IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received May 8, 2022, accepted May 24, 2022, date of publication June 8, 2022, date of current version June 14, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3181186

Exploration of Mined Block Temporarily Holding and Enforce Fork Attacks by Selfish Mining Pool in Proof-of-Work Blockchain Systems

YEAN-FU WEN, (Member, IEEE), AND CHUN-YU HUANG
Graduate Institute of Information Management, National Taipei University, New Taipei City 237303, Taiwan

Corresponding author: Yean-Fu Wen (yeanfu@mail.ntpu.edu.tw)

This work was supported in part by the Ministry of Science and Technology (MOST), Taiwan, under Grant 107-2221-E-305-003-MY2 and University System of Taipei, Taiwan, under Grant USTP-NTUT-NTPU-111-02.

ABSTRACT We explored mined block temporarily holding (MBTH), MBTH with enforce fork (MBTH-EF), and intermittent MBTH-EF (iMBTH-EF) attacks to understand the effect of selfish mining on winning rate and fairness from the pool operation perspective. The temporary holding time of the current mined block provides a pool additional time to begin mining the next block early with MBTH attack. In this situation, the winning probability increases for the next block, but the pool may risk losing the mined block. One enhanced method to maintain the winning probability of the held block is to ensure the holding pool sends out the mined block once it receives the mined block message from the other pools with the MBTH-EF attack. The explored MBTH attacks differ from the existing selfish miner and pool attacks. The operations and effects of the MBTH-EF attack are different from stubborn mining strategies and a self-holding attack integrates selfish and stubborn mining attacks. We propose a mining competition solution that does not involve actual hash calculation. It entails using one stochastic target hash value for batch racing simulation to evaluate the holding threshold, holding periods, mining time, mining difficulty, pool sizes, and the rate of fork occurrences according to the operation data of the Bitcoin system. Accordingly, the dynamic time-by-time, block-by-block, and pool-by-pool simulations are adopted to study these attacks. We analyzed MBTH and MBTH-EF attacks as well as evaluated the effects of the win rate on when and how long a block is held. Because the periodic adjustment of mining difficulty reduces the holding effect, we further evaluated how an intermittent MBTH-EF (iMBTH-EF) attack model balance the average mining time and mining difficulty according to the mining difficulty adjustment of a stage. The effect of intermittent holding is examined on the mining game win rate for a long-term competition. We also identified suitable attack detection methods for the future work according to the simulation results.

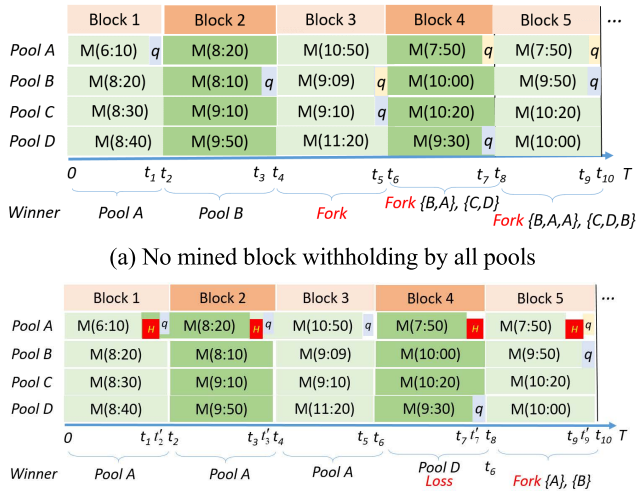
INDEX TERMS Attack model, blockchain, consensus, fork, selfish mining, simulation, temporarily holding.

I. INTRODUCTION

In pool-age blockchain mining, the pool operator aggregates mining results from individual miners by using a collaborative strategy [1]–[3]. The size of a pool is defined by the aggregated computing power required to generate the number of hash values per second (denoted as hash rate). A large pool generates values with a high hash generation rate; conversely, a small pool generates values with a low hash generation rate. According to the historical operation records of the Bitcoin system [4], the hash rate of the largest pool, which exceeds 18.4% of the entire network, is approximately 1.5 times higher than that of the second-largest pool (12.0%) [5]. If two or more pools cooperate as an alliance, their shared computing power might be twice that of any other pool. A high hash generation rate allows pools to adopt mined block temporary holding (MBTH), MBTH with enforce fork (MBTH-EF), and advanced intermittent MBTH-EF (iMBTH-EF) attacks to increase their win rate and revenue share. Thus, the present study analyzed the effects of the pool size, holding threshold, holding time, and fork occurrences on the win rate under MBTH, MBTH-EF, and iMBTH-EF attacks.

The main purpose of an MBTH attack by a pool is to use the partial mining time of the current competition period to search for the target hash value of the next block. The attack

The associate editor coordinating the review of this manuscript and approving it for publication was Jesus Felez.



(a) No mined block withholding by all pools

(b) Mined block withholding by pool A

FIGURE 1. Comparison between mining processes with and without an MBTH attack.

pool takes the risk of holding a mined block for it to be capable of directly mining the next block with the mined target hash value and obtaining additional mining time for the next block. A longer additional mining time leads to a higher win probability for the next block. The holding of a block by a pool affects the mining competition in the following several blocks. Fig. 1 illustrates an example of an MBTH attack executed by Pool A that compares normal mining processes. Block 2 is won by Pool A because this pool has additional mining time for this block than do other pools. However, Block 4 is mined by Pool D during the holding time; thus, Pool A incurs some risk to lose Block 4. In addition, Pool A withholds Block 5; however, Pool B mines a block at the same depth as Block 5 when the holding period is completed. Therefore, a pool incurs some risk when it withholds a mined block for a certain period because other contestants (miners or pools) might mine a block at the same depth within the holding time. The incentive of a mined block might be lost if other pools successfully mine the block during the holding period.

The main driving forces of the mining competition are opportunity, computing power, and a block that broadcasts all miners as the new head of the blockchain. The principal elements of the mining competition are (1) transactions that are generated and broadcast to all the miners, which collect the transactions into a local data structure known as a block; (2) the computing power at the disposal of the miners; and (3) a protocol for extending the blockchain at every miner such that the blockchains at all the miners are in consensus. The opportunity factor refers to a set of transactions with a field of nonce values that can be used to determine a hash value. Each block has a block header whose fields provide input to the crypto puzzle.

The term "mining" or "mine" in this work is involved the repeated computation of the following double SHA-256 hash function by a miner: $s = \text{SHA256}(\text{SHA256}(n + h + s' + Target + Timestamp))$, where the concatenation of strings with symbol '+' that includes a random nonce value n, the Merkle root of the transactions contained in the block body h, the previous block target hash value s', *Target*, and timestamp. If a miner firstly computes the hash values, which is small than Target, the miner is said to have mined a block [6]–[10]. Otherwise, the nonce value is incremented and recomputed until a target hash value is found for a block. In the pool concept, a miner has mined a block that means the belonging pool has mined a block. The hash value range provides the opportunity to have tens of exponent values such that a target hash value [which is smaller than the mining difficulty threshold for proof-of-work (PoW) systems] can be mined to win a block mining game. The opportunity factor allows a small pool with low computing power to determine the target hash value in a few seconds, whereas a large pool with high computing power might take a long time to search the target hash value. Given the stochastic nature of the mining game, any pool can randomly win the mining game on the basis of luck. A pool might not win the next block competition even when an MBTH attack is executed. Furthermore, larger number of pools compete the mining game results in larger opportunity probability to win the game among pools so that an MBTH attack is profitable compared to the honest mining.

The MBTH attack differs from the withholding attack [10]–[20]. The MBTH attack enables a mining pool to mine the next block directly according to its mined target hash value and the time spent mining the current block. During the holding period, other pools are still mining the already mimed block. Although an MBTH attack aims to set the holding time to be shorter than the time required by other pools to determine the target hash values, the crypto puzzle implies that the probability of mining a block is geometrically distributed. However, we conducted a simulation of the mining competition related to the computing power and opportunity to determine the winning rate. The stochastic nature of the searching time of contestants affects whether the attack pool decides to hold the mined block, as described in Section IV-C.

An MBTH attack should be adopted by a large pool; however, this attack does not significantly increase the win rate for large pools because setting the holding period is difficult. If the holding period is excessively long, the mined block might be lost; however, if the holding period is too short, a win cannot be ensured in the mining game for the next block. MBTH attacks do not provide a considerable advantage to the holding pool because of the stochastic nature of the mining process; thus, the effect of these attacks is limited.

A large pool with high computing power can calculate higher numbers of hash and nonce values per unit time than can a small pool with low computing power [21]. Over a long-term period, a pool with higher computing power has a higher win rate and, on average, requires a shorter time to mine a block [5]. Therefore, the risk of a pool losing a withheld block

decreases if one or more other pools discover a block at the same depth as the withheld block. In this situation, the attack pool sends out the mined block by enforcing a temporary fork within the time-differential broadcast delay. A temporary fork occurs when multiple contestants mine a block approximately simultaneously [6]. The time differential of approximation caused by transmission through the peer-to-peer network that results in the occurrence of a fork is determined within the broadcast delay [22].

The blockchain determines the winners along the blocks with the longest branch chains as the competition rule. In the current version of the Bitcoin system, winners are determined by the mined blocks along the longest chain comprising at least six blocks [10], [18], [20]. However, the length of the branch chain constitutes an incentive metric that provides a higher winning probability to a larger pool. The enforce fork attack is beneficial to a large pool when the holding pool receives the mined block from any other pool. The large pool uses its advantage to compete for n blocks to win the held block back. The present study evaluated in detail the effects of the MBTH-EF attack.

In an MBTH-EF attack, which is an improved version of an MBTH attack, the loss of a mined block through a fork chain competition is avoided. The MBTH-EF attack increases the win rate of an attack pool, even when the computing resource share of the pool is less than 50%. This attack maintains a holding period, which allows an attack pool to begin mining the next block early; thus, the win probability of the pool increases. Moreover, the mined block might not be lost when a fork chain is formed; thus, the win rate of the pool increases. However, an MBTH-EF attack results in an increase in the frequency of fork occurrences. Furthermore, this attack results in an increase in the mining time; thus, the mining difficulty decreases, and contestants can mine blocks in a short time, which increases the rate of losing the mined block.

In the long term, the MBTH-EF attack causes the mining difficulty to be underestimated because of the holding times of mined blocks. The holding attack increases the average mining time such that the mining difficulty tends to decrease with continuous MBTH-EF attacks. The win rate decreased with MBTH-EF attacks after several stages of mining difficulty adjustment because these adjustments reduced the mining difficulty and shortened the average mining time. The rate of fork occurrences increased when holding and fork attacks were continually used. This result indicates that MBTH-EF attacks cannot be used to maintain mining advantages over an extended period. Therefore, we analyzed iMBTH-EF attacks in depth. These attacks are initiated periodically according to the stages of mining difficulty adjustment. We assessed the effects of iMBTH-EF attacks on the average mining time, win rate, level of mining difficulty, and frequency of fork occurrences for each stage of mining difficulty adjustment.

