Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

RESEARCH ARTICLE

Block diffusion delay attack and its countermeasures in a Bitcoin network

Masahiro Sasabe* | Masanari Yamamoto | Yuanyu Zhang | Shoji Kasahara

[1]Graduate School of Science and Technology, Nara Institute of Science and Technology, Nara, Japan

Correspondence
*Masahiro Sasabe, Graduate School of Science and Technology, Nara Institute of Science and Technology, 8916-5 Takayama-Cho, Ikoma, Nara 630-0192, Japan. Email: m-sasabe@ieee.org

Summary

In the Bitcoin system, transactions and their collections (i.e., blocks) are distributed over a Peer-to-Peer (P2P) network (i.e., Bitcoin network) constructed by participating nodes. Each node maintains a distributed ledger (i.e., blockchain) consisting of retrieved blocks. Therefore, speedy block distribution over the Bitcoin network is essential for all nodes to reach a global consensus on the blockchain. On the other hand, Bitcoin clients are developed as open source software, and thus they can be modified by malicious users. Existing work has pointed out that an attacker can delay the block propagation between neighboring nodes by exploiting the regular timeout mechanism for unexpected slow block transfer caused by temporal network trouble. In this paper, we focus on the risk of block diffusion delay attack, where multiple attackers collude with a specific miner (i.e., a special node that aims to build blocks) to disturb the propagation of blocks generated by competing miners. Through simulation experiments, we first reveal that over 40% of honest nodes cannot normally retrieve a block when there are only 1% adversary nodes locating high-degree nodes in the system. To alleviate the block diffusion delay attack, we propose two kinds of countermeasures: a speedy recovery method from the interruption by adjusting the timeout value and a block retrieval node selection method based on the past download rate from each neighbor. Through simulation experiments, we show the countermeasures can effectively alleviate the risk.

KEYWORDS:
Bitcoin network, block diffusion delay attack, risk evaluation, countermeasure

1 | INTRODUCTION

Bitcoin,[1] the pioneer of cryptocurrency systems, has continuously been developed as open source software, e.g., Bitcoin Core.[2] Bitcoin realizes the remittance between individuals without a central authority such as countries and banks. In Bitcoin, the history of transactions is recorded as a distributed ledger (i.e., blockchain) in each node participating in the system. Each transaction is issued by a node and broadcasted over the Bitcoin network, which is an unstructured Peer-to-Peer (P2P) network.[3]

Some special nodes, called miners, collect the retrieved transactions and form a valid block through the proof-of-work, in which the miners need to find out (i.e., mine) an appropriate nonce in such a way that the hash value of the formed block is equal or lower than a certain target value. To acquire a reward from the system, the miner requires not only to succeed in mining a block

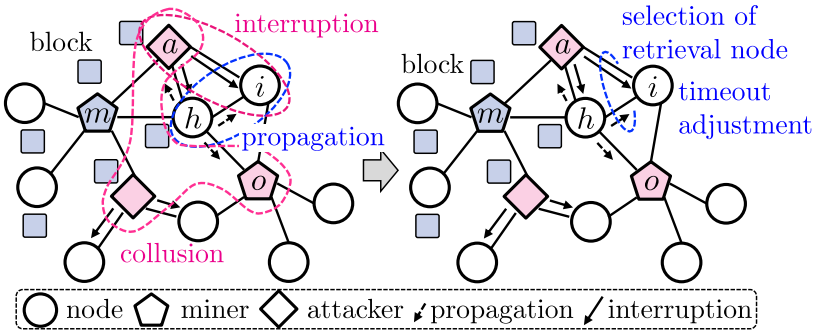


FIGURE 1 Overview of block diffusion attack and its countermeasures: selection of retrieval node and timeout adjustment.

but also to notify the block to all the other nodes. As a result, there are two kinds of competitions among miners, competitive block mining and competitive block diffusion.

Since the Bitcoin client, e.g., Bitcoin Core, has been developed as open source software, malicious users can modify it. As a result, various types of attacks have been pointed out for both the competitive block mining and competitive block diffusion.[4,5,6,7,8] The competitive block mining based on the hash calculation is potentially difficult to be tampered because of its random nature and validation from other nodes. On the other hand, the competitive block diffusion can be disturbed by network-based attacks. Specifically, block propagation delaying attack can interrupt the block propagation to a specific neighboring node by exploiting the regular timeout mechanism for block transfer.[9]

In this paper, we focus on the risk of block diffusion delay attack where multiple attackers colluding with a specific miner conduct the block propagation delay attack to disturb the diffusion of blocks generated by competitive miners. This situation is illustrated in the left of Figure 1 where an adversary a, which colludes with the miner o, tries to interrupt the propagation of a block, which was generated by the competing miner m, to its neighboring node i. Note that the existing work only focused on the risk of block propagation delay between neighboring nodes while we consider its extension to the network-wide attack. Through several simulation experiments, we first reveal how the number of attackers and their locations in the network affect the risk of block diffusion delay attack. In addition, we further propose two kinds of countermeasures against the block diffusion delay attack. The first one is timeout adjustment that considers the tradeoff between speedy recovery and undesirable interruption of normal block transfer. The second one is node selection for block retrieval based on the download rate estimation. The right of Figure 1 presents these countermeasures where the honest node i tries to adjust its timeout value and select a node to retrieve the block from its neighbors, i.e., a and h. The effectiveness of these countermeasures are also demonstrated through simulation experiments.

