Arbitrage attack: Miners of the world, unite!

Yuheng Wang[1], Jiliang Li[1], Zhou Su[1], and Yuyi Wang[2]

[1] School of Cyber Science and Engineering, Xi'an Jiaotong University, China
[2] ETH Zürich
wangdahu1211@stu.xjtu.edu.cn jiliang.li@xjtu.edu.cn zhousu@ieee.org
yuyiwang920@gmail.com

Abstract. Blockchain oracles are introduced to mitigate the gap between blockchain-based applications and real-world information. To solve the centralization problem of current oracle systems, many decentralized protocols have been designed. In this paper, we define the basic model for decentralized oracles that rely on unencrypted transactions for verification and adjustment tasks. Furthermore, we introduce Arbitrage attack against such decentralized oracles carried out by rational miners and mining pools. We analyze the attack based on game-theoretic methods. Moreover, we briefly discuss the price of anarchy to demonstrate the characteristic of attackers' cooperation union under different circumstances.

Keywords: Blockchain · Decentralized Price Oracle · Nash equilibrium · Price of Anarchy(PoA)

1 Introduction

If we take a look at all the impressive events that happened in 2021, the epic "Gamestop(GME) Short Squeeze" war that happened among the union of retail investors, certain hedge funds as well as short-sellers at the very beginning of this year can't avoid discussion [28]. Gamestop is an offline game retailer company selling games, game consoles, and accessories. Under the influence of e-commerce, the revenue of Gamestop had been greatly impacted. Therefore many hedge funds and short-sellers believe that the stock price is going to drop and took the opportunity to short sell Gamestop's stock. However, many Gamestop's loyal customers as well as some speculators, on the contrary, started to buy in Gamestop's stock in order to hinder venture capital firms' plan. Due to the internet propaganda, more and more retail investors and even some famous investors also participated in and started to buy in the stock. Consequently, Gamestop's stock price had surprisingly risen a lot, which caused many venture capital firms great loss [31].

The important reason for this great victory is that a large number of independent retail investors, who even don't know each other before, take the same action and manage to "change" and "manipulate" the price of Gamestop's stock price together. From the view of a bystander, it seems that these retail investors form a temporary union to achieve this goal. So will this kind of temporary union appear in other fields? Currently, there have been extensive discussions about possible applications of blockchain technology, and decentralized finance (DeFi) has become a main driver of blockchain adoption. Compared with the traditional finance industry, DeFi uses transparent and immutable on-chain smart contracts to realize trading activity instead of centralized custodians, banks, and brokers. So will a temporary union appears in the DeFi system and "manipulate" the DeFi market price to obtain high profit?

Noted that, most of DeFi projects rely on smart contracts to perform trading operations, and these smart contracts need an oracle (also known as data feed) to carry the real-world information (e.g., the current market price of Bitcoin) to the blockchain as evidence to trigger execution. This is because smart contracts are only able to use on-chain information, to guarantee the safety of blockchain systems. Most of the oracle systems currently being used are centralized oracles that depend on trusted third-party facilities or platforms (e.g., Town Criers [8]) to fetch outside information, and later the information will be given to the customer through a transaction proposed by the oracle's on-chain account.

Although such a method is effective, the deployment of centralized oracles brings back the problem of centralization. To mitigate the problem, decentralized oracles protocols have been proposed. Decentralized oracles (e.g., the NEST protocol [18]) try to avoid the centralization problem by letting different users propose opinions after hearing a query task and make decisions for the final output together. Consequently, some inaccurate and malicious personal opinions might inevitably be brought in. Therefore verification and adjustment methods will be the only guarantee of decentralized oracle's reliability. And in most cases, procedures of opinion proposition, verification as well as adjustment are realized in the form of on-chain transactions.

Most studies about decentralized oracles only take opinion proposers' influence into consideration and neglect miners as well as mining pools. It is well known that miners, especially mining pools, have considerable influence on the publishing of transactions, which can also cause damage to the reliability and accuracy of decentralized oracles. However, to our notice there have been very few decentralized oracle studies considering the power of miners and mining pools, especially when tampering with the publishing of certain transactions will benefit them more. We believe that under such circumstances, the miners and mining pools can form a temporary union just like retail investors in "Gamestop Short Squeeze" event to manipulate the output of decentralized oracles and obtain greater profit. So in this paper, we introduce the Arbitrage attack, an attack against decentralized oracles by the union of miners and mining pools, and carry out further analysis based on game theory.

1.1 Related Work

Oracles have been a very hot research topic these days due to the prosperous applications based on smart contracts. Different kinds of oracle protocols have been proposed. In general, oracles can be classified into two different types: centralized and decentralized.

For centralized oracles, there is always a third party involved in the information transportation task. Zhang et al. presented a very famous and widely used centralized oracle model, Town Crier (TC) [8]. TC guarantees reliability and confidentiality by using Software Guard Extensions (SGX). Each user's query request is executed by codes running in SGX, and corresponding information is fetched from corresponding official websites. Similarly, the Provable oracle in [29] directly fetches information from data sources and uses TLSNotary to prove the integrity. And the PriceGeth oracle from [30] continuously sends fetched data through the oracle's smart contract. However, the implementation of centralized oracles like the ones mentioned above brings centralized problems back to the blockchain system. Since safety, confidentiality, and reliability are once again dependent on a third party that needs to be trusted.

Therefore, various decentralized oracle protocols have been proposed to mitigate the shortcomings of centralized oracles [10]. In Augur, proposed by Peterson et al. [15], all users can vote on the possible outputs given by the oracle system, and each vote weights differently based on users' reputation. Adler et al. designed an oracle protocol ASTRAEA [14] which outputs results proposed and certified also by the votes of users. However, such an oracle can only report data in the form of Boolean propositions, each user will vote for agreement or disagreement for query questions e.g., today's weather is sunny. Similarly, in Witnet oracle, designed by de Pedro et al. [16], information is retrieved and attested by different users based on their reputation.

The rational mining pool is another important question attracting many researchers' attention. In [20], Eyal et al. analyze the feasibility and effect for mining pools to carry out selfish mining attack, which shows the power of mining pools in the blockchain system. Besides, there are other kinds of attacks due to the rationality of mining pools like bribing [24,25], and front-running [26,27] that may tamper with blockchain-based applications.

1.2 Our Contributions

In this paper, we first provide a basic model for decentralized oracles that depend on unencrypted transactions to realize adjustment and verification like NEST protocol [18]. This model contains the necessary procedures for a decentralized oracle of this type to form a reliable output.

Based on the proposed decentralized oracle model, we introduce the Arbitrage attack which is carried out by rational miners and mining pools. We argue that it is possible for rational miners and mining pools to maliciously manipulate the decentralized oracle's output by taking the same actions together like a temporary union in order to obtain more profit.

We further conduct a game-theoretic analysis on this attack. We use a multi-stage static game of perfect information to model the whole attack procedure and analyze each participant's Nash equilibrium strategies during the game to depict the progress of the whole attack under different circumstances. When possible. We also compute the price of anarchy, the ratio of the social costs of the worst Nash equilibrium, and the social optimum, to represent the influence when the attack union is lacking cooperation.

2 Preliminaries

In the following, we outline the required background of blockchains and decentralized oracles for our proposed Arbitrage attack.