An attack pool decides (1) whether to launch an attack, (2) how long should the attack pool withhold a mined block while maintaining an acceptable risk of losing the reward of the withheld block, (3) the duration of the effect of early mining on the win rate for the next block, (4) whether to launch an enforce fork attack to reduce the loss rate of mined blocks, and (5) whether to launch an iMBTH-EF attack to determine the average mining time and temporary fork occurrences in each stage of mining difficulty adjustment. The pool size, mining time, mining difficulty, and rate of fork occurrences affect the already mined block lost risk and decisions of a pool regarding the holding threshold and holding period.

We propose a stochastic target with batch racing for a mined block MBTH-EF (called STBR-HF) algorithm to simulate the effects of MBTH, MBTH-EF, and iMBTH-EF attacks. STBR-HF simulation is designed by adjusting the average lotto ball drawing rate for a yellow ball (i.e., the target ball). A single yellow ball is employed because the mining space is independent of miners and pools. The mining competition is complete when the first target hash value is found. When multiple pools simultaneously mined the target hash value, the broadcast time of the mined block is controlled and a temporary fork chain is enforced. Bitcoin operation log data [5], [23] were used to evaluate the deviation between the conventional mining process and the proposed solution. These attacks are worth being simulated because they change the fairness competition, mining cost-effectiveness, and the design goal of the system operation.

This paper proposes three novel types of attacks: MBTH, MBTH-EF, and iMBTH-EF attacks. The main contributions of this study are as follows:

1) This study analyzed the effect of the MBTH time on the fairness of the mining competition and the win rate by controlling the holding threshold and holding period for the mined block.
2) This study develops an STBR-HF solution for simulating MBTH, MBTH-EF, and iMBTH-EF attacks executed by tens of pools by using records generated by the Bitcoin system.
3) This study examined the effects of adjustments in the holding threshold and holding period, fork attacks, and the pool size on the win rate.
4) This study investigated the effects of the execution of MBTH-EF and iMBTH-EF attacks by a mining pool on the mining time, mining difficulty, frequency of fork occurrences, and win rate. We also identified suitable attack detection methods.

The remainder of this paper is organized as follows. Section II presents a review of the existing relevant literature. Section III describes the problem model and simulation procedures. The simulation results and discussion are provided in Section IV. Conclusions are drawn in Section V.

II. LITERATURE REVIEW

Some studies have investigated the block withholding attack from the perspective of miners. Dishonest miners withhold information on full hash outputs—that is, the target hash

values—instead of immediately sending the mined block with target hash values to the pool operator. This action results in the formation of an unfair reward mechanism [3], [11]–[19], [24]. Block withholding is considered an attack because the miner observes the reward with their computing power share from the pool operator but does not report the target hash outputs for the operator to win the reward in the blockchain system. The effect of this attack behavior is negligible because the probability of an individual miner mining the target hash value is small. This probability is small because the mining difficulty is adjusted in pool-age blockchain mining.

Several studies have investigated the withholding problem when a pool uses its computing power to attack other pools [13]–[15], [17], [19], [20]. Probability theory has been employed to investigate this problem in the context of multiple pools. Moreover, how the attacked pools lose the mining competition has been determined. Some studies have examined the withholding attacks of a set of blocks to enforce the blockchain system in the fork mode. Selfish miners hold a branch of blocks and enforce fork attacks to win the rewards of the blocks along the longest fork chain [3], [10], [18], [20], [25]. The attacking pool possesses a sufficiently large α value in the mining network to ensure that the blocks mined by it exist in the main chain; otherwise, the win rate might be low. Typically, α should be between 1/3 and 1/2.

Qin *et al.*. evaluated the computational power used by a pool to attack another pool through block withholding. They examined attack conditions through the consideration of two-pool scenario and found that an attack was not always beneficial for the attacking pool [26]. The researchers executed a withholding attack with an almost optimal ratio of computational power for attacking to maximize the reward for the attack pool. In their solution, multiple mining pools are not required and the attacking pool holds the mined target hash values, which are belonged to the attacked pool.

Li *et al.*. proposed a method based on the detection of anomalies in the properties of block statistic data to identify selfish miners. They also detected mining cartels, which comprise miners that secretly share real-time data with each other [12]. This work mainly considered the scenario in which a pool uses the MBTH attack to reduce the fairness of the mining competition. Any pool can launch an attack that can cause the wastage of other pools' resources. Large pools have an advantage in launching attacks because they have high computing power, which results in a high probability of successfully mining a block within a short period.

To increase the frequency of block withholding attacks by selfish miners among multiple pools, Chen *et al.* proposed an allocation algorithm for mining pool computing power to increase pool revenue share. This algorithm allocates the attack pool members to the target pool and executes block withholding attacks on the other pools [16]. The researchers extended the consideration of individual miners to that of a set of miners. Lee and Kim addressed the problem of selfish miners by adding the transaction creation time to the transaction data structure [27]. The transaction creation time should be reported to all contestants in a blockchain for any pool to be capable of detecting whether a mined block is withheld. The methods proposed in these two studies are based on the concept of holding the mined block. Information on whether a mining block is held should be reported to other contestants. Motlagh *et al.* analyzed the effect of selfish mining on Bitcoin network performance [10]. By contrast, this specific type of withholding attack that considers malicious miners in mining pools, which is a completely different scenario where holding attack is launched by a mining pool. The present study focused on the MBTH attack, in which a mined target hash value is only transmitted to its pool members and temporarily held by a certain pool. The held mined block is broadcasted to other pools after the selected holding period.

A pool can launch an MBTH attack when it has mined the target hash value for the current competition period. When this attack is executed, the pool holds a mined block and does not send it to other contestants immediately so that the current competition of the mined block is temporarily incomplete. This mechanism differs from that of a selfish mining attack [28], [29] and stubborn mining [30], [31]. Malicious miners launch selfish mining attacks to gain higher block rewards than their fair share. The attacker keeps discovered blocks secret and publishes the secret chain if it has one block. The public chain catches up or it has more than one block, and the lead is reduced to 1 [28]. Selfish miners securely invalidate compliant miners' competing blocks. Over the short term, attackers with less than 25% of the computational resources can still gain from selfish mining. There is always a successful selfish mining strategy that allows the earning of higher rewards relative to honest mining, regardless of the size of the attacker [29]. However, over the long term, the computing power of malicious miners should be higher than the sum of the computing power of all other contestants. In an MBTH attack, the mined block in a given period is retained only if the attack pool discovers the block earlier than a threshold time. An advanced MBTH-EF attack controls the published time within a holding period and allows the attacker to receive a block mined at a certain depth by any other contestant. If the attack causes a fork chain, any contestants can perform mining in any fork chain and win a block from the fork chain. A fork chain with a holding block possesses high computing power and many contestant pools and has a high probability of winning fork chain competitions.

Kartik *et al.*. [30] and Liu *et al.*. [31] have investigated stubborn mining problems in which trail-stubbornness and nontrivial combinations of stubborn mining strategies are adopted to withhold or transfer blocks. The aforementioned authors have modeled the lead, equal-fork, and trail-stubbornness mining problems using Markov chains. Three parameters, namely the computing power proportion of the attacker (α), the proportion of honest miners (β), and the fraction of honest miners that mine on the attacker's

fork chain (γ), are used for modeling the aforementioned problems. The present study differs from studies on stubborn mining attacks in three ways. First, studies on these attacks have considered whether to reveal the private chain and the number of blocks that must be revealed; however, we revealed the mined block according to the configured holding period, and any pool could perform mining at a certain depth. Second, studies on stubborn mining attacks have assumed the winning rate to be a linear function of the computing power; however, we found that the winning rate is an exponential function of the computing power. Third, a high value of γ (e.g., 0.9) was adopted in the aforementioned studies to achieve a high performance with stubborn mining attacks; however, in the present study, we did not rely on γ to win the fork chain. When the computing power proportion of the attacker (α) is less than 50%, some risk is incurred to withhold one or more blocks and one expects to win the following blocks. The stubborn mining attack method cannot control the value of γ to mine the attacker's fork chain.

The proposed simulation solution differs from relevant numerical analysis and Markov chain models proposed in the literature [30]. These models exhibit limitations related to (1) the number of contestants, (2) fork chain evaluation, and (3) effect of the computing power proportion on the win rate. Our simulation solution overcomes these limitations and can be used to study the effects of various variables, namely the holding threshold, mining time, mining difficulty, holding period, pool size, and frequency of fork occurrences, according to the operation data of the Bitcoin system.

In a mining competition, a blockchain forks (splits) when multiple contestants mine a block (i.e., determine target hash values) at the same depth almost simultaneously [19]. In fork after withholding (FAW) attacks, a large attacking pool has high win rates and withholds several successively mined blocks. Moreover, an attack pool can broadcast the held several blocks to cause fork chain attacks [11], [30]. Kwon *et al.* presented a FAW attack in which a larger pool has a higher probability of eventually winning the fork chain. A small pool might be fortunate to win a game; however, a large pool has a high average win rate for several blocks [11]. The researchers only addressed cases involving two pools. By contrast, we examined the holding period under a given threshold for several pools. The holding threshold and holding period were set such that the rate of fork occurrences was relatively low and the presence of FAW attacks can thus be disguised. The delay by a miner or a pool operator in broadcasting the mined target block to all the other contestants reduced its risk of losing the current mining game and changed the fairness of the block mining process.

Self-holding attacks have been discussed in [17] and [32]. These attacks combine selfish mining and mined block withholding. Malicious miners observe the reward from the reporting block that satisfies the computing power share threshold of the pool operator and not from the mined block that satisfies the revenue threshold of the blockchain system. A malicious miner or pool withholds several mined blocks and then broadcasts them to the other contestants to cause the formation of a fork chain on imperfect PoW blockchain networks. Yang *et al.*. investigated an imperfect PoW blockchain system that is subjected to a self-holding attack. They designed a state-space model to reflect the various behaviors of malicious miners in the system. The probability of the occurrence of a three-branch fork is low; therefore, the aforementioned model is limited to two fork chains. The main differences between a self-holding attack and the attacks proposed in this paper are presented in Table 1. Moreover, the proposed simulation method differs from the state-space model, and we adopted dynamic time-by-time, block-by-block, and pool-by-pool simulations to analyze MBTH-EF attacks.

Some works extended the selfish mining to consider the revenue share effect on difficulty adjustment algorithms [24], [33], [34]. Davidson and Diamond generalized the selfish mining strategy to blockchains with variable difficulty to measure profit among various cryptocurrencies. A strategy of intermittent selfish mining is proposed to earn higher reward than that achieved through honest mining at a relatively low mining difficulty. If the mining difficulty is decreased, blocks are mined more often, which enables miners or pools to earn more Coinbase rewards. They also examined the revenue share with several difficulty adjustment algorithms used for several cryptocurrencies.