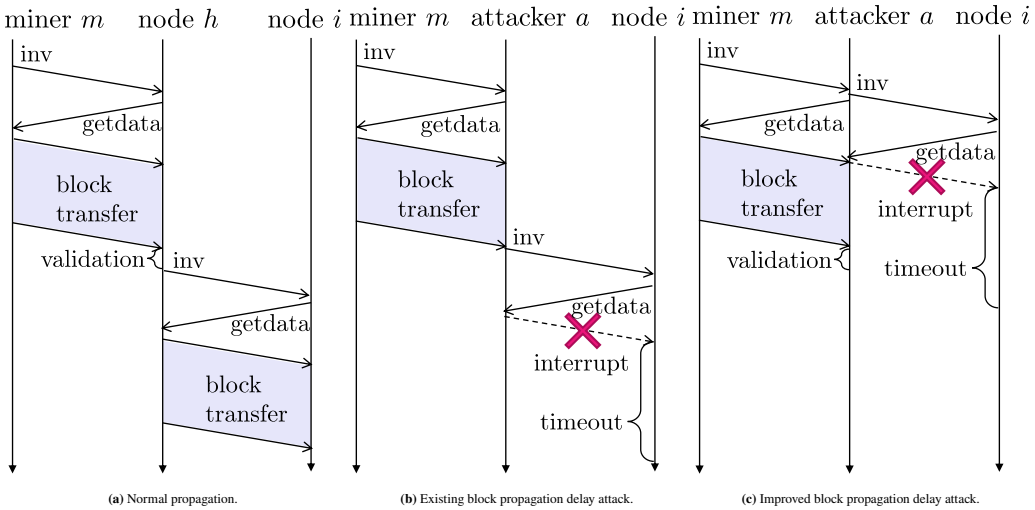
The rest of the paper is organized as follows. We first introduce the network aspect of the Bitcoin protocol in Section 2. Section 3 presents the block diffusion delay attack and evaluates the risk through simulation experiments. Sections 4 and 5 give the two kinds of countermeasures and show their effectiveness through simulation experiments. After introducing related work in Section 6, we finally conclude this paper in Section 7.

2 | BLOCK DIFFUSION OVER BITCOIN NETWORK

2.1 | Bitcoin network

The Bitcoin network is a P2P network, which is logically constructed by the participating nodes (called peers) running Bitcoin clients. Although there are various types and versions of Bitcoin clients, in this paper, we consider one of the major Bitcoin clients, Bitcoin Core, as a reference model. When a new node tries to join the Bitcoin network, it requires to obtain the information (i.e., IP addresses) of existing nodes. Since Bitcoin version 0.6, DNS is used as the bootstrapping mechanism where voluntary DNS servers maintain the IP addresses of existing nodes and return them in response to the requests from Bitcoin nodes. Once nodes join the Bitcoin network, they can also ask their neighbors for a list of available nodes through `addr` messages.

Each node manages a table of neighbor candidates based on the retrieved `addr` messages such that the neighboring nodes are well distributed over the IP address space. The Bitcoin network is originally designed to be a random network with low variance of node degree, i.e., the number of neighboring nodes. Each node initiates outgoing connections to up to eight nodes. On the



(a) Normal propagation. **(b)** Existing block propagation delay attack. **(c)** Improved block propagation delay attack.

FIGURE 2 Block propagation and interruption between neighboring nodes.

other hand, it can also accept incoming connections from others such that the sum of the number of incoming and outgoing connections is limited to 125.[10]

Because of the security and anonymity concerns, the Bitcoin system intentionally aims to conceal the structure of the Bitcoin network.[3] In recent years, several researchers tried to reveal the Bitcoin network.[11,12] Miller et al. first analyzed the topology of the real Bitcoin network by using the method called AddressProve.[11] They confirmed that most nodes had degrees in the range of 8–12, which were close to the maximum number of outgoing connections. However, they also revealed that several nodes had extremely large degrees, which exceeded the maximum number of incoming/outgoing connections, i.e., 125. These high-degree nodes were considered as groups of miners (i.e., mining pools) or wallet service operators. In the simulation experiments (Section 3.2), we construct a Bitcoin network considering the degree distribution observed in Miller et al.[11]

2.2 | Block diffusion

Figure 2a shows how a block is propagated between miner m, node h, and node i in Figure 1 according to the Bitcoin protocol. When the miner m generates a new block, it first notifies its neighboring node h of an inventory (`inv`) message, which contains the SHA-256 hash value of the block. When the node h has not received the block corresponding to the hash value yet, it will request the block from the miner m through a `getdata` message. Since the size of an `inv` message is much smaller than that of an actual block (e.g., 1 MB), this confirmation process can avoid the bandwidth consumption for unnecessary block transfer. On receiving the `getdata` message, the minder m transfers the corresponding block. After receiving the block, the node h verifies the legitimacy of the transactions in the block and the hash value. If the node h can confirm the validity of the block, it adds the block to its own blockchain and sends an `inv` message to its neighbor i. The block diffusion over the Bitcoin network is realized by the repetition of the block propagation between neighboring nodes.