2.1 Blockchains

Smart contract and DeFi. Currently, many blockchain platforms like Ethereum have supported smart contracts, which are based on pseudo-Turing complete programs to manage cryptocurrency assets. Thanks to the flexibility of smart contracts, blockchain network users are able to do more complex tradings besides transferring to accounts, e.g., lend and borrow assets [2], margin trade [3], short and long trading [3] and derivative assets creation [2], and all these complex tradings constitute the foundation of Decentralized Finance (DeFi).

Mining pool. A mining pool refers to a group of miners that gather their computational power together to solve the POW problem and divide the revenue for the creation of a new block according to each miner's contribution [5]. The chance of solving the cryptography puzzles is prominently increased by forming mining pools. Hence, miners' revenue becomes higher and more stable. At present, nearly 80% hashrate of Bitcoin belong to less than 8 mining pools and less than 3 mining pools controls 60% of Ethereum's total hashrate [6]. Although miners' revenue is guaranteed because of the emergence of the mining pool, the problem of centralization has appeared again. To be more specific, each mining pool has a pool server connecting to an on-chain account to gather the latest transaction information, construct a block template, and send it to pool miners. In this case, the pool server can decide which transactions will be selected [7] in the block template.

2.2 Oracle

Most blockchain applications, especially in DeFi projects, rely on certain information to trigger the execution of smart contracts, but only on-chain information can be reached by smart contracts because blockchain systems are isolated from the outside world to guarantee safety [10]. Although information like the latest trading price in Uniswap can be used, these kinds of on-chain information sources are unreliable due to the lack of variety and stability. To be more specific, in Uniswap, there may only be one or two transactions about two niche digital assets at a certain time, which is not convincing enough. Besides, the price of these transactions could be proposed by the same person, which means the price might be malicious. Therefore, oracles that carry information from the outside world back to the blockchain are designed to solve this problem [11,12]. Up till now, many oracles being used bring back the problem of centralization since a trusted third party is always required for transporting information to blockchain systems from the real world [13,8]. Consequently, decentralized oracles, whose output information is decided by different users is currently a hot research spot with many problems that need to be solved [14,15,16].

3 Decentralized Oracle Model

In this section, we propose the basic model of decentralized oracles relying on unencrypted transactions based on the NEST protocol [18], which is a currently widely used decentralized oracle.

3.1 Participants

A decentralized oracle can accomplish basic query tasks generally based on four different kinds of participants: customer, oracle platform, contributors: proposers & verifiers, and information sources.

Customer. Decentralized oracles' customers could be any blockchain network user or smart contract that needs information from the outside world. They can send transactions to oracle platforms to submit an information query task, and certain fees are required for each query which constitutes the main income source of oracle platforms.

Oracle platform. Oracle platform, which usually exists in the form of smart contracts and on-chain accounts, acts as a portal that collects query tasks and outputs required information. Oracle platform is also responsible for managing collected enquiry fees, distributing rewards, and adjusting the oracle protocol's parameters.

Contributors: Proposer & Verifier. According to the definition of the decentralized oracle, each oracle's output can actually be considered as the consensus of different users, who are also noted as Contributors. To reach consensus, contributors who first propose their opinions are called Proposers, and those who verify and make adjustments to these proposed opinions are called Verifiers. Noted that all messages from both two kinds of contributors are unencrypted, therefore can be viewed by any blockchain user without limitations. For instance, assuming that a decentralized oracle accepts a request to search Bitcoin's current market price. Proposers will propose their opinions about the current price based on the information sources they subscribed to. However, these personal opinions may be inaccurate due to narrow information resources or delays, and some may even be malicious. Therefore, verifiers are required or incentivized to make adjustments or verification based on their information sources to these personal propositions, and finally form a final oracle output at last.

Information sources. Decentralized oracles' outputs are formed based on users' personal opinions, and these users can form an opinion based on their freely chosen information sources like official websites as well as the latest on-chain transactions like mentioned in Section 2.2. As a result, the choice of information resources is not limited compared with centralized oracles and only using on-chain information, which makes the oracles' outputs more convincing.

3.2 Enquiry process

Consider a decentralized oracle platform O, a customer C, a set of n contributors $Con = \{Con_1, Con_2, \cdots, Con_n\}$ and m different information sources $I = \{I_1, I_2, \cdots, I_m\}$. To finish a query task Q proposed by C, all of the participants of decentralized oracles should carry out following procedures as shown in Fig.1:

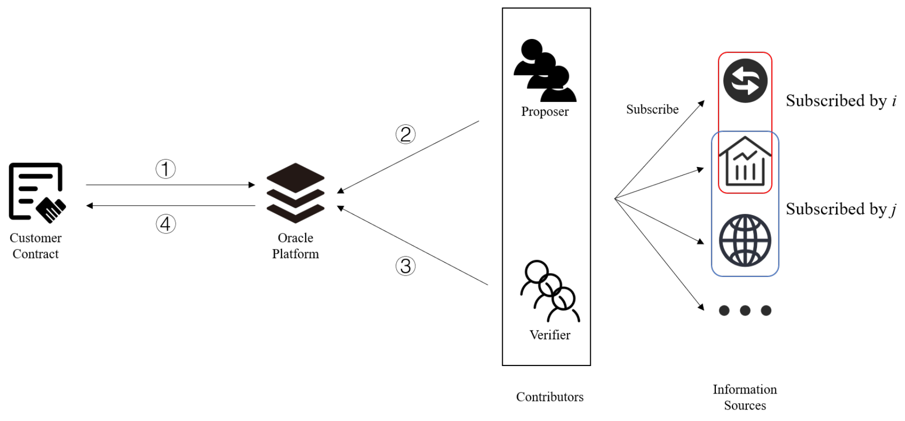

Fig. 1. Decentralized oracle model and necessary procedures for a query task proposed by a customer smart contract.

1. **Propose query request.** The customer C propose a query task Q with required query fees to decentralized oracle platform O through a transaction.
2. **Form and propose personal opinions.** After noticing a new query task, contributor Con_i will form a personal opinion p_i about the query based on personal subscribed information sources set $I_{Con_i} \subseteq I$. Noted that different personal information source sets I_{Con_i} and I_{Con_j} could be the same, totally different, or partly overlapped since there is no limitation for choosing information sources. After that, some of the contributors (proposers) will propose their personal opinion to O.

3. **Verification and adjustment** For each of the proposed personal opinions, some of the rest of the contributors, except the one who proposes this opinion, will be verifiers and may choose to make an adjustment to these personal opinions or propose verification for valid opinions. Such verification process may last for a limited time, e.g., after s blocks are mined.
4. **Deliver output.** The opinion after adjustment and verification will be the final output of O and given to C also in the form of transaction. Some encryption methods may be applied here to protect the customer's privacy and interest.

Note that in some decentralized oracle protocols, there may only be opinion proposers and no contributor specially dedicated as a verifier to propose adjustment and verification transactions. Instead, mechanisms or algorithms like reputation or voting are used by the oracle platform itself to adjust proposed opinions. Assume there are total n opinion transactions been proposed to the oracle platform for a query task, and if we set a specific transaction as the target opinion, then among the rest $n-1$ transactions, those opinions that are against the target opinions' will therefore be regarded as adjustment and verification opinions. The contributors who propose these transactions can relatively be considered as verifiers. In that case, the decentralized oracle model we proposed above can still apply to these kinds of decentralized oracle protocols. Besides, we also believe that the Arbitrage attack introduced in the following sections will also be feasible in these decentralized oracle protocols.