The mining difficulty is adjusted according to the total computing power, which affects the block generation cycle time [4], [35]. Zhang *et al.* analyzed the factors that influence the mining difficulty and formulated a framework for difficulty adjustment [36]. Kraft developed a method for varying the difficulty [37]. Higher computing power results in higher mining difficulty, which in turn necessitates the use of higher computing resources on average to mine a block. The block generation cycle time is not related to the mining time. The present study analyzed the effect of mining difficulty adjustment on the average mining time, win rate, and fork occurrences caused by MBTH-EF attacks. Thus, we advanced analyzes the effect on win rate caused by iMBTH-EF attacks.

Table 1 presents a comparison between the present study and previous state-of-the-art studies. This study fundamentally differs from those in the literature. For example, the ''withhold'' attacks described in the previous studies involve the miner holding the mined block and not broadcasting this block to its pool operator [3], [11]–[20]. However, the pool temperately holds the mined target block according to the launch time threshold and holding period. The win rates obtained in the present study and relevant previous studies cannot be fairly compared. This study has examined the broadcast delay of mined blocks under enforce fork attacks in which the mined target hash values are held for a reasonable amount of time. An MBTH-EF attack is unique in that involves a miner immediately broadcasting a held block to

TABLE 1. Comparison between related studies.

Attack	Relevant work	Difference
MBTH	The aims of MBTH are to mine blocks, observe rewards from the mining pool, and hold the mined target block. Miner attacks [3], [11]–[19] and pool attacks [13]–[15], [17], [19], [20] are used to observe rewards from an attacked pool without contributing to the target hash value. In MBTH, the manager of an attacked pool is assumed to have no knowledge of the existence of selfish miners in the pool.	• In an MBTH attack, limited mined target hash values are held according to the launch time threshold and period to achieve additional early mining time for the next block. • The goal of an MBTH attack is to win a block at the starting point of a blockchain. • The mined block is broadcast to all contestants after the holding period. • The pool operator observes the blocks mined by miners and temperately holds the blocks. • No selfish miners exist in our work because selfish miners do not have permission to check and hold a mined block through the supported software of the pool operator.
MBTH-EF	• In MBTH-EF, selfish mining is performed to hold a set of blocks (fork chain) and win a game by broadcasting one or more holding blocks [3], [11], [19], [20], [30]; however, the selfish pool might sometimes be lost. • Some studies have analyzed in detail the effect of the stale block rate on selfish mining [29], [38]. • Stubborn mining strategies are used to earn high mining rewards when honest miners with a high share of the total computing power perform mining on the fork chain of the attacker [30], [31]. • A self-holding attack integrates selfish and stubborn mining attacks [17], [32]. • These works used a simplified static reward model.	• Method MBTH-EF only one block is held within a limited period. • The held block is immediately transferred to win back the mined block and avoid countermeasure detection when another contestant conducts mining at the same block depth. • The fork chain is enforced to increase the number of blocks, which leads to the formation of a large pool with high computing power that can win a set of blocks along the fork chain competition. • Some blocks along the fork chain might be mined by other contestants. • The computing power fraction of honest miners on the fork chain of the attacker is not a control variable. • Dynamic time-by-time, block-by-block, and pool-by-pool simulations are adopted to study MBTH-EF attacks.
iMBTH-EF	The iMBTH-EF strategy involves alternating between attacks and honest mining to manipulate mining difficulty to measure profit in various cryptocurrencies [24], [33]. The revenue share is balanced between mining time and difficulty with or without an attack. An attack is performed in a stage with high mining difficulty, and honest mining is performed in a stage with low mining difficulty.	• This work proposes an intermittent attack to balance the average mining time and mining difficulty according to the mining difficulty adjustment of a stage. • The effects of the iMBTH-EF attack on mining difficulty and mining time are evaluated in several difficulty adjustment stages. • Mining difficulty adjustment is conducted according to the timing of the iMBTH-EF attack to achieve a high win rate in a stage with high mining difficulty.

win back the block and avoid countermeasures when other contestants mine blocks at the same height as the held block. Selfish mining involves holding a set of blocks (a fork chain) and winning the mining game by broadcasting one or more of these holding blocks. Moreover, the MBTH-EF is extended to analyze the mining difficulty adjustment effect on mining time, winning rate, and the rate of fork occurrences. We do not compare the evaluation results between the proposed attacks and the selfish mining attacks because the main idea and attack target are very different.

III. PROBLEM MODEL AND PROPOSED SOLUTION

This section describes MBTH, MBTH-EF, and iMBTH-EF models and the proposed simulation algorithms designed for evaluating the effect of these attacks on block mining.

A. MBTH MODEL

Small and large pools with low and high computing power, respectively, might have similar probabilities of winning the mining game for one block. However, in the long term, a pool with higher computing power has a higher probability of finding the target hash values first. Under similar rates of opportunity, a larger pool has a higher probability of winning a fork chain, which covers several mining blocks, with a higher hash rate value. The holding threshold and holding period are determined by individual pools. Any pool can hold the mined target hash values to increase its winning probability for the next block. The present study considered two actions: (1) the holding pool loses the game and (2) the holding pool immediately broadcasts its mined block to enforce fork attacks when any other pool attempts to mine the target hash values.

Fig. 1 displays a comparison between a normal situation and the situation during MBTH attacks. Fig. 1(a) indicates that in the original mining competition, mined blocks are broadcast by a miner (who does not join a pool) or the pool of the block to the other contestants. The blocks broadcasted by the miner or pool are received by the other nodes after transmission delays. These nodes use the data in a received block's header and block body to (1) validate the received block and (2) check that the header yields a valid solution to the crypto puzzle. The term Q denotes a set of mined blocks; the time used to mine a block q ($q \in Q$) is denoted as t_q; and the mining time format is (min, s) for each pool. Pool A mines the first block of the mining competition (Block 1) in the shortest time and broadcasts it to all the other contestants. After the other pools receive a copy of Block 1, they begin to mine the next block (Block 2). The length of the competition period for a block is influenced by the mining time. The difference in the block discovery time between Pools B and C is shorter than the broadcasting delay; therefore, the blockchain splits. The blockchain split is propagated to all the contestants. The fork competition is resolved when the next block is mined. The contestants of the blocks in the longest branch win the mining rewards after n blocks.

In the aforementioned mining scenario, Pool A launches an MBTH attack if it mines Block 1 in a shorter time than the holding threshold (e.g., 8 min, which is shorter than the average Bitcoin mining interval). The mined block is held for 2 min by the miner (according to the average holding period), and the mined blocks are then sent to the other pools. The average holding period is set as 2 min for the sum of the holding threshold and holding period to be 10 min, which is the average Bitcoin mining interval. As displayed in Fig. 1(b), Pool A holds the hash values for Block 1 between t_1 and t_1', and this time interval is denoted as H. Pool A begins to mine Block 2 at t_1. At this time, Pools B, C, and D are still mining

Block 1. At t_1', Pool A broadcasts the held Block 1 to all the contestants to maintain its winning state for Block 1. Pool A begins to calculate appropriate hash values for the second block (Block 2) 2 min before the other pools do. Thus, the probability of Pool A winning the next block increases.

In a regular mining process, Pool B mines Block 2 [Fig. 1(a)]. However, when an MBTH attack is executed by Pool A, it starts mining the second block (Block 2) 2 min before the other pools do. The mining time of Pool A for Block 2 is 6 min 10 sec, which is equal to the difference between the actual mining time (8 min 20 sec) and the holding time (2 min). The aforementioned mining time is shorter than the holding threshold; thus, Pool A launches a holding attack for Block 3. However, in this case, the mining time for Block 3 is longer than the holding threshold (e.g., 8 min). Therefore, Pool A terminates the holding attack for Block 3. Subsequently, Pool A launches a holding attack for Block 4 because its mining time for this block is shorter than the holding threshold. However, Pool D mines Block 4 during the holding time (e.g., mining times of 7 min 50 sec and 10 min for Pools D and A, respectively, with a holding period of 2 min 10 sec). Consequently, Pool A loses Block 4 because of Action 1. Pool A might take Action 2 to broadcast Block 4 immediately at t_7', thereby causing a blockchain split. The outcome of the subsequent mining race is determined by the following n blocks.

All the contestants except the one that initiates an attack end up wasting their computing power during the block holding period if they cannot mine the next block during this period [e.g., $t_2 - t_1$ in Fig. 1(b)]. This phenomenon is attributable to the fact that a miner must observe a block at depth n before appending a block at depth $n+1$ to its blockchain. After a contestant broadcasts a mined block to the other contestants, the other contestants stop mining their candidate blocks and begin mining the next block at times such as t_2, t_4, t_6, and t_8. However, under an MBTH attack, the current block is broadcast several minutes after being mined (denoted by H). The other contestants end up wasting resources by attempting to mine an already mined block during the holding period (e.g., $t_1 - t_1'$ or $t_3 - t_3'$). Moreover, the other contestants are late in mining the next block.

B. FORK MODEL

The holding pool uses the broadcasting time deviation [22] to hold the mined block and broadcasts the block only when another pool has mined and broadcasted the mined block [e.g., the holding of Block 4 in Fig. 1(b)]. In this situation, if the holding pool loses the held block, it can take the advantage of its high computing power to win back the held block in the following several blocks because the fork chain enables it to extend the number of competition blocks. The blockchain system enters the fork phase, and the length of the fork chain in the following n blocks determines the winners, where n is set as six in the Bitcoin system. Fig. 2 presents an example of forking in which Pools A and B mine Block $H+1$ simultaneously. In this scenario, a set of branch blocks

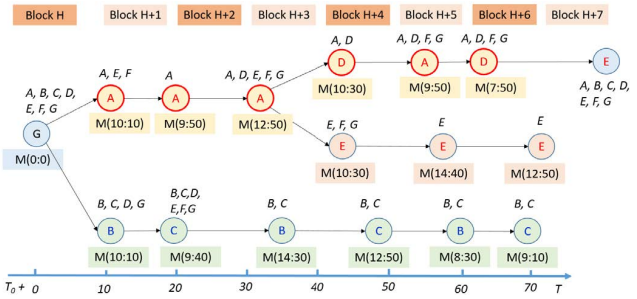


FIGURE 2. Example of a fork chain operation.

{A, A…E} is reserved for the following blocks because the length is the greatest according to the shortest time of the latest block. The holding pool wins the competitions along the branch chain, which covers several blocks, including Block $H+1$.