3 | RISK OF BLOCK DIFFUSION DELAY ATTACK

3.1 | Block diffusion delay attack

In the block diffusion delay attack, multiple attackers colluding with a specific miner conduct the block propagation delay attack[9] to disturb the diffusion of blocks generated by competitive miners. We first introduce the details of the existing block propagation delay attack between neighboring nodes in Section 3.1.1. In this paper, we assume that individual attackers independently conduct the block propagation delay attack. The collaboration among attackers would yield more complicated and sophisticated attacks but they are future direction. Since the locations of attackers will affect the impact of the block diffusion delay risk, we further consider three kinds of location patterns of attackers in Section 3.1.2.

3.1.1 | Block propagation delay attack between neighbors

In Figure 2a, we only focused on the block propagation between three nodes, i.e., miner m, node h, and node i in Figure 1. In the Bitcoin network, the node i may obtain `inv` messages for the same block from other neighbor a. In such a case, the node i requests the block from a single neighbor in a First-Come First-Served (FCFS) manner, so as to achieve speedy block retrieval without bandwidth waste. Since the block transfer may require much time, due to temporal network/node trouble, the Bitcoin protocol adopts a timeout mechanism. In Miller et al,[9] it was pointed out that the timeout value was set to be 20 [min].[1] Note that this timeout value has been revised to be 10 [min].[2] When the node i causes the timeout, it requests the block transfer from some other neighbor randomly selected from neighbors that sent the `inv` message to it.

Miller et al proposed the block propagation delay attack where an adversary delayed the block propagation to a specific neighbor by exploiting the timeout mechanism.[9] Figure 2b illustrates the process of the block propagation delay attack from the attacker a to the node i. Focusing on the interaction between the miner m and the attacker a, we confirm that it is almost the same as that between the miner m and the node h, which was explained in Section 2.2. The difference is that the attacker a sends an `inv` message to its neighbor i without verifying the retrieved block. To succeed in the attack, the attacker a needs to be the first node who sends the `inv` message to the neighboring node i. By skipping the validation process, the attacker a tries to increase the successful probability of its attack. After receiving a `getdata` message from the node i, it delays the block transfer to the node i until the timeout occurs.

In this paper, we extend this attack to increase the attack speed as in Figure 2c. When the attacker a receives an `inv` message from the miner m, it immediately sends the `inv` message to node i. If the attacker a receives a `getdata` message from the node i, it first waits for its own block retrieval. If the retrieved block is owned by the colluding miner, i.e., miner o, the attacker a immediately starts block transfer to the node i. Otherwise, it delays the block transfer to the node i until the timeout occurs, as in the original block propagation delay attack.

3.1.2 | Location patterns of attackers

We can expect that the impact of block diffusion delay attack may depend on the locations of attackers in the network. In what follows, we consider three kinds of location patterns of attackers. As mentioned in Section 2.1, the connection updating process of the Bitcoin protocol follows a randomized fashion, and thus we first consider a *random location pattern* where attackers are randomly placed in the network. Considering the rationality of attackers, we further consider two possible location patterns, which are expected to increase the impact of the block diffusion delay attack. Attackers will be required to exploit the Bitcoin connection updating process to be located at their desiring positions in the network. We expect that this can be achieved in the way similar to that used in Eclipse attack.[13]

We regard the Bitcoin network as a graph $G = (\mathcal{N}, \mathcal{E})$ where $\mathcal{N} = \{1, \dots, N\}$ is the set of N nodes and $\mathcal{E}$ is that of links. If an attacker becomes a high-degree node, it can disturb the block propagation to its many neighbors. Therefore, we consider a *degree-based location pattern* where attackers are placed in descending order of their degree centrality. The degree centrality $C_{\mathrm{D}}(i)$ of node i is given as follows:[14]

$$C_{\mathrm{D}}(i) = \frac{k_i}{N-1},$$

where k_i represents the degree of node i.

As mentioned in Section 3.1, an attacker can conduct the delay attack only after the `inv` reception. If an attacker is included in many block propagation routes, it may quickly acquire `inv` messages. Therefore, we further consider a *betweenness-based location pattern* where attackers are placed in descending order of their betweenness centrality. The betweenness centrality $C_{\mathrm{B}}(i)$ of node i is the possibility that node i is included in the shortest paths between two arbitrary nodes:[14]

$$C_{\mathrm{B}}(i) = \frac{2}{(N-1)(N-2)} \sum_{j \in \mathcal{N}, j \neq i} \sum_{k \in \mathcal{N}, k < j} \frac{n_{j,k,i}}{n_{j,k}},$$

where $n_{j,k}$ represents the number of shortest paths from node j to node k and $n_{j,k,i}$ represents the number of them including node i.

[1] https://github.com/bitcoin/bitcoin/pull/5608
[2] https://github.com/bitcoin/bitcoin/pull/7832

3.2 | Simulation results

In this section, we first explain the Bitcoin network topology, simulator, and simulation scenario for evaluation. Next, we evaluate the risk of the block diffusion delay attack in terms of the number of attackers and their location patterns.