4 Arbitrage attack

In this section, we propose a possible attack against the decentralized oracle model mentioned before, which is carried out by rational mining pools.

4.1 Mining pools' influence on decentralized oracle

As mentioned before, mining pools with overwhelming hashrate nearly have the right to decide which transactions would be added to the chain as well as the corresponding order. The accuracy of decentralized oracles' outputs can also be influenced by mining pools since in most cases, oracles' main procedures are also carried out through on-chain transactions sent by contributors. Therefore, if some transactions are delayed or ignored because of mining pools' intervention, then the oracle's outputs might be seriously affected.

Due to the negative influence of mining pools, many mitigation methods have been implemented in the updates of the existing blockchain system. Besides, many mining pools cut their hashrate voluntarily, since the negative influence of their dominating hashrate may do damage to the blockchain system and further cause a loss to their assets in the blockchain system. Consequently, the dominating mining pools have gradually been replaced by several different mining pools with less hashrate. In that case, the capacity of mining pools is greatly weakened and transactions will be less likely to be blocked or delayed because of one or two mining pools' willingness.

What can go wrong?

Because of the distribution of total hashrate, divergence may appear among mining pools, since decisions will only be made based on each one's own benefits. However, once their target overlaps during a time period, a temporary union may appear, which is similar to retail investors in the Gamestop event. When it comes to decentralized oracles, different mining pools may take the same action and cooperate to block contributors' certain transactions to manipulate oracles' outputs and arbitrage later. This is what we called the Arbitrage attack that may happen in a decentralized oracle.

4.2 Arbitrage attack model

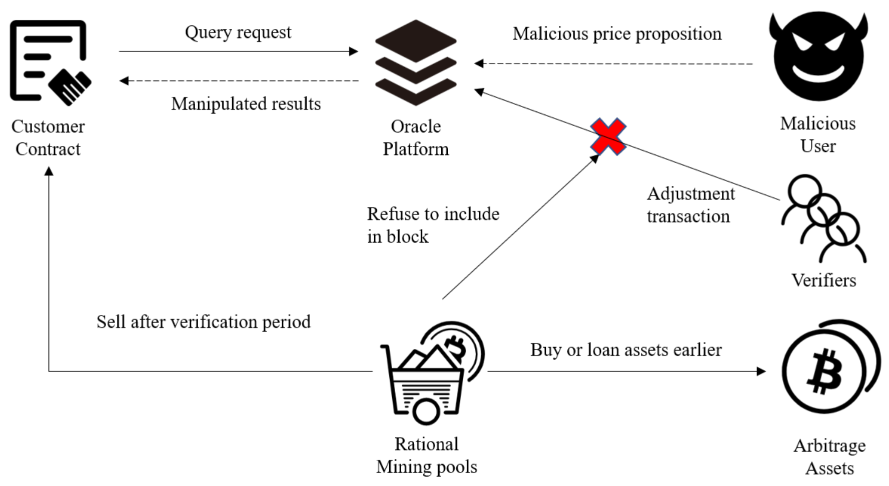

Fig. 2. Procedures of the whole Arbitrage attack. The customer contract will be caused huge damage due to the rationality of mining pools

Assuming that there are n rational mining pools in the blockchain network and search for high profit in the blockchain network (i.e., individual miners can be considered as mining pools with lower hashrate). And there is a decentralized oracle O that outputs the current price of Bitcoin, for instance, to all its consumers in the blockchain network.

When a malicious user wants to intervene in the oracle's output, it will first propose a malicious price quotation through transaction t_m which is greatly deviated from the real market price, much higher for example. According to the mechanism of O, an honest verifier may notice the difference during the verification procedure and propose an adjustment transaction t_a with gas fee. Generally, mining pools would be willing to include t_a into the next block they are currently mining and win normal profit A, which is the reward for following the protocol. However, after noticing the malicious transaction t_m, rational mining pools may hesitate because t_m represents an arbitrage opportunity for higher profit than A.

As a result, instead of simply including t_a, a rational mining pool may choose to buy or loan a large amount of Bitcoin as soon as they see the price quotation t_m, and prepare a set of arbitrage transactions to propose after the attack succeeds. Noted that these transactions are all against customer contracts of oracle O. Later they will together ignore any adjustment transactions that want to amend the quotation together like a union, since these transactions are also unencrypted and therefore can be distinguished easily. Finally, they can sell out all the Bitcoin they have by proposing a newly mined block containing arbitrage transactions set t_{arb} they want to make after the quotation t_m becomes valid. By doing so, mining pools are able to obtain higher arbitrage profit B. Consequently, customers using O as their information source will be severely influenced because of these arbitrage transactions. The whole attack process is shown in Fig.2. It is worth to be noted that, the detailed expressions and calculated methods of profit A and B should be uncertain and depend on specific decentralized oracle protocol. But generally speaking, the arbitrage profit B will always be higher than the normal profit A.

4.3 Attack feasibility

The profit of a successful Arbitrage attack is usually attractive and irresistible. Intuitively, it seems that mining pools will definitely choose to attack when B is larger than the normal profits A. However, not all of the rational mining pools will obtain B at last. Currently, most cryptocurrency transactions are based on AMM DEXs (automated market maker decentralized exchanges), like Uniswap. For AMM DEXs, an exchange with a large scale will cause a great slippage, which means the second user who wants to arbitrage will gain no profit [4]. To simplify the analysis, we propose the following assumption:

Assumption 1. *Each mining pool has sufficient property to propose large enough transactions to every victim smart contracts using oracle O.*

The assumption above is reasonable since the current prosperous DeFi system allows users to borrow specific crypto assets through services like Flash Loan [23]. Besides, what we mean by a transaction is "large enough" here is that the transaction will cause a great slippage. And based on Assumption 1 we can propose the following theorem:

Theorem 1. *There can be only one winner to obtain profit for the Arbitrage attack no matter success or not.*

On the one hand, Theorem 1 shows that the final arbitrage profit B could be tremendous compared to the normal transaction fee A, and rational mining pools are fully incentivized to carry out this attack. On the other hand, since there can be only one winner, some rational mining pools may give up before the oracle's output is changed (i.e. before the attack succeeds) and decide to obtain the normal transaction fee A, especially for those mining pools with less hashrate. However, even though a mining pool with less hashrate decides to stop the attack and break the union by publishing a block containing transaction t_a, the possibility of success is based on its hashrate which is also very low.

Consequently, a key question that needs to be figured out is that whether this temporary union of rational mining pools trying to manipulate the oracle's output will maintain during the whole oracle's adjustment period until the final decisive output. In the following section, we will analyze this question based on game-theoretical analysis.

5 Game theory analysis

5.1 Game model forming

Before formal analysis, we first use the following assumption to restrict the ability of mining pools:

Assumption 2. *A mining pool can only decide not to include certain transactions in the block building by itself, but will not ignore or block valid blocks containing certain transactions proposed by others.*

What assumption 2 guarantees is that mining pools can leave the attack union by publishing a new block containing the adjustment transactions. And we will not consider the situations like blockchain forking in this paper.