In the example illustrated in Fig. 2, Block H is mined at T_0. A fork is generated at T_0+ 10:10, and this fork causes Pools A and B to broadcast Block $H+1$, which is the mined block, within the broadcasting time. The other pools select a subchain to mine the next block. We considered the scenario in which a pool selects a fixed subchain, which includes a mined block. Thus, Pools A and B do not use other subchains from the ones they selected to increase their win rates for Block $H+1$. The other pools are free to mine the selected longest subchain to increase their winning probabilities in the following several blocks. Pools E and F select one subchain each when they receive the mined target hash value of Block $H+2$ from Pool C at T_0+ 19:50. However, Pool A mines Block $H+3$ before Pool B does; therefore, Pools D, E, F, G and shift, beginning to mine Block $H+4$ according to the observed hash values from Pool A. Two pools mine Block $H+4$ simultaneously such that a fork is generated in the {A, A, A} subchain. The computing power is distributed along three fork chains, which increases the average mining time. Finally, Pool D mines Block $H+6$ at the earliest time so that all the other pools continually mine Block $H+7$ along the {A, A, A, D, A, D} subchain. Other subchains are eliminated according to the consensus mechanisms of the Bitcoin system.

A large pool has a high probability of winning a subchain. Therefore, Pool A enforces a temporary fork after it receives the target hash value mined by other pools. As presented in Fig. 2, Pool A wins Block $H+1$ but might lose Blocks $H+2$ and $H+3$. The average mining time of Pool A is shorter than those of the other pools; thus, Pool A has a high probability of winning back held blocks after Block $H+6$. After Pool A wins back a held block, its probability of winning the subchain increases due to the presence of fewer contestants. Only five pools compete for Block $H+4$ on the {A, A, A} subchain. We evaluated a long-term mining competition in which a large pool has a high average win rate for thousands of mining blocks by using the proposed solution.

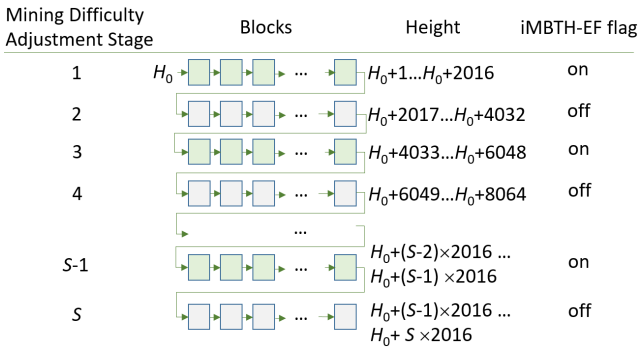


FIGURE 3. iMBTH-EF on–off attack model with a stage of mining difficulty adjustment.

C. IMBTH-EF MODEL

iMBTH-EF attacks are extensions of MBTH and MBTH-EF attacks that cover several stages of mining difficulty adjustment. A binary decision variable *iMBTH-EF_flag* is set as "on" or "off," indicating initiation and noninitiation, respectively, of MBTH and MBTH-EF attacks. Fig. 3 illustrates the mining difficulty adjustment stage based on the average mining rates of 2016 blocks. Given a block height H_0 that the adjusted mining difficulty is fixed for the following 2016 blocks in each stage. The decision variable of an iMBTH-EF attack can be set at any time. We examined the effects of an attack on the stages of mining difficulty adjustment and the influences of this adjustment on the average mining time, win rate, and fork occurrences.

We considered that S stages of mining difficulty adjustment and a regular on–off cycle are used in iMBTH-EF attacks. iMBTH-EF attacks involve the execution of MBTH-EF attacks according to the mining difficulty. The average mining time is overestimated under the execution of holding attacks. In this situation, iMBTH-EF attacks are launched to determine the average mining time and adjust the mining difficulty for the next stage. We employed the iMBTH-EF model to evaluate the win rate, mining time, and fork occurrence effects on the normal and abnormal average mining time and difficulty.

D. PROPOSED SOLUTION

In this study, the performance of different attacks was evaluated through the simulation of a scenario in which a fork chain is formed and an unlimited number of contestants. Although the block mining time can be approximately modeled as a random variable sampled from a homogenous Poisson distribution [39], MBTH, MBTH-EF, and iMBTH-EF attacks change the original block mining time. The proposed solution is applied to a mining game is simulated with a set of balls. Each ball represents a hash value that is not actually calculated using the SHA-256 algorithm [40]. A sequence of numbers on each ball represents the search sequence of a block with a nonce value, a time stamp, and a set of transactions. One yellow ball, which represents the target hash value, and $m-1$ black balls, which represent unsatisfied hash value, are considered. The repeated computation of the SHA-256 function by a miner by using block contents is an independent event in which two block contents might have the same hash value. However, the probability of two equivalent hash values being obtained from a large hash value space is approximately 0; therefore, the sampled balls without replacement in this study. The number of balls m represents the mining difficulty that the larger the value of m, the higher the mining difficulty. After several mining difficulty adjustments, the deviation between the simulated results obtained with the proposed algorithm and the historical data of the Bitcoin system was less than 0.69%. The STBR-HF algorithm was designed for a series of simulations.

The number of balls for each pool is initialized according to the historical data for the previous n difficulty adjustments in the Bitcoin system. The mining times in the proposed solution and Bitcoin system are different because the mining space is scaled down to one target hash value. The main reasons with only one target hash value are that (1) a one-block mining game ends once a pool receives mined block and (2) the time complexity to determine the mining time for each miner or pool is O(1), as proved in Lemma 1. The mining difficulty of the Bitcoin system is varied by adjusting the number of leading zeros according to the observed mining time for each set of the last 2016 blocks. The adjustment function is expressed as follows: new difficulty = old difficulty × actual mining time for the last 2016 blocks/20160 min.

The aforementioned adjustment depends on the difference between the actual mining time and average simulation time; specifically, the adjustment ratio is equal to the ratio between the actual mining time and the simulation time. The aforementioned ratio is based on the same concept as the mining difficulty adjustment equation, which relates the historical mining time and mining difficulty records of the Bitcoin system [23]. Algorithm 1 is a pseudocode for adjusting the mining difficulty to maintain the average mining time Γ according to the normalized computing power c_{pq}, where the pool $p \in P$ mines the block $q \in Q$. The number of pools is denoted by $|P|$. Accordingly, the mining difficulty is determined by the number of balls m. The simulation results obtained with Algorithm 1 were compared with the historical records of the Bitcoin system [23] to determine the deviation between the simulated results and historical records of the Bitcoin system.

Lemma 1: The time complexity is O(1), which is to calculate the time taken by the pool to draw the yellow ball (i.e., the target hash value).

Proof: The mining space for the SHA-256 function has a size of approximately 10^{77}, which represents a massive space for the current mining speed of approximately 10^{20} hash rate. Even when the mining speed is increased to 10^{30} hash rate, the probability of two block messages generating the same hash values is almost $10^{-47} \approx 0$. The mining process with the SHA-256 function is sampled without replacement, and the sampling events are independent and nonidentically distributed. Accordingly, we propose the

simulation of a stochastic mining process by using balls with random sequential values. One yellow ball is randomly distributed among black balls in each mining pool, and all the balls are numbered sequentially to represent the order of the mining process. The stochastic target ball with a stochastic mining process will result in randomly sequential ball values. Therefore, the mining time can be calculated as the number of sequential draws of yellow balls δ_{pq} divided by the hash rate c_{pq} of each pool $p \in P$ mines the block $q \in Q$. The time complexity to determine the mining time is O(1). ♦

$$t_{pq} = \left\lceil \frac{(\alpha \times \delta_{pq})}{c_{pq}} \right\rceil, \forall p \in P, q \in Q \tag{1}$$

where the variable α denotes the weight of the gap adjustment between the proposed simulation solution and the actual Bitcoin system. The value of α is determined by conducting a simulation with Algorithm 1 and adjusting the actual Bitcoin system over J iterations.

Algorithm 2 is a pseudocode for the proposed STBR-HF algorithm, which uses balls to simulate a mining game with MBTH, MBTH-EF and iMBTH-EF attacks. The attacking pool launches MBTH-EF attacks only when the *iMBTH-EF_flag* decision variable is set as 'on'. In the simulation with Algorithm 2, the hash outputs and validation results are assumed to be indicated by the number sequences on the balls in a pool. The target valid ball, namely the yellow ball, might be located at different positions because the message and transactions listed in a block of the holding pool are different from those listed in the blocks of other pools. A pool with higher computing power can draw a larger number of balls per unit time. If a ball drawn by a pool is valid, the pool wins the reward. If a pool holds the mined block according to the *iMBTH-EF_flag*, the mining time is equal to the mining time minus holding time of previous block. When multiple pools simultaneously draw the valid ball or when any other pool mines the block during the holding period, the mining time of the holding pool is changed to the mined time by other pool. The attacking pool broadcasts its mined block after receiving a block mined by any other pool. Therefore, a fork chain is generated. Only the longest branch is retained when the length of any branch is longer than n. Subsequently, the block winner on the branch is determined. Successful metric win rates w_{ps} are calculated according to the number of blocks mined for each pool $p \in P$ per stage $s \in S$. Once the winner of a block is identified, the reward is calculated according to the pool reward policies.

The main features of the proposed simulation solution include: (1) the mining process for each hash value process can be adopted for any size of computing power, (2) the large mining space results in mining independent among mining pools for various number of mining pools, (3) the competition results are determined by one block helps to evaluate relevant fork chain attack, and (4) the competition process covers several mining difficulty stages to simulate a long competition scenarios. These features can be adopted to simulate control variables or functional extensions in a PoW consensus mechanism so that the proposed algorithm can be applied to other attacks with similar favors to these attacks.

The limitations of the proposed solution are as follows:

- If the computing power proportion of a pool is insufficient, the simulation error is high.
- A finite number of balls are included without replacement in the simulation for a miner or pool to be capable of identifying the target yellow ball within a finite time. The proposed simulation solution is inapplicable in the case of an extremely long mining time.

