Countering Block Withholding Attack Efficiently

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

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

***Abstract*—Bitcoin, well-known cryptocurrency, selected Poof-of-Work (PoW) for its security. PoW mechanism incentivizes participants and deters attacks on the network. So far, Bitcoin's PoW has been adopted in many cryptocurrencies. 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 of countermeasures cause many changes in the mining algorithm itself, which makes it impractical. In this paper, we propose new countermeasure to prevent block withholding attack effectively. Mining pools can adapt our method without changing their mining environment.**

***Index Terms*—Blockchain, Bitcoin, Mining pool, Security analysis**

I. Introduction

In Bitcoin [10] and Ethereum [3], Poof-of-Work (PoW) is adapted to make its system secure, especially against Sybil attack. Miners organize mining pools and join in mining pools to mine blocks and share rewards efficiently. Bitcoin protocol has a value and mining difficulty which is automatically adjusted by block generation time. By adjusting this difficulty, Bitcoin network generates a block every 10 minutes, and a miner who generates the block gets a reward. Sometimes miners work together to maximize profits, which we call mining pool.

Researchers have shown several vulnerabilities of 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. Eyal [7] proved that his selfish mining is useful even if a miner has over 25% of total mining power.

Second, Block Withholding (BWH) attack is related to competition between mining pools. Mining pools compete with each other to get Bitcoin reward. The concept *difficulty*, numerical value, means that only blocks which have hash values 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 the tact they are mining by submitting partial blocks (partial Proof-of-Work; pPoW). pPoW has a bigger hash value than the *difficulty* in Bitcoin protocol. Eyal [5] 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 and keeping fPoW in other mining pools.

[*] Corresponding author

Contributions

- *Method to detect block withholding attack.* If there is no bad intent, miners have 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 [8]. 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 mining algorithm, 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.

II. Preliminary

A. Proof-of-Work of Bitcoin

Bitcoin uses PoW for its security property. PoW verifies time-consuming work spent on data. It needs to be verified 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. Blockchain system uses PoW to profit specific miner. Thus the block should depend on the miner's unique value.

In Bitcoin, *difficulty* is a specific numeric value, and means the difficulty of mathematical puzzle. Miners who try to generate a block should find nonce which makes a hash value of a block less than the difficulty of the current Bitcoin protocol.

The hash value is calculated by double iteration of the SHA256 hash algorithm. Miners try to find a nonce satisfyinh the hash value of 6 headers (*Version*, *Time*, *Bits*, *Nonce* (*ctr* in our algorithms), hashPrevBlock, hashMerkleRoot) is less than *difficulty*. This random process to find nonce is very more time-consuming work when *difficulty* is smaller.

Now let's look at the Bitcoin block structure. In particular, we need to focus on the following values, *merkleRootHash* and coinbase transaction. The *merkleRootHash* value depends on all transactions in the block, and the first transaction in the block must be the coinbase transaction. The coinbase transaction, or generation transaction, is a special transaction in the Bitcoin protocol that differs from a standard transaction as it creates coins from nothing. While regular transactions use the 'inputs' section to refer to their parent transaction outputs, a coinbase transaction has no parent (i.e., sender), and creates new coins from nothing. Coinbase transactions are always constructed by a miner and will contain a reward for efforts expended during the PoW mining process. Because it is directly related to miners' reward, the coinbase transaction is the basis for identifying who has mined a block.

B. Mining Pool

Since a PoW requires a very big mining power, a pooled mining is a dominant approach to generate a block. 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.

Even though miners work hard, there is a possibility that small miners cannot find any block in the whole time. Even though they did not find anything, but to prove that they worked hard, they submit pPoW 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 [2, 4].

III. Related Works

A. Block Withholding Attack

Rosenfeld [11] 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.

Eyal [5] advanced Rosenfeld's idea. He 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 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 2 in section IV. In this paper, we suggest a countermeasure against this kind of block withholding attack.

Kwon [8] showed that the attacker can earn more revenue if he releases fPoW of infiltration in a specific situation. 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.

B. Countermeasures

So far, researchers have suggested several counter methods. Main idea of previous research is based on the idea that the attack is possible providing that miners can distinguish between pPoW and fPoW. So they commonly change PoW into two steps of PoW. So it is called as 'two-phase PoW'. At first, Rosenfeld [11] modified the original PoW process with the secret value so that the miners might not check the validity of his block. Bag et al. [1] also suggested PoW schemes using secret values with rigorous definitions of security properties. Secondly, another two-phase PoW by Eyal and Sirer [6] employs new condition in order to decide PoW is valid. The condition is that the block which has the hash *SHA256(SIG(header, privkey))* of that signature is smaller than a second difficulty parameter Y is valid. Different from Rosenfeld's method, it needs the private key and its signature to divide the PoW process into the two-phase. Miners cannot know the validity of their block because they do not know the coinbase transaction's private key. There are another approaches. Luu [9] suggested 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 [5] 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. This method makes pools not flexible for miners since miners do not want to pay fees of course.