Assuming that there are n mining pools in the network denoted by $[n]$ and they are all rational, searching for high profits. w.l.o.g., the hashrate satisfy $P_1 > P_2 > ... > P_n$ and $\sum_{j=1}^{n} P_j = 1$. Noted that in reality, mining pools may use part of hashrate for selfish mining [20], which won't contribute to the attack. However such a situation will not have a fundamental effect on the following analysis, so we only consider the hashrate assumption mentioned above for simplicity. We also assume that these mining pools know each other's hashrate ratio:

Assumption 3. *Rational mining pools are all aware of each other's hashrate proportion.*

According to the attack model mentioned in Section 4.2, consider a malicious proposition t_m arrives in block b_0 and an adjustment transaction t_a is broadcast to the entire blockchain network at the very moment after b_0 is published (in the real scenario, there might be a longer period of time between the appearance of these two transactions), and $[n]$ rational mining pools will simultaneously notice this arbitrage opportunity. During the verification period which lasts for s blocks, if the adjustment transaction t_a is included in block $b_i, b_0 + 1 \leq b_i \leq b_0 + s$, the malicious proposition t_m will become invalid and the Arbitrage attack will fail, the mining pool which publish block b_i will be the only winner and obtain normal profit A. On the contrary, if no adjustment appeared during s blocks, then the rational mining pool that successfully mines the block with arbitrage transactions set t_{arb} in block $b_0 + s + 1$ will obtain the arbitrage profit B. And the possibility for a rational mining pool to mine a block approximately equal to its hashrate ratio P_i.

Consequently, when mining blocks between b_0 and $b_0 + s + 1$, each rational mining pool need to decide whether include the adjustment transaction t_a and t_{arb} into the next block it is mining right now (t_{arb} is only decided for block $b_0{+}s{+}1$). Therefore, we can denote the whole Arbitrage attack into a multi-stage static game, and according to Assumption 3, the game is of perfect information since mining pools know each others' hashrate ratio and the revenue for this attack. There are at most $s + 1$ static games of perfect information during the whole attack. From game 1 to game s, mining pools need to decide whether include transaction t_a in the block they are mining, and in game $s + 1$ mining pools will decide whether include arbitrage transactions set t_{arb}. The multi-stage game will end if a block with t_a is successfully mined. During the attack, after block $b_0 + i - 1$ has been added to the chain, $[n]$ rational mining pools' strategy for forming template of block $b_0 + i$, which is also game i, can be denoted as

$$T_i = (T1_i, T2_i, ..., Tn_i)$$

where

$$Tj_i = \begin{cases} Y & \text{include transaction } t_a \text{ in block } b_0 + i \\ N & \text{not include transaction } t_a \text{ in block } b_0 + i \end{cases}$$

Tj_i* represent the Nash equilibrium strategy and the utility function for each mining pool's strategy is

$$U(T_i) = (U_1(T_i), U_2(T_i), ..., U_n(T_i))$$

For simplicity, in the following part we will use i block ($1 \leq i \leq s + 1$) to represent block $b_0 + i$.

5.2 Nash equilibrium strategy

Based on the multi-stage static game of the perfect information model mentioned above, we can easily get the following conclusion

Theorem 2. *For a rational mining pool j, its strategy for block i should be the Nash equilibrium strategy for game i.*

Theorem 2 shows that it is necessary to analyze each stage game's Nash equilibrium strategy ${T_i}^*$. It is already known that the expression of the utility function of strategy for each stage game is crucial to finding the Nash equilibrium strategy. However, things are really complicated if we sequentially analyze the problem from game 1 to game $s{+}1$, since the utility function will be complex due to a large number of potential strategy combinations. Therefore, we need to find another method to obtain the Nash equilibrium strategy of each game. Intuitively, it is obvious that at block $s+1$ every rational mining pools' strategy will be the same Y, for they will have no chance to gain profit at last. Conclusively, we can learn from the idea of reverse induction and start to analyze the Nash equilibrium strategy at $s+1$ block, then infer the Nash equilibrium strategy for previous blocks.

Block $s+1$ After publishing s block, which doesn't include t_a, all rational mining pools need to decide whether include the arbitrage transactions set t_{arb} into the $s+1$ block they are building. Apparently, we can easily obtain the Nash equilibrium strategy for this final game by comparing the utility function:

Theorem 3. *The Nash equilibrium strategy for block* $s+1$ *is*

$$T_{s+1} = (\underbrace{Y, Y, ..., Y}_{n})$$

Block s Since all rational mining pools' Nash equilibrium strategies when deciding the $s+1$ block are determinate, it will be practical to obtain mining pool j's utility function for block s:

$$U_j(T_s) = \begin{cases} P_j A + P_{Ns}(j) P_j B & Tj_s = Y \\ (P_{Ns}(j) + P_j) P_j B & Tj_s = N \end{cases}$$

where P_{Ns} represent the sum of hashrate of mining pools whose strategy is N, except mining pool j, $P_{Ns} = \sum_{k \in \{k | Tk_s = N, k \neq j\}} P_k$. We use ${P_{Ns}}^*$ to represent the situation for Nash equilibrium strategy.

By comparing those two different utility functions, the Nash equilibrium strategy ${Tj_s}^*$ should be related to the ratio of two different profits and the hashrate:

$${Tj_s}^* = \begin{cases} Y & \frac{B}{A} < \frac{1}{P_j} \\ N & \frac{B}{A} > \frac{1}{P_j} \end{cases}$$

We can therefore consider Nash equilibrium strategies in three different scenarios:

Proposition 1. *When* $\frac{B}{A} > \frac{1}{P_n} > ... > \frac{1}{P_1}$, *for a random mining pool* j, ${Tj_s}^* = N$, ${T_s}^* = (\underbrace{N, N, ..., N}_{n})$ *and* $U_j({Tj_s}^*) = P_j B$

Proposition 2. *When* $\frac{B}{A} < \frac{1}{P_1} < ... < \frac{1}{P_n}$, *for a random mining pool* j, ${Tj_s}^* = Y$, $T_s^* = (\underbrace{Y, Y, ..., Y}_{n})$ *and* $U_j({Tj_s}^*) = P_j A$

Proposition 3. *When $\frac{1}{P_1} < \frac{B}{A} < \frac{1}{P_n}$, w.l.o.g. $\frac{1}{P_1} < ... < \frac{1}{P_R} < \frac{B}{A} < \frac{1}{P_{R+1}} < ... < \frac{1}{P_n}$. Then for a random mining pool j, Nash equilibrium strategy should be*

$$Tj_s{}^* = \begin{cases} Y & j \geq R+1 \\ N & j \leq R \end{cases}$$

and the utility function should be:

$$U_j(Tj_s{}^*) = \begin{cases} P_jA + P_{Ns}(j)^* P_jB & Tj_s^* = Y \\ (P_{Ns}(j)^* + P_j)P_jB & Tj_s^* = N \end{cases}$$

Propositions above show that the ratio of two different kinds of profit will influence the Nash equilibrium strategies for game s. If B greatly exceeds the normal transaction fee profit A, then more mining pools will persist in the attack, which fits our intuition. Based on the analysis of $s+1$ and s blocks, we can extend our analysis method to the more general scenarios like the Nash equilibrium strategies for i $(0 \leq i < s)$ block.