Algorithm 1 Algorithm for Mining Difficulty Adjustment

```
Input: a set of contestants P, the normalized computing power c_pq, number of
  evaluated blocks Q, rate threshold of fork chain r, and total balls m.
Output: the number of balls m, the number of rounds for each block w_q.
Set the number of balls m based on the Bitcoin system
for j = 0 to J do   // denote J the number of learning iterations
    for q = 0 to Q do   // to run each block q ∈ Q
        for p = 0 to P do   // each contestant p ∈ P
            δ_qq ← m×random(0,1) //follow exponential or geometric
            distribution
            t_pq ← ⌈(α×δ_pq) / c_pq⌉ // calculate t_pq time to find a yellow ball;

        end for // p ∈ P
        w_pq ← min(t_pq)   // among contestants P;
        a_t ← a_t + w_q;   // subtotal amount of mining time a_t.
        if fork == FALSE then   // not in fork chain status
            if |w_pq| == 1 then   // only one winner in this block q
                w_p ← w_p + 1  // increase 1 successful block for winner p;
            else
                fork ← TRUE
                fork_len ← 1  // the length of fork chains;
                a_f ← |w_pq|  // record the number of fork chains;
                mark_w_pf ← assign p to fork chain f in w_pq

            end if
        else // if fork == TRUE then
            for f ∈ a_f do
                if |w_pq| > 1 and p ∈ mark_w_pf then
                    a_f ← a_f + |w_pq| // increase the number of fork
                    chains;
                    mark_w_pf ← assign p to fork chain f in w_pq
                    mark the winner p in fork chain f;

                end if

            end for
            fork_len ← fork_len + 1
            mark the winner p in fork chain f;
            if (fork_len > x) then // to determine winner if x >= 6
                record the winner along the longest chain f
                w_p ← w_p + 1 // increase 1 successful block for winner p;
                reset a_f, fork_len, fork // not in fork chain status

            end if // fork_len > x

        end if // fork == FALSE

    end for // q ∈ Q

end for // j ∈ J
simulation_time ← mining_time / |Q|
m ← m×(1 - (simulation_time – actual_time_j) / simulation_time)
```

Algorithm 2 STBR-HF

Input: a set of contestants P, number of evaluating blocks $|Q|$, normalized computing power c_{pq}, actual mining time Γ [23], computing power c_{pq} [23], number of balls m, and attacking pool e_p.
Output: Win rate w_p, rewards γ_p and r_{pi} for pool p and miner i;
total_balls m = Mining_Difficulty_Adjustment (Γ, c_{pq}) // call Algorithm 1;
iMBTH-EF_flag ← TRUE // change over iMBTH-EF_flag to on or off
for $s = 0$ *to* S **do** // to run S mining adjustment stage;
  for $q = 0$ *to* Q **do** // to run each block $q \in Q$
    for $p = 0$ *to* P **do** // each contestant $p \in P$
      $\delta_{pq} \leftarrow m \times$random(0,1) //follow exponential or geometric distribution
      $t_{pq} \leftarrow \left\lceil \frac{(\alpha \times \delta pq)}{cpq} \right\rceil$ //the number of time slots to find a yellow ball;
      If *iMBTH-EF_flag* == TRUE **then**
        $t_{pq} = t_{pq}$−pre_block_holding_time(p).
      end if
    end for // $p \in P$
    $w_{\theta p} \leftarrow \min(t_{pq})$ // among contestants P;
    if *iMBTH-EF_flag* == TRUE and $e_p \notin w_{\theta p}$ **then**
      $t_{pq} \leftarrow t_{\theta p}$
    end if
    $w_{\theta p} \leftarrow \min(t_{pq})$
    if fork == **FALSE then** // not in fork chain status
      if $|w_{pq}| == 1$ **then** // only one winner in this block q
        $w_p \leftarrow w_p + 1$ // increase 1 successful block for winner p;
      else
        fork ← TRUE
        fork_len ← 1 // the length of fork chains;
        $a_f \leftarrow |w_{pq}|$ // record the number of fork chains;
        *mark_*w_{pf} ← assign p to fork chain f in w_{pq}
      end if
    else // if *fork* == TRUE then
      for $f \in a_f$ **do**
        if $|w_{pq}| > 1$ **and** $p \in$ mark_w_{pf} **then**
          $a_f \leftarrow a_f + |w_{pq}|$ // increase the number of fork chains;
          mark_w_{pf} ← assign p to fork chain f in w_{pq}
          mark the winner p in fork chain f;
        end if
      end for
      fork_*len* ← fork_*len* + 1
      mark the winner p in fork chain f;
      if (fork_*len* > β) **then** // to determine winner if $\beta >= 6$
        record the winner along the longest chain f
        $w_p \leftarrow w_p + 1$ // increase 1 successful block for winner p;
        reset a_f, fork_*len*, fork // not in fork chain status
      end if // fork_*len* > β
    end if // fork == **FALSE**
  end for // $q \in Q$
end for // $s \in S$
$w_{ps} \leftarrow w_p/|Q|$

IV. SIMULATION AND DISCUSSION

A series of simulations were conducted by varying the pool size, temporary holding threshold, and holding period to examine the effects of these factors on the win rate, mining time, mining difficulty, and rate of fork occurrences.

TABLE 2. Evaluation environment and parameters.

Parameter	Description
Number of contestants ($\|P\|$)	5 and 20
Total normalized computing power (h/s)	data in [5]
Number of simulation balls (m) for each adjustment	Algorithm 1
Number of hash values for successful mining (κ)	1
Number of simulation blocks (Q)	10^6
Number of blocks to determine the winners of a branch chain (n)	6
Average mined block broadcasting delay (η)	1 sec
Number of evaluations for each simulation result	10
Number of mining difficulty adjustment stages (S)	5

A. SIMULATION ENVIRONMENT

The simulations were conducted using Google Colab as a running platform and Python as a general-purpose programming language. The web-based development environment and the program is run on the cloud computing system by using Nvidia K80, T4, P4, or P100 GPU processing hardware. Table 2 presents the simulation parameters related to the Bitcoin system [23]. We randomly selected a mining difficulty adjustment stage with a block depth of 622 944 (between the depths of 546 336 and 645 120). The mining difficulty is equal to 13.91T. The simulation solution matched the number of balls ten times of mining difficulty $10 \times 13.91 \times 10^{12} \times 2^{32}$, which is set as the initial number of balls. This value was then iterated in Algorithm 1 to narrow the gap between the simulated and actual mining difficulty of the Bitcoin system. Finally, the mining difficulty was simulated for the following series of evaluations. The computing power of a pool is presented in [5] and the hash rate of the entire network was 99.59 EH/s. The hash rate is evaluated as the number of balls drawn per second by the complete system. The average mined block broadcast delay (η), which has an exponential distribution. The batch time is compared with the broadcast delay to check whether the blockchain splits. The blockchain splits when the batch time and broadcast delay are equal. The batch time is aggregated to determine the mining time for each block. A series of simulations were performed using various parameters to investigate MBTH-EF attacks. The main evaluation metric was the win rate. The win rate of a pool is equal to the number of blocks won by a pool divided by the total number of simulation blocks $|Q|$. We adopted three metrics, namely the average mining time per difficulty adjustment (over 2016 blocks), mining difficulty, and rate of fork occurrence, to understand the effects of MBTH-EF attacks.

Several control variables, including the pool size, holding threshold, and holding period, were employed to evaluate the effects of MBTH attacks. For example, we investigated the effect of the win rate on pools of various sizes under a fixed holding threshold and holding period. When the holding threshold was fixed at a high level, the possibility of executing an MBTH attack decreased. The holding period was adjusted according to the time used for successfully mining a block. A longer mining time led to a higher risk of losing the mined block because the probability of the other

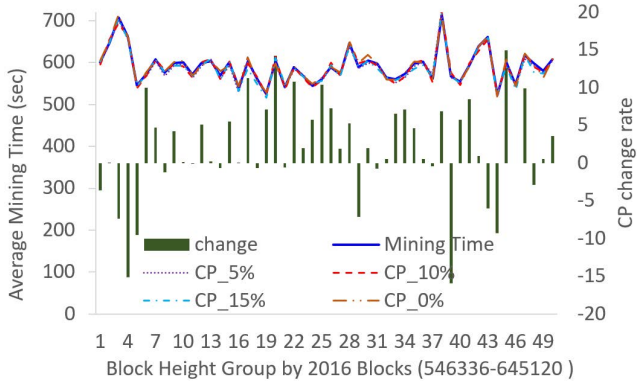


FIGURE 4. Simulation time bias and computing power distribution.

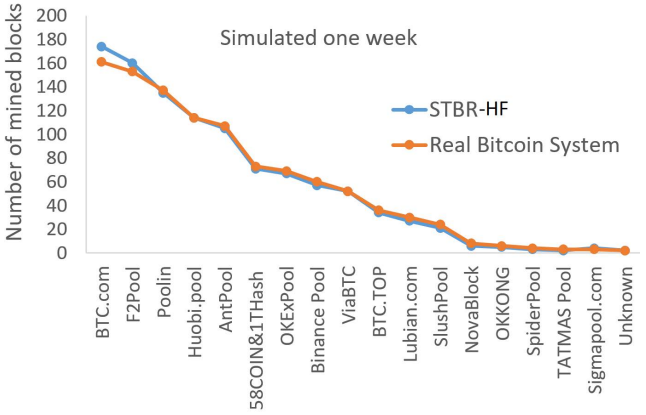


FIGURE 5. Deviation between the simulation and system operation results for the number of blocks mined by the top 17 pools in 1 week (block heights of 649 642–650 683).

contestants mining the block increased. A higher probability of initiating a holding attack resulted in a lower holding time for announcing the mined block. The mining difficulty affected the influence of the mining time on the holding attack.

B. EVALUATION OF BIAS IN THE PROPOSED SOLUTION

Fig. 4 displays the average simulation time, which is comparable with the mining time in the actual mining records of the Bitcoin system. The simulation bias for a blockchain height of 546 336–645 120 was 0.69%; thus, the proposed STBR-HF algorithm could accurately simulate the Bitcoin system. The mining difficulty was regularly adjusted, and the adjusted value was maintained for an average of 600 s with the last 2016 blocks. The computing power of the top 20 pools randomly varied by 5%, 10%, and 15% [5]. The results indicated that computing power variations for an individual pool did not significantly affect the mining time. Major fluctuations in the mining time were caused by variations in opportunity and in the network computing power.

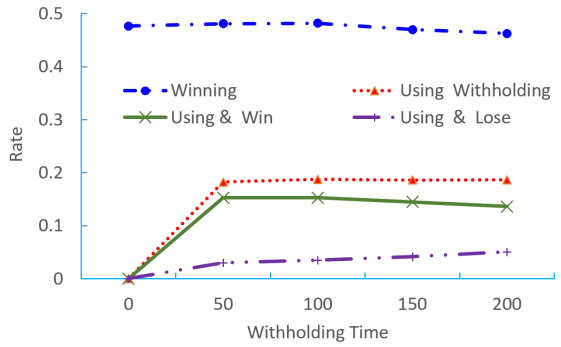
Fig. 5 displays the evaluation results for successful block mining by the top 17 mining pools. The other small pools are included in the ''Unknown'' group. The 1-week evaluation data for the top 17 pools indicated that small pools did not win the game during the evaluation period. For the top 10 pools, the difference in the number of mined blocks between the simulation and system operation results was less than 8%. The gap between the top 10 and top 17 pools was large because the number of mined blocks was small. Overall, the results indicate that the proposed simulation method is only suitable for pools that can mine more than 30 blocks within the given evaluation period. An acceptable difference between the results of the proposed simulation method and those of the actual Bitcoin system is $\leq 3\%$. The computing power percentage of a pool or the number of mined blocks should be sufficiently high to achieve such a low difference.