3.2.1 | Simulation settings

We prepare a Bitcoin network for evaluation according to the degree distribution observed in Miller et al[11] and settings used in the Bitcoin-Simulator.[15] There are 6000 nodes including 16 miners. We set the geographical location of each node based on the statistical information of the geographical distribution of nodes in Bitnodes.[16] We also set the propagation delay and bandwidth between neighboring nodes based on their geographical locations and settings of the Bitcoin-Simulator. As for the propagation delay, its average is 112.52 [ms]. On the other hand, the maximum bandwidth from node i to node j ($i, j \in \mathcal{N}$), $R_{i,j}^{\max}$, is defined by $\min(U_i, D_i, U_j, D_j)$, where U_v and D_v ($v = i, j$) are the upload capacity and download capacity of node v, respectively. The average $\overline{U}$ of U_i (resp. $\overline{D}$ of D_i) among nodes is 0.82 (resp. 2.95) [MB/s]. Note that the actual download rate from node i to node j, $r_{i,j}$, will decrease with the number of nodes that aim to retrieve the block from node i. Actually, we observe the actual average download rate is 0.2 [MB/s] in the following evaluations. As mentioned in Section 2, miners tend to have high degree, and thus we set the degree of miners to follow a uniform distribution between 700 and 800. On the other hand, the degree of regular nodes follows the degree distribution in Miller et al.[11]

To focus on the fundamental characteristics of the block diffusion delay attack, we create a discrete-event simulator written in Java language. We set the fraction of attackers, α, to be 0, 0.01, 0.05, 0.1, and 0.15. The degree centrality and betweenness centrality are calculated by NetworkX.[17] We assume that all nodes hold the same blockchain at the initial state ($t = 0$). At the same time, a miner randomly chosen from the 16 miners succeeds in block generation. We focus on the diffusion and delay of this block over the network.

We set the block size to be 0.534 [MB], which is the mean size of 10,000 blocks generated from May to November 2015.[18] This period is the same as the observation period in Arthur et al.[15] The block verification time and timeout value are set to be 0.1 [s] and 600 [s], respectively. Table 1 summarizes the parameter settings. The following results are the average of 1,600 independent simulation runs.

3.2.2 | Impact of the number of attackers

First, we reveal the impact of the fraction of attackers, α, on the block diffusing delay risk under the random location pattern. Figure 3 presents the transition of the fraction of honest nodes completing block retrieval (*synced nodes*), when $\alpha = 0, 0.01, 0.05, 0.1$, and 0.15. We mainly focus on the time period $[0, 600]$ by taking account of the average block generating interval of 600 [s].

We first observe that the normal block diffusion ($\alpha = 0$) is speedily achieved and all nodes can retrieve blocks until $t = 186$. We also confirm that a small number of attackers can drastically decrease the initial block diffusion ratio and the degree of the delay risk grows with the increase of α. In the time period $[0, 600]$, the block diffusion seems to converge at a certain diffusion ratio depending on α but this is because of the relatively long timeout value, i.e., 600 [s], which is the same as the average interval between two successive block generation in the current Bitcoin protocol.

As for the long-term trend of the block diffusion, we can confirm that the fraction of synced nodes almost increases in a stepwise manner because nodes recovered from the attacks retrieve blocks or receive attacks every timeout of 600 [s]. Note that the

TABLE 1 Simulation settings.

Parameter	Value
Number of nodes, N	6000
Number of miners, M	16
Ratio of attackers, α	0, 0.01, 0.05, 0.1, 0.15
Block size B_{s}	0.534 [MB]
Block validation time T_{v}	0.1 [s]
Timeout value T_{o}	600 [s]

actual block diffusion will become more complex, e.g., fork occurrence, because a new block may be generated around $t = 600$. The detail analysis of such complex situations is future work.

3.2.3 | Impact of location patterns of attackers

Next, we evaluate the impact of the location patterns of attackers on the block diffusion delay risk. Figure 4 shows the transition of the fraction of synced nodes under three location patterns when $\alpha = 0.01$. We first observe that both the degree-based location pattern and betweenness-based location pattern have higher attack risk even under the same fraction of attackers, compared with the random location pattern. In particular, the degree-based location pattern shows the highest risk among the three location patterns. Since the degree-based location pattern only requires attackers to acquire more neighbors, it can be easily achieved compared with the betweenness-based location pattern.

In succeeding sections, we further propose countermeasures against the block diffusion delay attack.

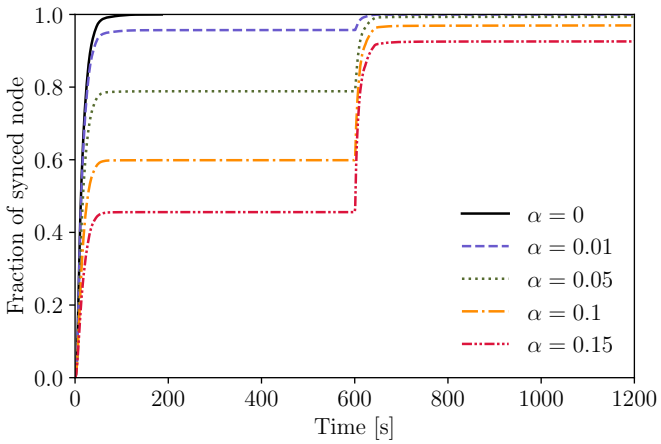


FIGURE 3 Impact of fraction of attackers, α, on transition of the fraction of synced nodes (location pattern of attackers: random, $t = [0, 1200]$).