Block *i* Similar to s block, a random mining pool j's utility function at i block should be related to its Nash equilibrium utility function in $i+1$ block:

$$U_j(T_i) = \begin{cases} P_jA + P_{Ni}(j)U_j(Tj_{i+1}{}^*) & Tj_i = Y \\ (P_{Ni}(j) + P_j)U_j(Tj_{i+1}{}^*) & Tj_i = N \end{cases}$$

and therefore the Nash equilibrium strategy should be

$$Tj_i{}^* = \begin{cases} Y & U_j(Tj_{i+1}{}^*) < A \\ N & U_j(Tj_{i+1}{}^*) > A \end{cases}$$

Similar to the analysis for the s block, we also consider the Nash equilibrium strategies in three different scenarios.

Proposition 4. *When $\frac{B}{A} > \frac{1}{P_n} > ... > \frac{1}{P_1}$, for a random mining pool j, $U_j(Tj_s{}^*) = P_jB > A$holds. By mathematical induction, the Nash equilibrium strategy for game i should be $T_i{}^* = (\underbrace{N, N, ..., N}_{n})$*

Proposition 5. *When $\frac{B}{A} < \frac{1}{P_1} < ... < \frac{1}{P_n}$, for a random mining pool j, $U_j(Tj_s{}^*) = P_jB < A$holds. By mathematical induction, the Nash equilibrium strategy for game i should be $T_i{}^* = (\underbrace{Y, Y, ..., Y}_{n})$*

When $\frac{1}{P_1} < \frac{B}{A} < \frac{1}{P_n}$, divergence will appear in the attack union compared to the two other conditions mentioned above, it is worthwhile to figure out the change of Nash equilibrium strategies of each mining pool during the whole attack process.

In order to depict the change of Nash equilibrium strategy for different stage game, here we also assume that $\frac{1}{P_1} < ... < \frac{1}{P_R} < \frac{B}{A} < \frac{1}{P_{R+1}} < ... < \frac{1}{P_n}$, we first give following theorem :

Theorem 4. *For a random mining pool j, if its Nash equilibrium strategy for block $i+1$ is Y, then the Nash equilibrium strategy for block i is also Y.*

From Theorem 4 we can conclude that

Corollary 1. *For a mining pool j, it is impossible that the Nash equilibrium strategy for game i is N and Y for game $i+1$.*

Corollary 2. *$P_{Ni}(j)^*$ should increase with i increase to s*

Have these corollaries in mind, we only need to consider mining pool j's Nash equilibrium strategy for game i when the equilibrium strategy for game $i+1$ is N.

Theorem 5. *For a mining pool j, if its Nash equilibrium strategy for game $i+1$ is N and Y for j. Then for any mining pool h with $P_h < P_j$, there should be ${Th_i}^* = Y$.*

From Theorem 5 we can get the following results:

Corollary 3. *With block number i increases, mining pools with more hashrate will change their strategy from Y to N more sooner than mining pools with less hashrate.*

Corollary 4. *There won't be a Nash equilibrium strategy ${T_i}^*$ for game i where ${Th_i}^* = N, {Tj_i}^* = Y$ $(P_h < P_j)$.*

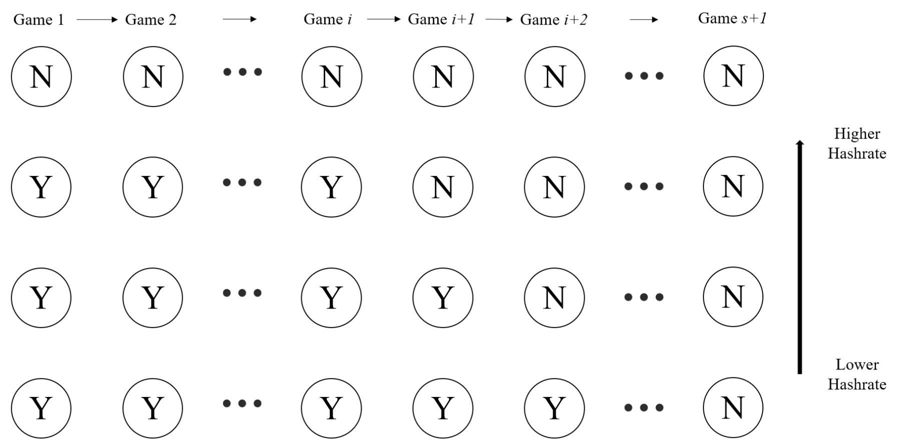

Fig. 3. Example of four different mining pool's Nash equilibrium strategies' changes during the whole attack.

Combine Theorem 4 and 5 as well as corollaries above we can depict the change of Nash equilibrium strategy for mining pool j at game i $(0 \leq i \leq s)$ when $\frac{1}{P_1} < \frac{B}{A} < \frac{1}{P_n}$, Fig.3 shows an example with four different mining pools:

- If j's Nash equilibrium strategy is Y at game $i+1$, for any other game $g, g < i+1$, mining pool j's Nash equilibrium strategy will also be Y.
- If j's Nash equilibrium strategy is N at game $i+1$. Then Nash equilibrium strategy at game i will be Y if $P_j B \prod_{k=i}^{s}(P_N k(j) + P_j) < A$, else the Nash equilibrium strategy will be N.

With the Nash equilibrium strategy change under different circumstances mentioned above, each mining pool is able to decide their strategies for each block they are building with the attack proceeding. The whole process for a rational mining pool to decide the final strategy is described as an algorithm pseudo-code in Appendix B.

5.3 Price of Anarchy

The ratio between the worst Nash equilibrium and the social optimum is the price of anarchy (PoA), the formal definition can be denoted as:

$$PoA = \frac{min_{t \in N} profit(t)}{max_{t \in T_{all}} profit(t)}$$

where N is the set of Nash equilibrium strategies and T_{all} is the set of all possible strategies during the process of attack. $profit(t)$ represents the expected profit for the union of rational mining pools when the attack ends, no matter successful or not.

The price of anarchy provides an insight into the effects of lack of corporation. To be more specific, PoA represents the gap between system performance when players all behave selfishly and follow central coordination. When the price of anarchy is close to 1, selfish players don't severely influence the union's total profit, which means the union is stable. In contrast, the low price of anarchy shows that the union is loose.

Since the ratio of two kinds of profit B and A will influence the Nash equilibrium strategies, we can determine the price of anarchy in different situations.

Corollary 5. *When $\frac{B}{A} > \frac{1}{P_n} > ... > \frac{1}{P_1}$, the price of anarchy is $PoA = 1$.*

Corollary 6. *When $\frac{B}{A} < \frac{1}{P_1} < ... < \frac{1}{P_n}$, the price of anarchy is $PoA = \frac{A}{B}$.*

Corollary 7. *When $\frac{1}{P_1} < \frac{B}{A} < \frac{1}{P_n}$, the price of anarchy is*

$$PoA = \frac{(1 - \prod_{i=1}^{s} P_{Ni})A + \prod_{i=1}^{s} P_{Ni} B}{B}.$$

6 Conclusion, Limitations and Extensions

In this paper, we introduce an Arbitrage attack against decentralized oracles carried out by rational mining pools. We show that when potential arbitrage profit weights far more than regular profits, different mining pools will take the same actions to delay or block transactions like a temporary union to carry on the attack. And as the attack progresses, the union becomes more and more stable. In the parts below, we will further discuss the limitations and possible extensions of this paper.