C. SIMULATION OF MBTH ATTACKS

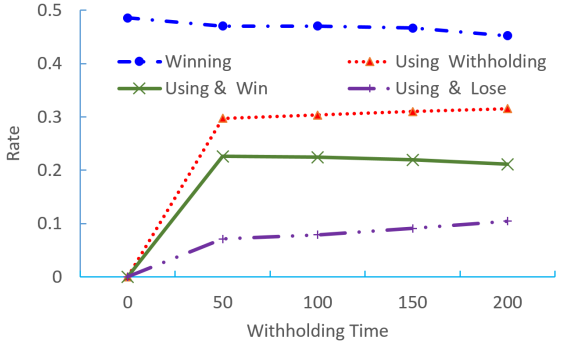
Fig. 6 presents the simulation results regarding the win rates of the pools that executed MBTH attacks. The rates of winning were determined when pools engaged in temporary holding and won or lost blocks. The probability of executing temporary holding is equal to the number of held blocks divided by the number of simulation blocks. The probability of executing temporary holding and winning is defined as the number of blocks won after holding divided by the number of simulation blocks. Moreover, the probability of executing temporary holding and losing is defined as the number of blocks lost after holding divided by the number of simulation blocks. The win probability is the most crucial indicator of pool revenue. Other probabilities, such as the frequency of fork occurrences, are used to examine the effects of different factors on the win probability and MBTH attacks.

Although small pools can initiate MBTH attacks, the probability of such attacks is low because the win rates of small pools are low and the average lengths of their holding periods are short. Consequently, the aforementioned attacks have a small effect on the following blocks. For a large pool, the rate of mining time lower than the holding threshold and win rate, which exert effects on the revenue share, are high. However, the results displayed in Fig. 6 indicate that the holding threshold and holding time do not significantly affect the win rate. This result is obtained because of the stochastic nature of the mining game, which may result in some pools having a short mining time for the current block but a long mining time for the next block and other pools having the opposite—that is, a long mining time for the current block but a short mining time for the next block. The holding period is not easy to control for increasing the win rate. If the holding period is excessively long, other pools might have a higher probability of winning the held block. Figs. 6(a) and 6(b) demonstrate the effect of the holding period on the win rate. The win rate is unaffected by changes in the holding threshold and holding period, which is partially attributable to the fact that the rates of winning and losing are comparable under MBTH attacks.

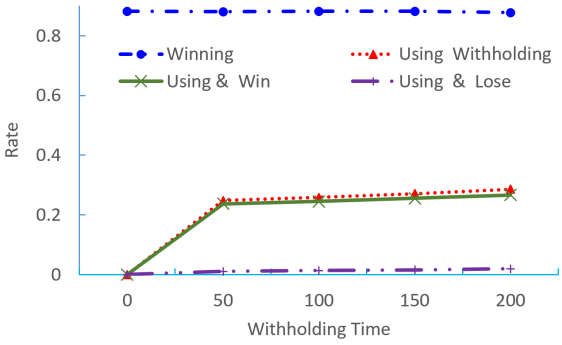
Although the computing size is four times larger than the second pool, the improvement in the win rate is not significant because the original win rate is high when no holding attack



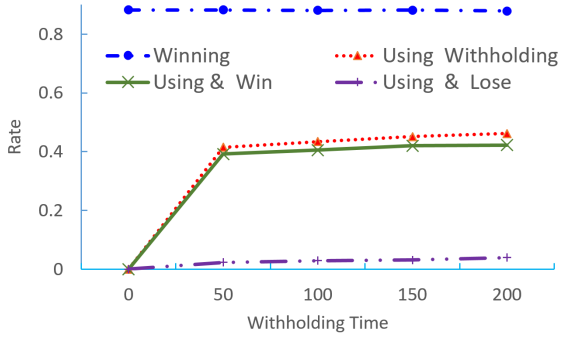
(a) Win rates of multiple pools with varying computing power ratios {0.40, 0.15, 0.15, 0.15, 0.15} when the holding threshold was set as 300 s.



(b) Win rates of multiple pools with varying computing power ratios {0.40, 0.15, 0.15, 0.15, 0.15} when the holding threshold was set as 500 s.



(c) Win rates of two major contestants {0.80, 0.197, 0.001, 0.001, 0.001} when the holding threshold was 300 s.



(d) Win rates of two major contestants {0.80, 0.197, 0.001, 0.001, 0.001} when the holding threshold was 500 s.

FIGURE 6. Win rates of pools with varying computing power engaging in MBTH attacks under different holding thresholds.

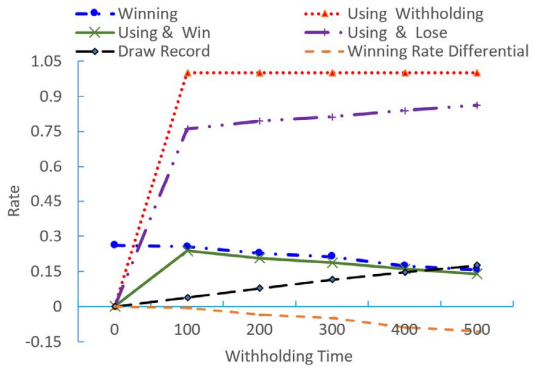
is launched. The increasing win rate is small although the probability of executing temporary holding and winning is considerably higher than those of executing temporary holding and losing. The original winning probability of the next block is approximately 0.85. Figs. 6(c) and 6(d) indicate that the probability of winning a block, which might be lost even when no attack is used, is low. In summary, a holding attack should be adopted by a large pool. However, this attack does not considerably increase the win rate for large pools. MBTH attacks do not significantly affect the win rate because setting the holding period is difficult. If the holding period is overly long, the mined block might be lost; however, if the holding period is overly short, a win cannot be ensured in the mining game for the next block.

D. SIMULATION OF MBTH-EF ATTACKS

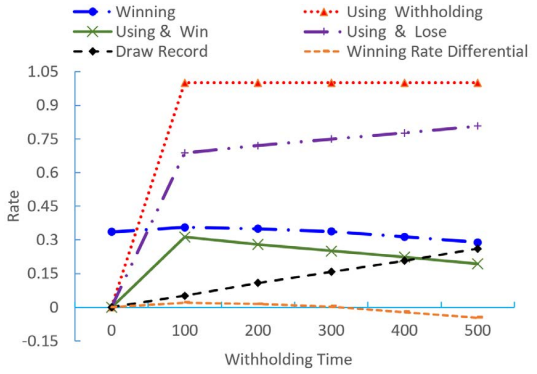
Simulations were performed to evaluate how MBTH-EF attacks affect the win rate. The enforce fork is used when the holding pool receives the block mined by any other pool. Subsequently, the holding pool broadcasts its mined block to launch an enforce fork attack. Fig. 7 presents the simulation results for the evaluation of MBTH-EF attacks under varying computing power allocations among five pools. The holding threshold was set as 6000 s to ensure that the evaluated pool always executed an MBTH attack. The holding period was changed from 0 to 500 s. At a holding time of 0 s, no holding attack was used; in other words, a general pool operation was launched. The results obtained were the baseline for the comparison of the evaluation metric settings. As shown in Fig. 7(a), the win rate achieved with the MBTH-EF attack was low even when the holding period was long. Fig. 7(b) demonstrates that an increase in the holding time caused an increase in the win rate. When the holding time increased, the curve corresponding to executing a MBTH-EF attack and winning flattened (denoted as 'Using and win') but the rate of fork occurrences increased, which allowed the holding pool to win back blocks mined by other pools. Figs. 7(c) and 7(d) indicates that the computing power of the holding pool was sufficient for achieving a high win rate. The ratio of computing power to the whole network is not as high as the win rate; however, the win rate of a pool increases from 0.54 to 0.63 (a 16.67% additional revenue share) with a computing power share of 45%, the highest in a set specifying the computing power held by each unit {0.45, 0.175, 0.15, 0.125, 0.1}. The win rate increased when the rate of 'Using and win' decreases, but the rate of 'Using and lose' increases. Under a fork attack, these lost blocks failed and the holding pool won back some mined blocks. If a large pool is four times larger than the other pools, the increase in the win rate under the aforementioned scenario exceeds 27%.

E. SIMULATION OF HOLDING AND ADJUSTMENT

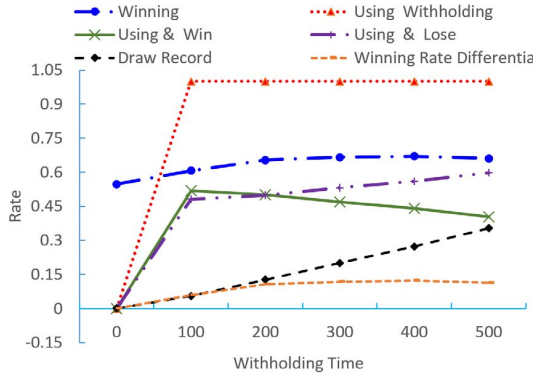
Fig. 8(a) displays the variations in the holding time with the average mining time for a fixed computing power allocation among pools. Before mining difficulty adjustment, a longer holding threshold and holding period resulted in a longer average mining time. The average mining time decreased with adjustments in the mining difficulty. The average mining



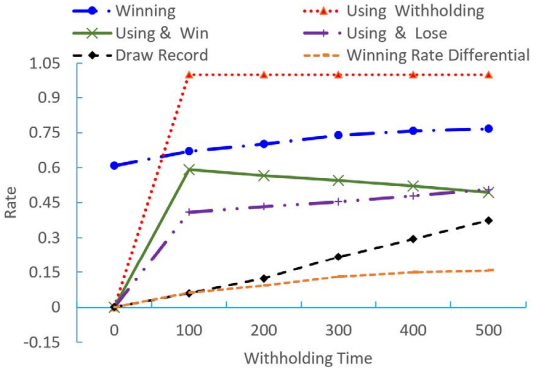
(a) Set of computing power: {0.25, 0.195, 0.19, 0.185, 0.18};



(b) set of computing power: {0.3, 0.19, 0.18, 0.17, 0.16};

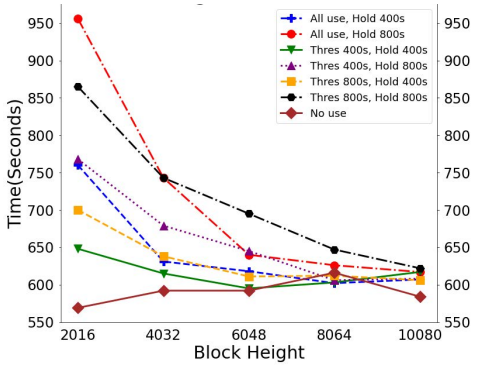


(c) set of computing power: {0.45, 0.175, 0.15, 0.125, 0.1};

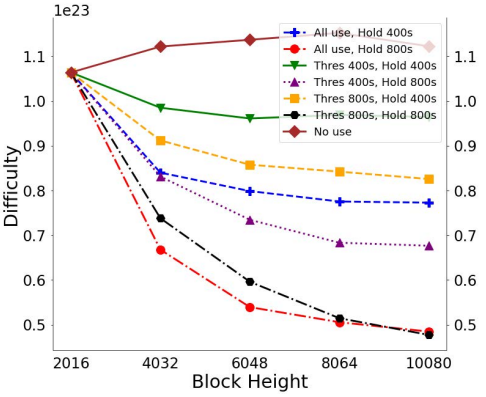


(d) set of computing power: {0.5, 0.125, 0.125, 0.125, 0.125}.