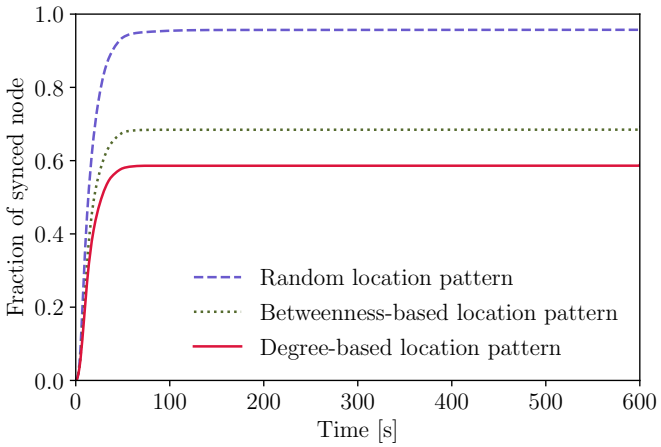


FIGURE 4 Impact of attackers' location patterns on transition of the fraction of synced nodes.

4 | SPEEDY RECOVERY BASED ON TIMEOUT ADJUSTMENT

In this section, we propose a speedy recovery method in response to the attack reception by appropriately adjusting the timeout value.

4.1 | Tradeoff between speedy recovery and imperfect block download

From the viewpoint of the speedy recovery after the attack reception, we should shorten the timeout value. On the other hand, if we set the timeout value to be too short, the normal block transfer would also be interrupted, which results in the bandwidth waste. Therefore, setting the timeout value to be slightly larger than the time required for the normal block transfer can achieve the speedy recovery without disturbing the normal block transfer.

In what follows, we show the relationship between the timeout value and the amount of bandwidth waste caused by the download interruption. For simplicity of description, we define the set of honest nodes as $\mathcal{N}_{\mathrm{P}}$. Each honest node $i \in \mathcal{N}_{\mathrm{P}}$ selects a node $n(i)$ from its own set of neighboring nodes $\mathcal{N}_i$ as a block retrieval node. At this time, the node $n(i)$ may receive block retrieval requests from not only the node i but also other neighbors of $n(i)$. In this case, the node $n(i)$ will process these requests in an FCFS manner. As a result, the retrieval request from each honest node $i \in \mathcal{N}_{\mathrm{P}}$ to node $n(i)$ results in the following three cases:

1. Normal block retrieval;
2. Timeout occurrence before block transfer; and
3. Timeout occurrence during block transfer.

We define the amount of bandwidth waste caused by the timeout occurrence during the block transfer from node $n(i)$ to node i as $d_{n(i),i}$. In case 1 and case 2, $d_{n(i),i} = 0$. In case 3, $d_{n(i),i}$ is given by the amount of data transferred from the time that the block transfer starts (i.e., $t^{\mathrm{st}}_{n(i),i}$) to the time that the timeout occurs (i.e., $t^{\mathrm{to}}_{n(i),i}$): $d_{n(i),i} = r_{n(i),i}(t^{\mathrm{to}}_{n(i),i} - t^{\mathrm{st}}_{n(i),i})$. Here $r_{n(i),i}$ ($r_{n(i),i} \leq R^{\max}_{n(i),i}$) represents the transfer rate from retrieval node $n(i)$ to node i.

As a result, the total amount of bandwidth waste, d_{w}, is given as $d_{\mathrm{w}} = \sum_{i \in \mathcal{N}} d_{n(i),i}$. Therefore, our goal is to set the timeout value as small as possible such that $d_{\mathrm{w}} = 0$ is satisfied.

4.2 | Analysis of lower bound of iimeout

Suppose that each node can independently retrieve the block. Given the maximum bandwidth between each pair of two neighboring nodes $i, j \in \mathcal{N}$, $R^{\max}_{i,j}$, we can derive the lower bound of the timeout value to achieve $d_{\mathrm{w}} = 0$ as follows. Note that the actual transfer rate will dynamically change depending on the background traffic and the block retrieval node selection of each node. There are various kinds of bandwidth estimation methods.[19] First, each honest node $i \in \mathcal{N}_{\mathrm{P}}$ has the maximum bandwidth from each neighboring node $j \in \mathcal{N}_i$ as $R^{\max}_{j,i}$, and thus it will select a neighboring node $\arg\max_{j \in \mathcal{N}_i} R^{\max}_{j,i}$ as the block retrieval node $n(i)$, from which it can expect the maximum block transfer rate. At this time, the lower bound of the block transfer time from retrieval node $n(i)$ to node i, $t^{\min}_i$, is given as $t^{\min}_i = B_{\mathrm{s}}/R^{\max}_{n(i),i}$. If we use an identical timeout value t_{o} in the entire system, we can achieve the speedy recovery without bandwidth waste ($d_{\mathrm{w}} = 0$) by setting t_{o} to be slightly larger than $\max_{i \in \mathcal{N}} t^{\min}_i$, which is the block transfer time from the node that requires the longest time to retrieve the block.