Generality. As mentioned before, the attack discussed in this paper is against decentralized oracles that depend on transactions to make adjustments and verification, therefore a very promising topic to discuss is whether this attack will be feasible to a more general decentralized oracle model.

And in another aspect, this attack may be feasible to not only decentralized oracles. To be more specific, similar to decentralized oracles, many other blockchain applications also realized certain functions based on proposing different transactions. Therefore, we believe that the temporary union in Arbitrage attack may happen in other blockchain applications. For instance, consider a DAO managing the parameters of a transaction pool of two kinds of digital assets like Curve [19], members of DAO are supposed to vote to decide the specific parameters like the exchange rate of these two assets for the next several days. Assume that all members need to vote by proposing transactions to DAO's official voting account, and the voting period will last for several blocks for example. In that case, the rational mining pools may take the same actions in Arbitrage attacks by blocking or delaying voting transactions against their will during the voting period, to manipulate the final decision. Consequently, although efficient incentive mechanisms are applied to guarantee DAO's rational members to vote honestly, the final decision still can be influenced because of the power of mining pools.

Game theory analysis. The game theory analysis in this paper can become more complete. For one thing, in this paper, we only depict the possible process during the attack when the two revenue A and B are stable. However, in reality, the revenue A and B could be dynamic, since the number of adjustment transactions could increase which will makes A increase during the process. By considering changes of A may help to depict the whole attack process more dynamically. Similarly, the attack revenue B could also change due to the price fluctuation of the digital assets associated with the attack. For the other, even a mining pool becomes the final winner of the attack, the negative social influence of the attack may also cause the winner a great loss. Such a potential loss may exert influence on mining pools' strategy during the attack. However, such kind of loss is difficult to describe, which also makes the game theory analysis pretty challenging.

Forking. In Assumption 2, we make some restrictions about mining pools' ability since the attack is closely related to the forking problem. Without the restriction, mining pools that wish to continue the attack can choose to fork and ignore the block containing adjustment transactions published by mining pool that chooses to give up the attack. Besides, at the end of the attack, loser mining pools can even call for forking by claiming that the only winner is the "evil attacker" in order to make the winner unable to obtain the revenue. These kinds of problems are complicated but deserve further discussion.

Mitigation methods. Detailed mitigation methods are not given in this paper, and we plan to do further exploration about feasible solutions to this attack. So far, we believe there may be two practical ways to solve the problem. One is to adjust the time of the verification process s. Intuitively, longer s will make the attack less likely to succeed. But longer verification time will also make the oracle's outputs less time-sensitive, which could be a deadly drawback for oracles designed for scenarios like the high-frequency trading market. Therefore the optimal verification time or period can only be obtained by detailed analysis. The other possible solution is to bring in cryptography methods like "secret ballots" [32] in the internet voting system. The mining pools will be unable to distinguish transactions with adjustments information if transactions' contents are encrypted. But the detailed procedure needs to be well designed to ensure safety.

References

1. Z. Zheng, S. Xie, H.-N. Dai, W. Chen, X. Chen, J. Weng, and M. Imran, "An overview on smart contracts: Challenges, advances and platforms," *Future Generation Computer Systems*, vol. 105, pp. 475–491, 2020.
2. C. Finance, "Compound finance," `https://compound.finance/`, 2021.
3. bZx network, "bzx network," `https://bzx.network/`, 2021.
4. L. Zhou, K. Qin, C. F. Torres, D. V. Le, and A. Gervais, "High-frequency trading on decentralized on-chain exchanges," *arXiv preprint arXiv:2009.14021*, 2020.
5. M. Apostolaki, A. Zohar, and L. Vanbever, "Hijacking bitcoin: Routing attacks on cryptocurrencies," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 375–392.
6. H. Shi, S. Wang, Q. Hu, X. Cheng, J. Zhang, and J. Yu, "Fee-free pooled mining for countering pool-hopping attack in blockchain," *IEEE Transactions on Dependable and Secure Computing*, 2020.
7. A. M. Antonopoulos, "Mastering bitcoin," 2019.
8. F. Zhang, E. Cecchetti, K. Croman, A. Juels, and E. Shi, "Town crier: An authenticated data feed for smart contracts," in *Proceedings of the 2016 aCM sIGSAC conference on computer and communications security*, 2016, pp. 270–282.
9. K. Yamashita, Y. Nomura, E. Zhou, B. Pi, and S. Jun, "Potential risks of hyperledger fabric smart contracts," in *2019 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*. IEEE, 2019, pp. 1–10.
10. H. Al-Breiki, M. H. U. Rehman, K. Salah, and D. Svetinovic, "Trustworthy blockchain oracles: Review, comparison, and open research challenges," *IEEE Access*, vol. 8, pp. 85 675–85 685, 2020.
11. X. Xu, C. Pautasso, L. Zhu, V. Gramoli, A. Ponomarev, A. B. Tran, and S. Chen, "The blockchain as a software connector," in *2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA)*. IEEE, 2016, pp. 182–191.

12. H. Moudoud, S. Cherkaoui, and L. Khoukhi, "An iot blockchain architecture using oracles and smart contracts: The use-case of a food supply chain," in *2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. IEEE, 2019, pp. 1–6.
13. G. Cloud, "Building hybrid blockchain/cloud applications with ethereum and google cloud," `https://cloud.google.com/blog/products/data-analytics/building-hybrid-blockchain-cloud-applications-with-ethereum-and-google-cloud`, 2021.
14. J. Adler, R. Berryhill, A. Veneris, Z. Poulos, N. Veira, and A. Kastania, "Astraea: A decentralized blockchain oracle," in *2018 IEEE international conference on internet of things (iThings) and IEEE green computing and communications (GreenCom) and IEEE cyber, physical and social computing (CPSCom) and IEEE smart data (SmartData)*. IEEE, 2018, pp. 1145–1152.
15. J. Peterson, J. Krug, M. Zoltu, A. K. Williams, and S. Alexander, "Augur: a decentralized oracle and prediction market platform," *arXiv preprint arXiv:1501.01042*, 2015.
16. A. S. de Pedro, D. Levi, and L. I. Cuende, "Witnet: A decentralized oracle network protocol," *arXiv preprint arXiv:1711.09756*, 2017.
17. S. Wang, W. Ding, J. Li, Y. Yuan, L. Ouyang, and F.-Y. Wang, "Decentralized autonomous organizations: concept, model, and applications," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 5, pp. 870–878, 2019.
18. N. Protocol, "the NEST protocol," `https://nestprotocol.org/`, 2021.
19. C. Fiance, "Curve finance," `https://curve.fi/`, 2021.
20. I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," in *International conference on financial cryptography and data security*. Springer, 2014, pp. 436–454.
21. L. Zhou, K. Qin, C. Ferreira Torres, A. Gervais *et al.*, "High-frequency trading on decentralized on-chain exchanges," in *IEEE Symposium on Security and Privacy, 23-27 May 2021*, 2021.
22. S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Business Review*, p. 21260, 2008.
23. D. Wang, S. Wu, Z. Lin, L. Wu, X. Yuan, Y. Zhou, H. Wang, and K. Ren, "Towards a first step to understand flash loan and its applications in defi ecosystem," in *Proceedings of the Ninth International Workshop on Security in Blockchain and Cloud Computing*, 2021, pp. 23–28.
24. K. Liao and J. Katz, "Incentivizing blockchain forks via whale transactions," in *International Conference on Financial Cryptography and Data Security*. Springer, 2017, pp. 264–279.
25. P. McCorry, A. Hicks, and S. Meiklejohn, "Smart contracts for bribing miners," in *International Conference on Financial Cryptography and Data Security*. Springer, 2018, pp. 3–18.
26. P. Daian, S. Goldfeder, T. Kell, Y. Li, X. Zhao, I. Bentov, L. Breidenbach, and A. Juels, "Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 910–927.
27. S. Eskandari, M. Moosavi, and J. Clark, "Sok: Transparent dishonesty: front-running attacks on blockchain," 2019.
28. U. W. Chohan, "Counter-hegemonic finance: The gamestop short squeeze," *Available at SSRN*, 2021.
29. provable, "Provable documentation," `https://docs.provable.xyz`, 2021.

