hash calculation. Another scheme in [13] using the secret value of mining pools need to modify overall Bitcoin protocol. Although it is not explicit, their methods even can occur considerable overhead on additional calculation. If it is severe so that mining pool should waste their resource on overhead, they cannot meet Condition 1 *No Loss*.

Pools fee method by Eyal [14] is improper to Condition 1 *No Loss*. This method directly gives new miners penalty, even though the mining pool does not get a disadvantage. This fee gives a negative impact on the participation of new minors.

In [18], the policy to give more compensation to fPoW was discussed. This policy does not meet Condition 3 *Fairness on Miners* as it is hard to find fPoW for small miners.

Lastly, our method seems to satisfy all conditions we suggested in Section 3.3. It does not mean our method is an optimal solution. As we referred to Section 6.1, this method can have some drawbacks in a real environment without assumptions.

6.3 Adaptibility

This method is using the fact that PoW task (valid blocks) always includes a beneficiary address of its miner to give a reward of PoW. This method can be adapted to almost PoW blockchains. We can use it in Bitcoin-based blockchains(Bitcoin Gold, Bitcoin Cash, Litecoin) for sure. In Ethereum [8], it is adaptable as its header contains a beneficiary address directly. Besides, our method applies to other cryptocurrencies, for example, Ethereum [10] and Dogecoin [7], using PoW.

7 Conclusion

In this paper, we propose a two-phase attack against the attack. Our method exploits the structure in which block withholding attacks share the work of the victim pool within its pool. Initially, the detective pool infiltrates the attacker pool to check that its task is shared. If the attack is detected by confirming the infiltration, it decreases the damage by reducing the profits shared by the attacker pool.

This method has broad compatibility and applicability because PoW mining takes advantage of the essential characteristics of generating beneficiary-

dependent blocks. It seems to be safe against modified block withholding attacks. Besides, it is challenging to detour even if the format of the task that the attacker delivers to miners is changed. Since it uses features common to blockchains using PoW, it applies to other cryptocurrencies using PoW. While other existing countermeasures do not meet various conditions to be a useful method, our method seems to meet all the conditions we proposed.

References

[1] 10 best and biggest bitcoin pools. https://www.buybitcoinworldwide.com/mining/pools/. Accessed: 2018-10-04.

[2] Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf. Accessed: 2018-10-04.

[3] Bitcoin protocol documentation. https://en.bitcoin.it/wiki/Protocol_documentation. Accessed: 2018-10-04.

[4] Bitcoin wiki : Coinbase. https://en.bitcoin.it/wiki/Coinbase. Accessed: 2018-10-04.

[5] Bitcoin wiki : Comparison of mining pool. https://en.bitcoin.it/wiki/Comparison_of_mining_pools. Accessed: 2018-10-04.

[6] Bitcoin wiki : Pooled mining. https://en.bitcoin.it/wiki/Pooled_mining. Accessed: 2018-10-04.

[7] Dogecoin project. https://dogecoin.org/. Accessed: 2018-10-04.

[8] Ethereum: A secure decentralised generalised transaction ledger byzantium version. https://ethereum.github.io/yellowpaper/paper.pdf. Accessed: 2018-10-04.

[9] Mining hardware comparison. https://en.bitcoin.it/wiki/Mining_hardware_comparison. Accessed: 2018-10-04.

[10] A next-generation smart contract and decentralized application platform. https://github.com/ethereum/wiki/wiki/White-Paper. Accessed: 2018-10-04.

[11] P2p pool: Decentralized bitcoin mining pool. http://p2pool.org/. Accessed: 2018-10-04.

[12] Wikipedia free encyclopedia: Mining pool. https://en.wikipedia.org/wiki/Mining_pool. Accessed: 2018-10-04.

[13] Samiran Bag, Sushmita Ruj, and Kouichi Sakurai. Bitcoin block withholding attack: Analysis and mitigation. *IEEE Transactions on Information Forensics and Security*, 12(8):1967–1978, 2017.

[14] Ittay Eyal. The miner's dilemma. In *Security and Privacy (SP), 2015 IEEE Symposium on*, pages 89–103. IEEE, 2015.

[15] Ittay Eyal and Emin Gün Sirer. How to disincentivize large bitcoin mining pools. *Blog post: http://hackingdistributed. com/2014/06/18/how-to-disincentivize-large-bitcoin-mining-pools*, 2014.

[16] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. *Communications of the ACM*, 61(7):95–102, 2018.

[17] Yujin Kwon, Dohyun Kim, Yunmok Son, Eugene Vasserman, and Yongdae Kim. Be selfish and avoid dilemmas: Fork after withholding (faw) attacks on bitcoin. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 195–209. ACM, 2017.

[18] Loi Luu, Ratul Saha, Inian Parameshwaran, Prateek Saxena, and Aquinas Hobor. On power splitting games in distributed computation: The case of bitcoin pooled mining. In *Computer Security Foundations Symposium (CSF), 2015 IEEE 28th*, pages 397–411. IEEE, 2015.

[19] Loi Luu, Yaron Velner, Jason Teutsch, and Prateek Saxena. Smart pool: Practical decentralized pooled mining. *IACR Cryptology ePrint Archive*, 2017:19, 2017.

[20] Meni Rosenfeld. Analysis of bitcoin pooled mining reward systems. *arXiv preprint arXiv:1112.4980*, 2011.

Algorithm 4 Detective Pool P_d against Mining Pool P_k

variables
  List $workers$ ▷ registered miners
  List $isSensor$ ▷ registered miners for sensing
  List $sensorID$ ▷ registered miners' ID in target pool
  List $jobRate$ ▷ amount of submitted PPoW
  Float revenue ▷ total reward of mining pool
  Float infiltRevenue ▷ total reward of target mining pool
  Bool isDetect
  String coinbase ▷ coinbase transaction string of mining P_i
end variables

procedure MANAGE MINING
  for each $a \in \mathcal{W}orkers$ **do**
    if $isSensor(a)$ **then**
      $task(a) \leftarrow taskFromPool(sensorID(a), i)$
      if $isMyTask(task(a), coinbase)$ **then** ▷ check sensors
        $isDetect \leftarrow True$
    else
      $task(a) \leftarrow newTask(i)$
    $send(a, tasks(a))$

  $totalPoW \leftarrow 0$
  $revenue \leftarrow 0$
  $infiltRevenue \leftarrow 0$

  for each $a \in \mathcal{W}orkers$ **do** ▷ get PoW from workers
    $(pPoW, fPoW) \leftarrow recv(a)$
    $jobRate(a) \leftarrow pPoW$
    if $isSensor(a)$ **then**
      $SendPoWtoPool(sensorID(a), pPoW, fPoW)$
    else
      $revenue \leftarrow revenue + publish(tasks(a), fPoW)$

  for each $a \in \mathcal{W}orkers$ **do**
    if $isSensor(a)$ **then**
      $infiltRevenue \leftarrow infiltRevenue + recv(a)$
  $revenue \leftarrow revenue + infiltRevenue$

  if $isDetect$ **then** ▷ Punish the reward of the attacker
    $punishRate(workers, totalPoW, jobRate)$

  $totalPoW \leftarrow$ each $jobRate(a)$ ▷ do payment
  for each $a \in \mathcal{W}orkers$ **do**
    $pay(a, revenue \times jobRate(a)/totalPoW)$