C. Necessary Conditions for Practical BWH Countermeasure

Countermeasures against the attack should be practical. Luu [9] discussed seven desired properties to review countermeasures. Kwon [8] also discussed drawbacks of existing countermeasures. Through the comprehensive analysis, we will suggest below three necessary conditions for the countermeasure against the block withholding attack. Also, we use these criteria for comparing all the previous methods including ours at Section VI-B.

- **No loss**: A counter method 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 the 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. Thus, if a countermeasure is adaptable in existing ASIC miners, it is a better countermeasure.
- **Fairness**: A pool should treat miners only by their contribution, not by their scale or other perspectives when the pool adopts a method. If a method does not have a fair policy, some miners do not want to join the pool because of unfair treatment.

IV. Model of Mining Pool

We modeled the elements of the mining pool by referring to the model of Eyal [5]. The format and some contents of the algorithms is little bit modified for readability. In this paper, the parameters indicate as follows.

H is the SHA256 Hash function
T is the difficulty of the Bitcoin protocol
D is the difficulty of the mining pools for contribution
ctr is nonce
$workers$ is the registered miners
$contribution$ is the amount of submitted pPoW
rev is the total revenue of mining pool
$isInfilt$ is the list of miners for infiltration mining
$isSensor$ is the registered miners for sensing

A. Assumption

We assumed two items in this paper.

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.
2) Task with Coinbase Transaction: The mining pool sends PoW task containing its coinbase to the miners.

B. Elements of Mining Pool

1) Honest Miner: Honest 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 PoW value which includes $nonce$ value if it meets the difficulty condition. They get revenue from the mining pool based on their share of PoW.

2) Honest Pool: Our pool model is the simple model which has a centralized pool manager. Algorithm 1 shows how **Honest Pool** P_i operates with miners. It has the list of miners(*workers*) and the list of their amount of pPoW share(*jotRate*). The manager of P_i gives a task to its 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(*contribution*).

Algorithm 1 Mining Pool P_i

```
1:  procedure MANAGE POOLED MINING
2:      $reward \leftarrow 0$
3:      $w \leftarrow CreateTask()$
4:      for each $a \in Workers$ do
5:          $send(a, w)$
6:      for each $a \in Workers$ do
7:          $(ctr \parallel w) \leftarrow recv(a)$
8:          if $(H(ctr \parallel w) < T)$ then          ▷ only full PoW
9:              $rev \leftarrow rev + publish(ctr \parallel w)$
10:                             ▷ publish a block and get Bitcoin
11:         $contribution[a].add()$
12:                                ▷ add contribution to PoW
13:     $totalPoW \leftarrow$ each $contribution(a)$
14:     for each $a \in Workers$ do
15:         $pay(a, rev \times \frac{contribution(a)}{totalPoW})$
16:                             ▷ distribute revenue to miners
```

C. Elements of Block Withholding Pool

1) Honest Miner: In [5], they modeled an honest miner and a block withholding miner respectively. To perform the block withholding attack proposed initially by Rosenfeld [11], we need a model of the block withholding miner. However, as we mentioned in section III-A, this paper deals with the block withholding attack between pools proposed by Eyal [5]. That is, like Eyan's scenario, we assume that miners in the block withholding pool are honest as miners in the honest pool.

2) Block Withholding Pool: Figure 1 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. 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 1. 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 does not submit fPoW to mining pool P_i.

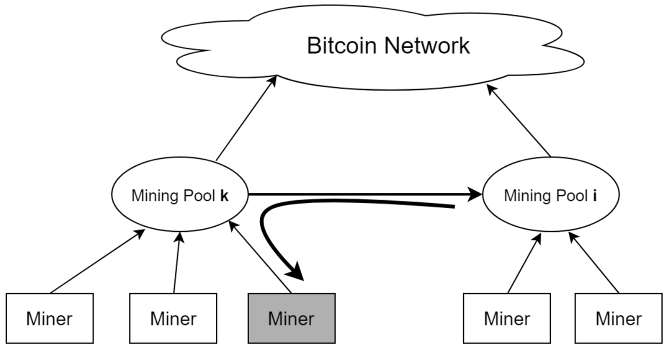

Figure 1. Block Withholding Attack

Algorithm 2 shows the block withholding attack controlled by a mining pool manager.