30. S. Eskandari, J. Clark, V. Sundaresan, and M. Adham, "On the feasibility of decentralized derivatives markets," in *International Conference on Financial Cryptography and Data Security.* Springer, 2017, pp. 553–567.
31. A. Anand and J. Pathak, "Wallstreetbets against wall street: The role of reddit in the gamestop short squeeze," *IIM Bangalore Research Paper*, no. 644, 2021.
32. H. Wu, P. L. Vora, and F. Zagórski, "Privapollo-secret ballot e2e-v internet voting." in *Financial Cryptography Workshops*, 2019, pp. 299–313.

A Proofs

Theorem 1. *There can be only one winner for the Arbitrage attack no matter success or not.*

Proof. In the case of failure, since there can be only one block containing adjustment transaction t_a, therefore this block's publisher should be the only winner with normal profit A.

In the case of success, each rational mining pool can publish a smart contract in the blockchain including all the transaction it would like to make after the oracle's output is manipulated. This smart contract has enough time to become valid during the oracle's verification period. After the output has been changed, the mining pool can propose a newly mined block containing arbitrage transactions set t_{arb} and exploit all arbitrage opportunities. □

Theorem 2. *For a rational mining pool j, its strategy for block i should be the Nash equilibrium strategy for game i.*

Proof. The Nash equilibrium strategy at game i for mining pool j should satisfy

$$U_1({T_i}^*) \geq U_1(\overline{{T_i}^*})$$

where ${T_i}^*$ represents the equilibrium strategy and $\overline{{T_i}^*}$ represents other strategies. Thus a rational mining pool will choose the equilibrium strategy while making decision for block i. □

Theorem 3. *The Nash equilibrium strategy for block $s+1$ is*

$$T_{s+1} = (\underbrace{Y, Y, ..., Y}_{n})$$

Proof. According to the attack model, all the rational mining pools will try to mine their own block containing the arbitrage transactions set t_{arb}, so the utility function for a mining pool j can be easily denoted as:

$$U_j(T_{s+1}) = \begin{cases} P_j B & Tj_{s+1} = Y \\ 0 & Tj_{s+1} = N \end{cases}$$

Where P_j is the hashrate ratio of mining pool j. Therefore, all the rational mining pools will choose to add the transaction to the block. □

Proposition 1. *When* $\frac{B}{A} > \frac{1}{P_n} > ... > \frac{1}{P_1}$*, for a random mining pool* j*,* $Tj_s{}^* = N$*,* $T_s{}^* = (\underbrace{N, N, ..., N}_{n})$ *and* $U_j(Tj_s{}^*) = P_jB$

Proof. For all rational mining pools that participated in the attack, the profit ratio $\frac{B}{A}$ is larger than the reciprocal of any mining pool's hashrate. As a result, according to the utility function of two different actions, every mining pool's Nash equilibrium strategy will be N, which will guarantee the attack will succeed and game $s+1$ will be conducted, therefore the utility will be P_jB for a random mining pool j. □

Proposition 2. *When* $\frac{B}{A} < \frac{1}{P_1} < ... < \frac{1}{P_n}$*, for a random mining pool* j*,* $Tj_s{}^* = Y$*,* $T_s^* = (\underbrace{Y, Y, ..., Y}_{n})$ *and* $U_j(Tj_s{}^*) = P_jA$

Proof. For all rational mining pools that participated in the attack, the profit ratio $\frac{B}{A}$ is less than the reciprocal of any mining pool's hashrate. Similarly, according to the utility function of two different actions, every mining pool's Nash equilibrium strategy will be Y, which will guarantee the attack will fail and game $s+1$ will be not conducted, therefore the utility will be P_jA for a random mining pool j. □

Proposition 3. *When* $\frac{1}{P_1} < \frac{B}{A} < \frac{1}{P_n}$*, w.l.o.g.* $\frac{1}{P_1} < ... < \frac{1}{P_R} < \frac{B}{A} < \frac{1}{P_{R+1}} < ... < \frac{1}{P_n}$*. Then for a random mining pool* j*, Nash equilibrium strategy should be*

$$Tj_s{}^* = \begin{cases} Y & j \geq R+1 \\ N & j \leq R \end{cases}$$

and the utility function should be:

$$U_j(Tj_s{}^*) = \begin{cases} P_jA + P_{Ns}(j)^* P_jB & Tj_s^* = Y \\ (P_{Ns}(j)^* + P_j)P_jB & Tj_s^* = N \end{cases}$$

Proof. According to the utility function, mining pools whose hashrate ratio's reciprocal is higher than $\frac{B}{A}$ will choose Y and N for the rest of the mining pools, and the utility can be easily obtained based on their Nash equilibrium strategies. □

Proposition 4. *When* $\frac{B}{A} > \frac{1}{P_n} > ... > \frac{1}{P_1}$*, for a random mining pool* j*,* $U_j(Tj_s{}^*) = P_jB > A$*holds. By mathematical induction, the Nash equilibrium strategy for game* i *should be* $T_i{}^* = (\underbrace{N, N, ..., N}_{n})$

Proof. According to Theorem 1, for a random mining pool j, $U_j(Tj_s{}^*) = P_jB > A$ holds. Therefore, for game $s-1$, the Nash equilibrium strategy and corresponding utility can be easily obtained:

$$T_{s-1}{}^* = (\underbrace{N, N, ..., N}_{n})$$

$$U_j(Tj_{s-1}{}^*) = P_jB > A$$

Then by mathematical induction, rational mining pools' Nash equilibrium strategy for every game should be $(\underbrace{N, N, ..., N}_{n})$. □

Proposition 5. *When $\frac{B}{A} < \frac{1}{P_1} < ... < \frac{1}{P_n}$, for a random mining pool j, $U_j(Tj_s{}^*) = P_jB < A$ holds. By mathematical induction, the Nash equilibrium strategy for game i should be $T_i{}^* = (\underbrace{Y, Y, ..., Y}_{n})$*

Proof. According to Theorem 2, for a random mining pool j, $U_j(Tj_s{}^*) = P_jA < A$ holds. Therefore, for game $s-1$, the Nash equilibrium strategy and corresponding utility can be easily obtained:

$$T_{s-1}{}^* = (\underbrace{Y, Y, ..., Y}_{n})$$
$$U_j(Tj_{s-1}{}^*) = P_jA < A$$