4.3 | Simulation results

4.3.1 | Simulation settings

We basically use the same evaluation settings used in Section 3.2.1, except the following settings. We first set the fraction of attackers, α, to be 0 and 0.01, and use the degree-based location pattern, which has the highest risk of block diffusion delay. We change the timeout value t_{o} in the range of $[20, 600]$. The following results are the average of 1600 independent simulation runs for each timeout value. As for the evaluation criteria, we use the fraction of synced nodes and the total amount of bandwidth waste.

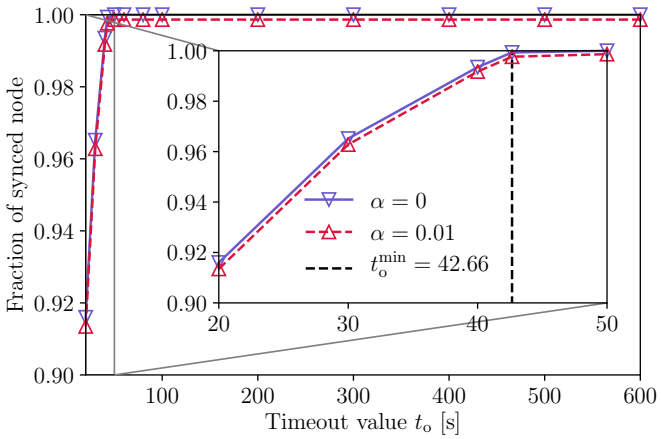


FIGURE 5 Impact of timeout value, t_o, on transition of the fraction of synced nodes (location pattern of attackers: degree-based, $t_o = [20, 600]$).

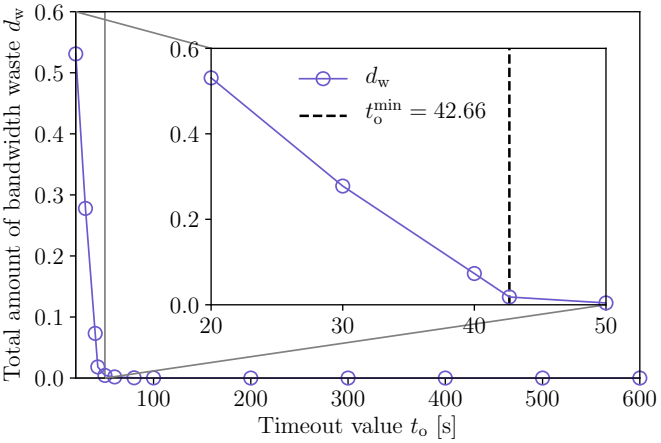


FIGURE 6 Impact of timeout value, t_o, on total amount of bandwidth waste, d_w (location pattern of attackers: degree-based, $t_o = [20, 600]$).

4.3.2 | Impact of timeout adjustment

Figure 5 shows the relationship between the timeout value and the fraction of synced nodes in case of $\alpha = 0, 0.01$. We first observe that all the honest nodes can retrieve the block if there is no adversary ($\alpha = 0$) and $t_o \geq 50$. On the other hand, when there are attackers ($\alpha = 0.01$), we confirm that 0.5% honest nodes cannot retrieve the block even under the large timeout value, e.g., 600. This is because there are some honest nodes that have a limited number of neighbors and all of them are adversaries.

Next, focusing on the range of $20 \leq t_o \leq 50$ (the enlarged graph), we can confirm that the fraction of synced nodes deteriorates with the decrease of the timeout value, regardless of the existence of attackers. This is because the normal block transfer between nodes is also interrupted by the insufficient timeout value. This can also be confirmed from Figure 6, which shows the relationship between the timeout value t_o and the total amount of bandwidth waste d_w when $\alpha = 0.01$. Note that the total amount of bandwidth waste is normalized by $B_S(1 - \alpha)(N - 1)$, which is the total amount of data that should be sent to all honest nodes.

In case of the evaluation network, the lower bound of the timeout, t_o^{min}, is derived as 42.66, which is also shown in the enlarged graphs of Figs. 5 and 6. We can confirm that the analytical lower bound of the timeout is close to the simulation-based appropriate timeout value, i.e., 50, even under the simplified assumption.

5 | SELECTION OF BLOCK RETRIEVAL NODE BASED ON ESTIMATED DOWNLOAD RATE

In this section, we propose a block retrieval node selection method based on estimated download rate as a countermeasure to suppress the reception of the block diffusion delay attack.

5.1 | Overview

In order to realize the normal and speedy block diffusion, each node needs not only to avoid receiving the attack from attackers but to select a good neighboring node for the high-speed block download. We should note here that each node does not initially have any information about download rate of each neighbor. Since the Bitcoin system continuously generates blocks at an average interval of 10 [min], each node has opportunities to observe the download rate from neighbors selected as block retrieval nodes. These obtained knowledge can be used for the future selection of block retrieval nodes. As for the similar work, Aoki and Shudo aimed at speedy diffusion of transactions and blocks by proposing a method where each node refines its neighbors to be physically close nodes based on observed propagation delay of `inv` messages.[20] Recall that the block diffusion delay attack enables adversaries to send `inv` messages more speedily than honest nodes, as mentioned in Section 3. This means that this method will not work well in case of the block diffusion delay attack because it tends to select adversaries as neighbors, due to speedy reception of `inv` messages.