Algorithm 2 Block Withholding Pool P_k against Pool P_i

```
1:  procedure MANAGE MINING
2:      for each $a \in Workers$ do
3:          if $isInfilt(a)$ then
4:              $task(a) \leftarrow taskFromPool(i)$
5:          else
6:              $task(a) \leftarrow createTask()$
7:          $send(a, tasks(a))$
8:      for each $a \in Workers$ do
9:          $(ctr \parallel w) \leftarrow recv(a)$
10:         $check \leftarrow H(ctr \parallel w)$
11:         if $isInfilt(a)$ then
12:             if $(check > T \ \& \ check < D)$ then
13:                 $SendPoWtoPool(pPoW, Null)$
14:         if $(H(ctr \parallel w) < T)$ then
15:             $rev \leftarrow rev + publish(ctr \parallel w)$
16:         $contribution[a].add()$
17:     for each $a \in Workers$ do
18:         if $isInfilt(a)$ then
19:             $rev \leftarrow rev + recv(a)$
20:     $totalPoW \leftarrow$ each $contribution(a)$
21:     for each $a \in Workers$ do
22:         $pay(a, rev \times contribution(a)/totalPoW)$
```

V. Our Method against Block Withholding Attack

Our algorithm consists of two steps, 'detection of infiltration' and 'punishment of attacker'.

A. Phase 1. Detection of Infiltration

The manager of the block withholding attacker pool K joins to the victim pool D as a miner. The miners under the attacker pool work for infiltration without their knowing. In Figure 2, the miner in the gray box work as infiltration power. The role of the attack manager is a sort of PoW proxy.

But at this very moment, if the victim pool D joins to the block withholding attacker pool K, in the same way, the victim pool can investigate which PoW task the attacker pool send to miners. In figure 2, we assume the miner with dotted line joins the block withholding attacker pool. Then it works as a sensor of the attack. This infiltrating miner works honestly, that is, it sends both pPoW and fPoW like other miners of the pool. We 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 2, 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 can put sensor miners in other pools. If the detective pool finds the task including its own coinbase transaction in the tasks of infiltrating sensor miners, it can conclude that the attack has begun. More

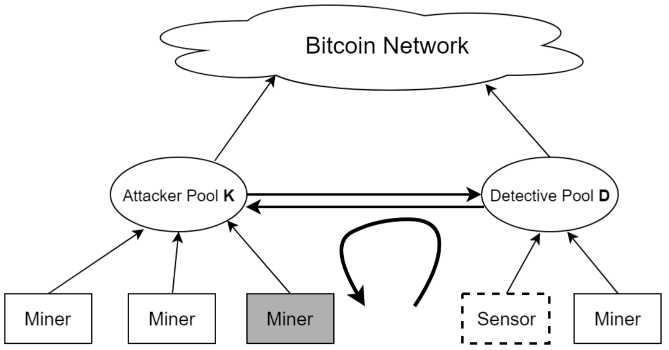

Figure 2. Sensing of Block Withholding Attack

specifically, the block withholding attack can be implemented by the proxy mining structure as algorithm 2 and figure 2. Our detection method consumes little resource because a pool does not need to put a lot of mining power to sensors.

Algorithm 3 Detective Pool P_d against Mining Pool P_k

```
1:  procedure MANAGE MINING
2:      for each $a \in Workers$ do
3:          if $isSensor(a)$ then
4:              $task(a) \leftarrow taskFromPool(k)$
5:              if $isMyTask(task(a), coinbase)$ then
6:                  $isDetect \leftarrow True$
7:          else
8:              $task(a) \leftarrow newTask(i)$
9:          $send(a, tasks(a))$
10:     for each $a \in Workers$ do
11:         $(ctr \parallel w) \leftarrow recv(a)$
12:         if $isSensor(a)$ then
13:             $SendPoWtoPool(sensorID(a), (ctr \parallel w))$
14:         else$(H(ctr \parallel w) < T)$
15:             $rev \leftarrow rev + publish(ctr \parallel w)$
16:         $contribution[a].add()$
17:     for each $a \in Workers$ do
18:         if $isSensor(a)$ then
19:             $rev \leftarrow rev + recv(a)$
20:     if $isDetect$ then    ▷ Punish the reward of the attacker
21:         $punish(workers, totalPoW, contribution)$
22:     $totalPoW \leftarrow$ each $contribution(a)$
23:     for each $a \in Workers$ do
24:         $pay(a, rev \times contribution(a)/totalPoW)$
```

B. 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 after detection.

Since the detective pool knows which mining pool is attacking it, it can react by adjusting the compensation which he sends to the pool. That is, a pool can modify the share of block withholding attack miners in order to punish that attacker. We set the punishment parameter as h. The loss from block withholding attack is zero when h value is as below theorem V.1. The reason why the detective pool does not need to disconnect the attack miners is about **Public Pools** assumption in section IV-A. 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 punish the share of attacker miners.