Then by mathematical induction, rational mining pools' Nash equilibrium strategy for every game should be $(\underbrace{Y, Y, ..., Y}_{n})$. □

Theorem 4. *For a random mining pool j, if its Nash equilibrium strategy for block $i+1$ is Y, then the Nash equilibrium strategy for block i is also Y.*

Proof. Assuming that mining pool j's Nash equilibrium strategy for block $i+1$ is Y, then

$$U_j(T^*_{i+2}) < A$$

Therefore

$$\begin{aligned}
U_j(Tj_{i+2}{}^*) &< A \\
U_j(Tj_{i+1}{}^*) &= P_jA + P_{N(i+1)}(j)^*U_j(Tj_{i+2}{}^*) \\
&< (P_j + P_{N(i+1)}(j)^*)A \leq A \\
\Rightarrow U_j(Tj_{i+1}{}^*) &< A
\end{aligned}$$

The Theorem is therefore proved. □

Corollary 1. *For a mining pool j, it is impossible that the Nash equilibrium strategy for game i is N and Y for game $i+1$.*

Proof. According to Theorem 4, a mining pool's Nash equilibrium strategy for game i can only be Y, if its Nash equilibrium strategy for game $i+1$ is Y. □

Corollary 2. *$P_{Ni}(j)^*$ will not decrease with i increase to s*

Proof. According to Corollary , once a mining pool's Nash equilibrium strategy is N for a game, then its Nash equilibrium strategy will not change to Y in later games. Instead, it is possible for mining pools with Y as Nash equilibrium strategy to change in later games. Therefore, the total hashrate of mining pools with Nash equilibrium strategy N will not decrease with the process of the whole attack. □

Theorem 5. *For a mining pool j, if its Nash equilibrium strategy for game $i+1$ is N and Y for i. Then for any mining pool h with $P_h < P_j$, there should be $Th_i{}^* = Y$.*

Proof. We can prove the theorem with contradiction. Assuming that there is a mining pool h with $P_h < P_j$, and its Nash equilibrium strategy for game i is N.

According to Theorem 4, mining pool h's Nash equilibrium strategy for game $i+1$ should be N, therefore $U_h(Th_{i+1}{}^*) = P_h b \prod_{k=i+1}^{s}(P_N k(h) + P_h)$ should be greater than A

$$U_h(Th_{i+1}{}^*) > A$$

However, for mining pool j

$$U_j(T^*_{i+1}) = P_j B \prod_{k=i+1}^{s} (P_N k(j) + P_j) < A$$

since the Nash equilibrium strategy changes to Y. Notice that

$$U_h(Th_{i+1}{}^*) < U_j(T^*_{i+1})$$

because $P_h < P_j$. Then a contradiction happens. □

Corollary 3. *With block number i increases, mining pools with more hashrate will change their strategy from Y to N more sooner than mining pools with less hashrate.*

Proof. According to Theorem 5, if a mining pool j's Nash equilibrium strategy is N for game $i+1$ and Y for i, then in game i the Nash equilibrium strategy for all the mining pools with less hashrate will also be Y, which will not change for the game before i according to Theorem 4. Consequently, with block number i increase, mining pool j's Nash equilibrium strategy will change from Y to N before the mining pools with less hashrate. □

Corollary 4. *There won't be a Nash equilibrium strategy $T_i{}^*$ for game i where $Th_i{}^* = N, Tj_i{}^* = Y(P_h < P_j)$.*

Proof. Corollary 3 shows that with block number i increases, mining pools with higher hashrate will change its strategy from Y to N sooner, besides it is not possible to change from N to Y. Conclusively, there won't be a Nash equilibrium strategy $T_i{}^*$ for game i where $Th_i{}^* = N, Tj_i{}^* = Y(P_h < P_j)$. □

Corollary 5. *When $\frac{B}{A} > \frac{1}{P_n} > ... > \frac{1}{P_1}$, the price of anarchy is $PoA = 1$.*

Proof. According to Proposition 4, all rational mining pools will choose N during the whole s blocks period, then

$$\begin{aligned} PoA &= \frac{profit(N \text{for all})}{profit(N \text{for all}} \\ &= \frac{B}{B} \\ &= 1 \end{aligned}$$

□

Corollary 6. *When $\frac{B}{A} < \frac{1}{P_1} < ... < \frac{1}{P_n}$, the price of anarchy is $PoA = \frac{A}{B}$.*

Proof. According to Proposition 5, all rational mining pools will choose Y during the whole s blocks period, then

$$\begin{aligned} PoA &= \frac{profit(Y \text{for all})}{profit(N \text{for all})} \\ &= \frac{A}{B} \end{aligned}$$

□

Corollary 7. *When $\frac{1}{P_1} < \frac{B}{A} < \frac{1}{P_n}$, the price of anarchy is*

$$PoA = \frac{(1 - \prod_{i=1}^{s} P_{Ni})A + \prod_{i=1}^{s} P_{Ni}B}{B}.$$

Proof. When $\frac{1}{P_1} < \frac{B}{A} < \frac{1}{P_n}$, since there will always be mining pools that decide to publish the adjustment contract t_a during the s blocks and gain profit A, the only situation to obtain profit B is when mining pools whose Nash equilibrium strategy is N successfully mine the block □

B Algorithm

Algorithm 1 Mining pool q's Nash equilibrium strategy for block k

```
Input: Pool q's hashrate ratio $P_q$, block number for verification period $s$, mining
    pools' total number $n$, target block number $k$, profit $A, B$, mining pools' hashrate
    sequence $P = (P_1, P_2, P_3 ... P_n)$  //in descending order
Output: Strategy $T$  //$T = Y$ represents including adjustment transaction in block $k$
    template, $T = N$ represents not including.
 1: ArrayN=[ ]; //array for mining pools with strategy N
 2: ArrayT=[$Y, Y ... Y$]; //strategy array
 3: for $j = 1, i = 1; j \leq n; j++$ do
 4:     if $\frac{1}{P_j} < \frac{B}{A}$ then
 5:         ArrayN[i++]=$P_j$;
 6:         ArrayT[j]=$N$;
 7:     end if
 8: end for
 9: if len(ArrayN)==n then //Nash equilibrium strategy for every block is N
10:     return $N$;
11: end if
12: if len(ArrayN)==0 then //Nash equilibrium strategy for every block is Y
13:     return $Y$;
14: end if
15: if q $\notin$ ArrayN then //Nash equilibrium strategy for block $s$ is $Y$, return $Y$
16:     return $Y$;
17: end if
18: for $i = s - 1; i \geq k + 1; i--$ do
19:     COMPARE($A$,$U(Tq^*_{i+1})$) //compare the utility with $A$ to decide final strategy
20:     if $A$ is larger then
21:         return $Y$
22:     end if
23:     for $q \in$ ArrayN do
24:         if $A > U(Tj_{i+1}{}^*)$ then
25:             POP(ArrayN,j) //delete mining pools whose strategy changes
26:         end if
27:     end for
28:     $U(Tq^*_i)$=SUM(ArrayN)*$U(Tq^*_{i+1})$
29: end for
30: return $N$
```