In case of the block retrieval, the download time (rate) has a greater impact on the retrieval time than the propagation delay because of the large block size. Therefore, in this section, we propose a block retrieval node selection method based on the past block download rate from each neighboring node, so as to achieve both avoiding attacks and speedy block retrieval. In what follows, for simplicity, we assume that the Bitcoin network is static and the set of neighbors of each node $i \in \mathcal{N}$, $\mathcal{N}_i$, does not change. Since both the average block generation interval and the default timeout value are 10 [min], we assume that each node can select at most one neighbor as its block retrieval node at each block generation event.

5.2 | Download rate estimation

At each block generation event of average 10 minute interval, each node can try to retrieve a corresponding block from one of its neighboring nodes. As a result, it can observe the actual block download rate from the selected neighbor. Note that it would observe zero or very slow download rate from the adversary node. At the initial state, each node does not have the information about the download rate from each neighbor but it can gradually learn the download rate information during the continuous block generation and diffusion events. Note that the download rate from a neighbor may fluctuate depending on the background traffic and/or other nodes' retrieval requests to the neighbor.

Considering the above points, we propose the following download rate estimation method. At the initial state, each honest node $i \in \mathcal{N}_\mathrm{P}$ does not hold the information about the download rate from each neighboring node $j \in \mathcal{N}_i$, and thus its estimated download rate $\widehat{r}^0_{j,i}$ is initialized as follows:

$$\widehat{r}^0_{j,i} = B_\mathrm{s}/T_\mathrm{o} \qquad i \in \mathcal{N}_\mathrm{P}, j \in \mathcal{N}_i,$$

which is equal to the lower bound of the download rate required for completing download without the timeout occurrence.

We focus on the lth ($l = 1, \ldots, 10$) block diffusion in continuous block generation events. When the honest node $i \in \mathcal{N}_\mathrm{P}$ selects a neighboring node $j \in \mathcal{N}_i$ as its block retrieval node $n(i)$ at the lth block diffusion (Section 5.3 will give the detail), i can observe the download rate $r^l_{n(i),i}$ from $n(i)$. On the other hand, as mentioned above, each node can send a block retrieval request to at most one neighboring node at each block generation event, and thus the node i cannot observe the download rate from neighboring nodes $j \in \mathcal{N}_i \setminus \{n(i)\}$, which are not selected as the block retrieval node.

Given these limited information, each node i calculates the estimated download rate $\widehat{r}^l_{j,i}$ from each neighbor j:

$$\widehat{r}^l_{j,i} = \beta r^l_{j,i} + (1-\beta)\widehat{r}^{l-1}_{j,i} \qquad i \in \mathcal{N}_\mathrm{P}, j \in \mathcal{N}_i,$$

which adopts Exponential Moving Average (EMA) to improve the adaptability to download rate fluctuation where β represents a smoothing coefficient that ranges in $[0, 1]$. Note that each node has $r^l_{n(i),i}$ for the block retrieval node $n(i)$ but does not have $r^l_{j,i}$ for other neighbors $j \in \mathcal{N}_i \setminus \{n(i)\}$. As for the neighbors $j \in \mathcal{N}_i \setminus \{n(i)\}$, node i substitutes the latest estimated download rate $\widehat{r}^{l-1}_{j,i}$ into $r^l_{j,i}$, which results in $\widehat{r}^l_{j,i} = \widehat{r}^{l-1}_{j,i}$.

5.3 | Selection of block retrieval node

Each honest node $i \in \mathcal{N}_\mathrm{P}$ can manage and update the estimated download rate $\widehat{r}^{l}_{j,i}$ from each neighboring node $j \in \mathcal{N}_i$, with the help of the download rate estimation method in Section 5.2. According to $\widehat{r}^{l}_{j,i}$, the honest node i cannot only avoid sending the retrieval requests to attackers but select a neighboring node with the expectation of high-speed block download. In what follows, we explain the details of the block retrieval node selection method based on the estimated download rate.

At the beginning of the lth block diffusion, each honest node $i \in \mathcal{N}_\mathrm{P}$ holds the latest estimated download rate $\widehat{r}^{l-1}_{j,i}$ from each neighboring node $j \in \mathcal{N}_i$. According to this information, the node i can classify its own set of neighboring node $\mathcal{N}_i$ into the following three subsets.

- $\mathcal{N}_i^+ = \{j \in \mathcal{N}_i \mid \widehat{r}^{l-1}_{j,i} > B_\mathrm{s}/T_\mathrm{o}\}$: The set of neighbors that are expected to be honest because of the past normal block retrieval.
- $\mathcal{N}_i^- = \{j \in \mathcal{N}_i \mid \widehat{r}^{l-1}_{j,i} < B_\mathrm{s}/T_\mathrm{o}\}$: The set of neighbors from which the node i attempted to retrieve a block in the past but failed, due to the timeout. Note that $\mathcal{N}_i^-$ not only includes adversaries but also honest nodes with very slow download rate. From the viewpoint of the speedy block diffusion, both types of nodes are not suitable for the block retrieval node.
- $\mathcal{N}_i^* = \{j \in \mathcal{N}_i \mid \widehat{r}^{l-1}_{j,i} = B_\mathrm{s}/T_\mathrm{o}\}$: The set of neighbors from which the node i has never attempted to retrieve a block yet. Note that $\mathcal{N}_i^*$ might include the case where all past download rates are $B_\mathrm{s}/T_\mathrm{o}$. Since this case rarely occurs in the long term, we ignore it to make the method as simple as possible.