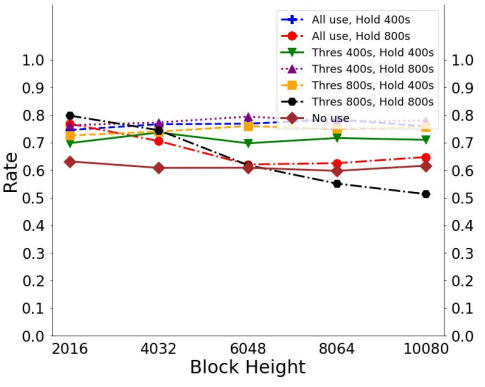
FIGURE 7. Win rates of five pools with varying computing power allocations that executed MBTH-EF attacks.



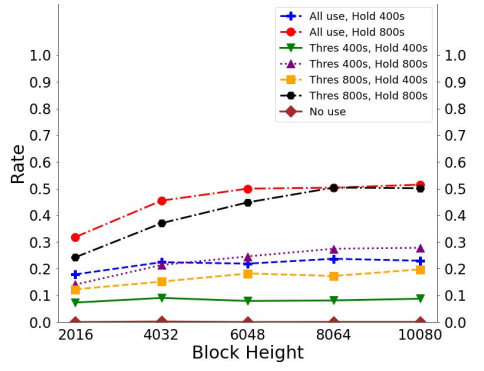
(a) Mining time;



(b) mining difficulty;



(c) win rate;



(d) rate of fork occurrences.

FIGURE 8. Effects of MBTH-EF attacks in several stages of mining difficulty adjustment.

time approached 10 min when Algorithm 1 was employed. When the holding strategy was used, the mining time was longer than that when attacks were not launched. The mining difficulty was lower than the actual total computing power for the next stage when the MBTH-EF is launched in this stage. Fig. 8(b) indicates that the mining difficulty decreased following mining difficulty adjustments. After the first difficulty adjustment, the mining difficulty decreased to half of the initial difficulty.

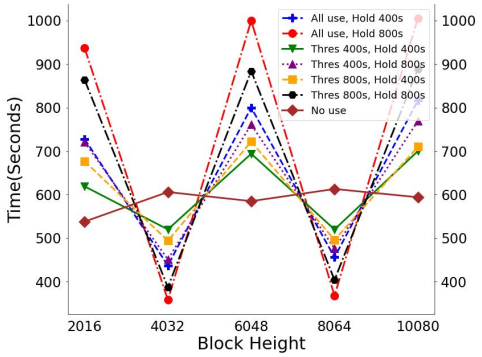
Fig. 8(c) indicates that the win rate is higher than the pool without attack when the threshold and holding period are long (e.g., a holding period of 800 s). The win rate decreased after several stages of mining difficulty adjustment because these adjustments reduced the mining difficulty and shortened the average mining time. As shown in Fig. 8(d), that the rate of fork occurrences increased when holding and fork attacks were continually used. This result demonstrates that such attacks cannot be used to maintain mining advantages over an extended period.

When a MBTH-EF attack is executed, the mining time increases such that the mining difficulty does not exactly reflect the computing power. Although lower mining difficulty results in lower computing power, MBTH-EF attacks are adopted by few pools when the other pools are continually mining. Therefore, when using such attacks, the mining time increases but the required computing resources do not decrease. Furthermore, lower mining difficulty would reduce the mining time of all pools other than the holding pool. In this situation, the probability of winning a block against the competition by using a MBTH-EF attack is reduced.

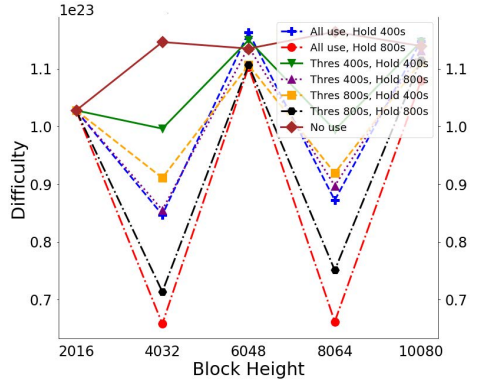
F. SIMULATION OF INTERMITTENT ATTACKS

MBTH attacks increase the mining time per block, which results in low mining difficulty. The average mining time for each block is longer than the mining time that causes the difficulty adjustment underestimate. Low mining difficulty results in the availability of a short time for determining the target hash value. Because of the short holding threshold and holding period, the attacking pool exhibits a low win rate after several difficulty adjustments. Thus, iMBTH-EF attacks are adopted to maintain a high win rate intermittently during mining difficulty adjustment.

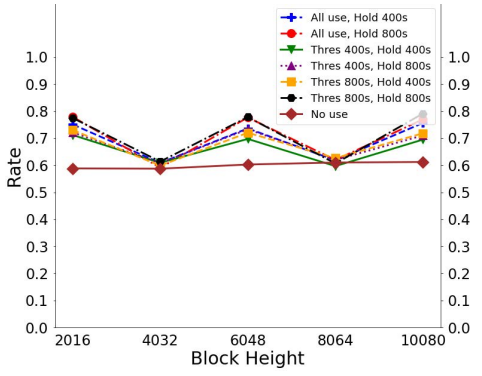
Fig. 9 depicts the simulation results for the effects of MBTH-EF attacks on the mining competition cycle for each block during several stages of mining difficulty adjustment. Fig. 9(a) indicates that the mining time decreased with the execution of MBTH-EF attacks. MBTH-EF attacks lead to a long mining time; however, the mining time is still shorter than the mining time observed without MBTH-EF attacks [Fig. 9(b)]. The aforementioned result is mainly ascribable to mining difficulty adjustments. Therefore, the average win rates are high at block heights of 2016, 6048, and 10 080. Fig. 9(c) indicates that iMBTH-EF attacks resulted



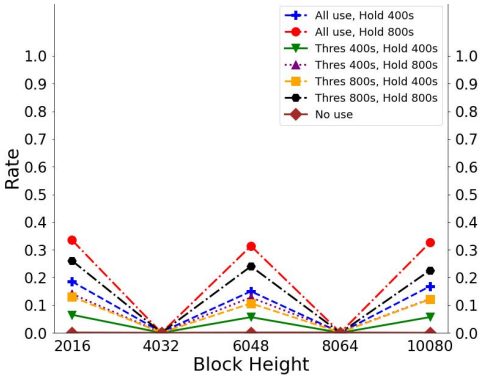
(a) Average mining time per 2016 blocks;



(b) mining difficulty;



(c) win rate of the attacking pool;



(d) ratio of fork occurrences.

FIGURE 9. Effects of the attack mining competition cycle for each block on various parameters under several stages of mining difficulty adjustment.

in high win rates under normal mining difficulty for the stages between block heights of 2016–4031, 6048–8063, and 10 080–12 095. These results are consistent with those presented in Fig. 8(c). Fig. 9(d) indicates that the rate of fork occurrences increased when iMBTH-EF attacks were executed. The rate of fork occurrences was approximately half the win rate achieved with MBTH-EF attacks when a pool always used MBTH attacks, and the longest holding period was 800 s. A long holding period resulted in an approximate increase of 47% in the frequency of enforce fork attacks, which increased the win rate by approximately 0.21. Extended MBTH-EF and iMBTH-EF attacks significantly increase the win rates within one and several periods of mining difficulty adjustment, respectively. However, the additional revenue is low when the computing power share of a pool is higher than 90% because the win rate is already higher than 0.9. The simulation results indicate that the ratio of fork occurrences was almost zero when no pool launched a MBTH-EF attack.

G. DISCUSSION

The objective of a pool in executing a block holding attack is to compute the next block earlier than other pools can. One method for determining whether a pool has adopted an MBTH-EF attack is to check whether a difference exists between the mining completion time for the previous block and the transaction time included in the next block. However, the log data of Bitcoin operation do not include the transaction request time. The transaction request time is the time when a user launches (or requests) a transaction. However, the Bitcoin operation log data used do not include the transaction request time.

If the checking of the mining completion time fails for one mined block, one mined block does not affect the proposed method that the statistical data are observed to cover a set of mined blocks. If the checking of the mining completion time fails for several mined blocks, the honest miners or pools detect the malicious miner and reach consensus in not accepting or ignoring the mined blocks, which form a stale branch. The details of the proposed detection method are set as future work.

A large pool has a high win rate when it initiates MBTH attacks. Although other pools might successfully mine the current block during the holding period, the enforce fork attack provides a large pool with a high probability of winning a fork chain. If a pool can mine a block early and hold this block, it can obtain more time than other pools can to mine the next block; in other words, the holding pool can mine the next block faster than the other contestants can. Consequently, the next third block is still stay in extra early mining time at a starting point. Numerous iterations are conducted until the mining time exceeds the holding threshold. However, as indicated in one of the experiments, this strategy does not provide a considerable advantage to the holding pool because of the stochastic nature of the mining process. A pool might observe a mined block at heights of H and $H+1$ over a short and long time, respectively; thus, the effect of MBTH attacks is limited.

From the viewpoint of the blockchain system, MBTH attacks result in an unfair mining competition. These attacks can reduce the mining difficulty and thus the required computing power. Moreover, they can cause the wastage of computing resources by other contestants as they continue to mine an already mined block held by the attacking pool. Changing mining rules to ensure that a mined block is immediately broadcast by the holding pool to all other mining contestants constitutes a pivotal issue. A pool can launch a holding attack to begin mining the next block earlier than other pools can and to extend the number of competition blocks with fork attacks. Such an attack may be advantageous for large pools. However, it causes an unfair mining competition and a wastage in the computing power of the other pools, which aside from searching for an already mined block begin searching for the next block later than the holding pool does.