Algorithm 4 Punishment function

```
1: procedure PUNISH($workers, totalPoW, contribution$)
2:     for each $a \in Workers$ do
3:         if $isAttacker(a)$ then
4:             $contribution(a) \leftarrow h \times contribution(a)$
5:     $totalPoW \leftarrow eachcontribution(a)$
```

Theorem V.1. *When the detecting pool detects the attack, the pool should modify the reward of suspicious miners less than the proportional parameter value below.*

$$h = \beta + \alpha\tau$$

where

α *is hashrate of the selfish pool*
β *is hashrate of the detecting pool*
τ *is a proportion of infiltration of the selfish pool*

Proof. The parameter h can be solved by below equation. In this calculation, we assumed sensor mining power is negligible.

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

□

When the attacker's mining power is 0.2, proportional to total mining power 1, the modified border value h varies by the mining power of the detecting pool. In addition, we assume the attacker uses optimal infiltration mining power to maximize revenue. Table I shows several sample values.

Table I
PUNISHMENT COEFFICIENT RANGE OF SEVERAL CONTEXT

α	β	Valid punishment value
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

VI. ANALYSIS

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

A. Security Analysis

We should consider the possibility that attackers can detour or exploit our method. It is to check the various attack scenarios to defeat our method. If need, we consider attackers even cross our assumptions (section IV-A) 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 IV-A is broken. The attacker can set all transactions in the block to mine and calculate all header value except nonce. Then he sends PoW task to his miners without exposing the coinbase transaction. In this case, algorithm 3 cannot check the attack. However, merkleRootHash value must be exposed to PoW in the header. This value is dependent on transactions in the block. Thus, providing that we check merkleRootHash 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 strategy in section V-B. The theorem V.1 assumed we could control all share of infiltration miners when we detected. The attacker can, however, hide information of the infiltrating 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 IV-A, closed private pools exist in the real world. The closed pool means the pools mine with their private mining power. It does not allow free join/leave of miners. For example, BTC.TOP and Bitfury are known as pools cannot be joined freely [12]. For this reason, the block withholding attack is not possible to the closed pool.

Between the concepts of the public pool and the closed pool, we can conjecture that the existence of an intermediate hybrid pool with a big proportion of private mining powers in its public pool. Here, we can consider an attacker who cannot be infiltrated to detect. If a closed pool tries block withholding attack, our method is not valid as the detective pool cannot join to the closed pool. 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.

B. Comparison with other methods

Table II
COMPARISON OF COUNTERMEASURES

Properties	Our Method	[11]	[6]	[1]	[5]	[9]
1. No Loss	O	O	O	O	X	O
2. Compatibility	O	X	X	X	O	O
3. Fairness	O	O	O	O	O	X

Based on the conditions in Section III-C, we evaluate our method and compare other countermeasures as shown in table II. The two-phase method by Rosenfeld [11] does not meet Condition 2 *compatibility*. Because they changed the mining process and need additional fields. Also, another two-phase method by Eyal [6] is not proper with Condition 2 *compatibility* since miners should solve additional and different hash calculation. The scheme in [1] 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 mining pools should waste their resource on overhead, they cannot meet Condition 1 *No Loss*.

Pools fee method by Eyal [5] 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 [9], the policy to give more compensation to fPoW was discussed. It does not meet Condition 3 *Fairness on Miners* for that it is hard to find fPoW for small miners.

Lastly, our method seems to satisfy all condition. It does not mean our method is an optimal solution. As we referred to section VI-A, it can have some drawbacks in a real environment without assumptions.

C. Adaptibility

Our method is using the fact that PoW task (valid blocks) always includes a beneficiary address of its miner to give a reward of PoW. Thus it can be adapted to almost PoW blockchains. We can use it in Bitcoin-based blockchains (Bitcoin Gold, Bitcoin Cash, Litecoin) for sure. In Ethereum [13], it is adaptable as its header contains a beneficiary address directly. Besides, our method can applies to other cryptocurrencies if they use PoW.

VII. Conclusion

In this paper, we propose the efficient countermeasure against the block withholding attack. Our method exploits the structure of the block withholding attack which shares the work of the victim pool. Initially, the detective pool infiltrates the attacker pool to check that its task is shared. If the attack is detected by checking PoW task, it decreases the damage by reducing the profits shared by the attacker pool. Furthermore, it can earn more revenue than before.

Our method provides broad compatibility and applicability as PoW mining takes advantage of the essential characteristics of generating beneficiary-dependent blocks. It seems to be safe against variants of block withholding attacks. Besides, it is challenging to detour even if the format of the task that the attacker delivers to miners is changed. Due to using common features of PoW blockchains, we believe it can be applied to other cryptocurrencies using PoW. While other existing countermeasures do not meet the various conditions to be a useful method, our method seems to meet all the conditions we suggested.

Acknowledgement

This work was supported by Institue for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No.2018-0-00532, Development of High-Assurance ($\geq$ EAL6) Secure Microkernel)

References

[1] 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.

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

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

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

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

[6] 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.

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

[8] 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.

[9] 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.

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

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

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

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