A pool that executes MBTH-EF attacks might broadcast the mined block after it receives a mining message from any other contestant. In this regard, a fork attack can reduce the loss rate of held blocks. When MBTH-EF attacks are executed, the blockchain system determines whether the held block is one of the following x blocks. A large pool, which has high computing power, has a high probability of winning a fork chain. The frequency of fork occurrences is determined to identify whether MBTH-EF attacks are initiated. This rate varies considerably because of the massive hash space and long mining time. The number of contestants is limited because many miners are aggregated as one contestant in pool-age blockchain mining. The ratio of fork occurrence is calculated for each mining difficulty stage. If ratio is higher than historical results in a stage, it means the attack might be launched. Furthermore, the mining difficulty increases when the computing power increases and the average mining time approaches to ten minutes except a special event caused large enough computing power decreased.

V. CONCLUSION

Pool-based mined block MBTH-EF attacks were analyzed and evaluated herein. The STBR-HF algorithm was employed to understand the effects of these attacks on postponing the announcement of the mined target block. A large pool possesses high computing power for mining the target block in a short time. Thus, such a pool has a high probability of mining the target block. It can use a holding attack to hold the mined block and begin mining the next block earlier than other pools can. Although a larger pool does not have a higher win rate for the next block, it has a higher win rate for the following several blocks. Thus, when large pools use fork chain attacks, they have a higher probability of winning a set of blocks than do small pools. This study also evaluated the effect of MBTH-EF attacks on the mining competition. The mining difficulty adjustment process results in a reduction in the mining difficulty. Consequently, the

mining difficulty does not exactly correspond to the mining time or the computing power for the entire network in each phase of mining difficulty adjustment. The reduction in the required mining time leads to a higher probability of a large pool launching a block holding attack but a shortening of the holding period. Advanced iMBTH-EF attacks maintain a high win rate for the attacking pool during mining difficulty adjustments. In a future study, we will examine how to improve the fairness of the mining competition.

REFERENCES

[1] R. Qin, Y. Yuan, and F. Wang, "Research on the selection strategies of blockchain mining pools," *IEEE Trans. Computat. Social Syst.*, vol. 5, no. 3, pp. 748–757, Sep. 2018.
[2] Y. Liu, K. Qian, K. Wang, and L. He, "Effective scaling of blockchain beyond consensus innovations and Moore's law: Challenges and opportunities," *IEEE Syst. J.*, early access, Jun. 28, 2021, doi: 10.1109/JSYST.2021.3087798.
[3] E. Ittay and S. Emin, "Majority is not enough: Bitcoin mining is vulnerable," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.*, vol. 8437, 2013, pp. 436–454, doi: 10.48550/arXiv.1311.0243.
[4] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Bitcoin Whitepaper*, Oct. 2008, Art. no. 21260.
[5] *BTC Pool Distribution*. Accessed: Nov. 22, 2020. [Online]. Available: https://btc.com/stats/pool
[6] C. Chen, X. Chen, J. Yu, W. Wu, and D. Wu, "Impact of temporary fork on the evolution of mining pools in blockchain networks: An evolutionary game analysis," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 1, pp. 400–418, Jan. 2021.
[7] N. Houy, "The Bitcoin mining game," *Ledger J.*, vol. 1, no. 13, pp. 53–68, Dec. 2016.
[8] T. Wang, X. Bai, H. Wang, S. C. Liew, and S. Zhang, "Game-theoretical analysis of mining strategy for bitcoin-NG blockchain protocol," *IEEE Syst. J.*, vol. 15, no. 2, pp. 2708–2719, Jun. 2021.
[9] M. Saad, J. Choi, D. Nyang, J. Kim, and A. Mohaisen, "Toward characterizing blockchain-based cryptocurrencies for highly accurate predictions," *IEEE Syst. J.*, vol. 14, no. 1, pp. 321–332, Mar. 2020.
[10] S. G. Motlagh, J. Misic, and V. B. Misic, "The impact of selfish mining on bitcoin network performance," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 1, pp. 724–735, Jan. 2021.
[11] Y. Kwon, D. Kim, Y. Son, E. Vasserman, and Y. Kim, "Be selfish and avoid dilemmas: Fork after withholding (FAW) attacks on bitcoin," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, New York, NY, USA, Oct. 2017, pp. 195–209.
[12] S.-N. Li, Z. Yang, and C. J. Tessone, "Mining blocks in a row: A statistical study of fairness in bitcoin mining," in *Proc. IEEE Int. Conf. Blockchain Cryptocurrency (ICBC)*, Toronto, ON, Canada, May 2020, pp. 1–4.
[13] S. Kim and S.-G. Hahn, "Mining pool manipulation in blockchain network over evolutionary block withholding attack," *IEEE Access*, vol. 7, pp. 144230–144244, 2019.
[14] Y. Chen, H. Chen, M. Han, B. Liu, Q. Chen, and T. Ren, "A novel computing power allocation algorithm for blockchain system in multiple mining pools under withholding attack," *IEEE Access*, vol. 8, pp. 155630–155644, 2020.
[15] S. Shalini and H. Santhi, "A survey on various attacks in bitcoin and cryptocurrency," in *Proc. Int. Conf. Commun. Signal Process. (ICCSP)*, Chennai, India, Apr. 2019, pp. 220–224.
[16] I. Eyal, "The miner's dilemma," in *Proc. IEEE Symp. Secur. Privacy*, San Jose, CA, USA, May 2015, pp. 89–103.
[17] R. Yang, X. Chang, J. Misic, V. Misic, and H. Kang, "On selfholding attack impact on imperfect PoW blockchain networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 4, pp. 3073–3086, Oct. 2021.
[18] H. Kang, X. Chang, R. Yang, J. Misic, and V. B. Misic, "Understanding selfish mining in imperfect bitcoin and ethereum networks with extended forks," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 3, pp. 3079–3091, Sep. 2021, doi: 10.1109/TNSM.2021.3073414.
[19] M. Saad, J. Spaulding, L. Njilla, C. Kamhoua, S. Shetty, D. Nyang, and D. Mohaisen, "Exploring the attack surface of blockchain: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 1977–2008, 3rd Quart., 2020.
[20] S. Bag, S. Ruj, and K. Sakurai, "Bitcoin block withholding attack: Analysis and mitigation," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 8, pp. 1967–1978, Aug. 2017.
[21] Z. Liu, N. C. Luong, W. Wang, D. Niyato, P. Wang, Y.-C. Liang, and D. I. Kim, "A survey on blockchain: A game theoretical perspective," *IEEE Access*, vol. 7, pp. 47615–47643, 2019.
[22] X. Feng, J. Ma, Y. Miao, X. Liu, and K.-K.-R. Choo, "Social characteristic-based propagation-efficient PBFT protocol to broadcast in unstructured overlay networks," *IEEE Trans. Dependable Secure Comput.*, early access, Aug. 12, 2021, doi: 10.1109/TDSC.2021.3104465.
[23] *BTC Difficulty Statistic*. Accessed: Nov. 28, 2020. [Online]. Available: https://btc.com/stats/diff
[24] K. A. Negy, P. R. Rizun, and E. G. Sirer, "Selfish mining re-examined," in *Proc. Int. Conf. Financial Cryptography Data Secur.*, 2020, pp. 61–78.
[25] K. Wang, Y. Wang, and Z. Ji, "Defending blockchain forking attack by delaying MTC confirmation," *IEEE Access*, vol. 8, pp. 113847–113859, 2020.
[26] R. Qin, Y. Yuan, and F.-Y. Wang, "Optimal block withholding strategies for blockchain mining pools," *IEEE Trans. Computat. Social Syst.*, vol. 7, no. 3, pp. 709–717, Jun. 2020.
[27] J. Lee and Y. Kim, "Preventing bitcoin selfish mining using transaction creation time," in *Proc. Int. Conf. Softw. Secur. Assurance (ICSSA)*, Seoul, South Korea, Jul. 2018, pp. 19–24.
[28] R. Zhang and B. Preneel, "Lay down the common metrics: Evaluating proof-of-work consensus protocols' security," in *Proc. IEEE Symp. Secur. Privacy (SP)*, San Francisco, CA, USA, May 2019, pp. 175–192.
[29] A. Sapirshtein, Y. Sompolinsky, and A. Zohar, "Optimal selfish mining strategies in bitcoin," *Financial Cryptography and Data Security* (Lecture Notes in Computer Science), vol. 9603, J. Grossklags and B. Preneel, Eds. Berlin, Germany: Springer, 2016.
[30] K. Nayak, S. Kumar, A. Miller, and E. Shi, "Stubborn mining: Generalizing selfish mining and combining with an eclipse attack," in *Proc. IEEE Eur. Symp. Secur. Privacy (EuroSP)*, Mar. 2016, pp. 305–320.
[31] Y. Liu, Y. Hei, T. Xu, and J. Liu, "An evaluation of uncle block mechanism effect on ethereum selfish and stubborn mining combined with an eclipse attack," *IEEE Access*, vol. 8, pp. 17489–17499, 2020.
[32] X. Dong, F. Wu, A. Faree, D. Guo, Y. Shen, and J. Ma, "Selfholding: A combined attack model using selfish mining with block withholding attack," *Comput. Secur.*, vol. 87, Nov. 2019, Art. no. 101584.
[33] M. Davidson and T. Diamond, "On the profitability of selfish mining against multiple difficulty adjustment algorithms," IACR Cryptol. ePrint Arch., Tech. Rep. 94, Jan. 2020.
[34] J. Ke, P. Szalachowski, J. Zhou, Q. Xu, and Z. Yang, "IBWH: An intermittent block withholding attack with optimal mining reward rate," in *Proc. Int. Conf. Inf. Secur.*, in Lecture Notes in Computer Science, vol. 11723, 2019, pp. 3–24.
[35] L. Wan, D. Eyers, and H. Zhang, "Evaluating the impact of network latency on the safety of blockchain transactions," in *Proc. IEEE Int. Conf. Blockchain (Blockchain)*, Atlanta, GA, USA, Jul. 2019, pp. 194–201.
[36] X. Zhang, R. Qin, Y. Yuan, and F.-Y. Wang, "An analysis of blockchain-based bitcoin mining difficulty: Techniques and principles," in *Proc. Chin. Autom. Congr. (CAC)*, Xi'an, China, Nov. 2018, pp. 1184–1189.
[37] D. Kraft, "Difficulty control for blockchain-based consensus systems," *Peer-Peer Netw. Appl.*, vol. 9, no. 2, pp. 397–413, Mar. 2016.
[38] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Vienna Austria, Oct. 2016, pp. 3–16.
[39] J. Parra-Moyano, G. Reich, and K. Schmedders, "Urns filled with bitcoins: New perspectives on proof-of-work mining," *SSRN Electron. J.*, Jun. 2019, doi: 10.2139/ssrn.3399742.
[40] S. Cheng and S.-J. Lin, "Mining strategies for completing the longest blockchain," *IEEE Access*, vol. 7, pp. 173935–173943, 2019.