Suppose that the node i receives an `inv` message for the block from its neighboring node j. If $j \in \mathcal{N}_i^+ \cup \mathcal{N}_i^*$, the node i can expect to normally retrieve the block from the neighbor j, and thus i selects j as the block retrieval node $n(i)$. Otherwise, i.e., $j \in \mathcal{N}_i^-$, the neighbor j may be an attacker or an honest node with very slow download rate, and thus the node i skips the `inv` message from j and waits for the next arrival of an `inv` message from another neighboring node.

As a result, the node i will experience one of the following cases at the lth block diffusion.

1. Normal block retrieval;
2. Timeout occurrence before block transfer;
3. Timeout occurrence during block transfer; or
4. Failure in finding a block retrieval node.

In case 1, node i updates the estimated download rate from block retrieval node $n(i)$, $\widehat{r}^{l}_{j,i}$, based on the actual download rate $r^{l}_{j,i}$. Otherwise, it updates $\widehat{r}^{l}_{j,i}$ by using $r^{l}_{j,i} = \widehat{r}^{l-1}_{j,i}$.

With the above mechanism, we can expect that each honest node gradually achieves the normal block retrieval (case 1) by selecting an appropriate retrieval node even under mixed neighbors of honest nodes and adversaries. Note that some honest node might always fail in block retrieval if all of the neighbors are adversaries. In such cases, it needs to update the set of neighbors appropriately, which is our future work.

5.4 | Simulation results

5.4.1 | Simulation settings

We basically use the same evaluation settings used in Section 3.2.1, except the following settings. Considering the fluctuation of the network bandwidth depending on the background traffic, we set the maximum download rate from node j to node i ($i \in \mathcal{N}_\mathrm{P}, j \in \mathcal{N}_i$) at the lth ($l = 1, \ldots, 10$) block diffusion, $R^{\max,l}_{i,j}$, to be $\min(U^l_i, D^l_i, U^l_j, D^l_j)$, where D^l_υ and U^l_υ ($\upsilon = i, j$) are the upload capacity and download capacity of node υ at lth block diffusion event, respectively. In this section, we assume that the download capacity D^l_i of node i at the lth block diffusion fluctuates according to the normal distribution with average of D_i and standard deviation of γD_i ($0 \le \gamma \le 1$). Similarly, the download capacity U^l_i of node i at the lth block diffusion also fluctuates according to the normal distribution with average of U_i and standard deviation of γU_i.

Considering that both the average block generation interval and the default timeout value are 10 [min], we assume that each node can select at most one neighbor as its block retrieval node at each block diffusion event. We set the fraction of attackers, α, to be 0.01, and use the degree-based location pattern, which has the highest risk of block diffusion delay. Through the preliminary experiments, we set the smoothing coefficient of EMA, β, to be 0.5. In what follows, we regard ten consecutive block generation/diffusion events as one trial and show the average of 100 independent simulation runs.

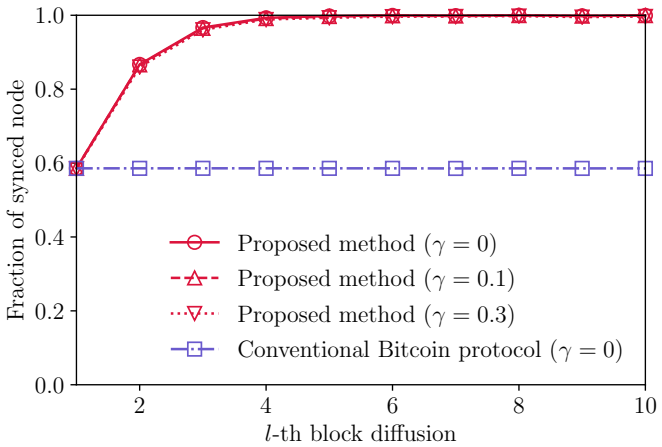


FIGURE 7 Impact of the number of block diffusion, l, on transition of the fraction of synced nodes (location pattern of attackers: degree-based).

5.4.2 | Impact of selection of block retrieval node

We first examine the effectiveness of the block retrieval node selection method based on the estimated download rate through simulation experiments in the ideal case where the network bandwidth of each link is static ($\gamma = 0$). For the comparison purpose, we also evaluate the conventional Bitcoin protocol where each node requests the block from a neighbor that first sent an `inv` message.

Figure 7 illustrates how the fraction of synced nodes changes with the progress of the block diffusion events. First, there is no performance difference between the two methods at the first block diffusion ($l = 1$). This is because all the honest nodes aim to retrieve blocks from the first neighbors sending `inv` messages to them. We observe that the fraction of synced nodes is about 60%, which indicates that about 40% of honest nodes cannot retrieve blocks, due to the attacks. In case of the conventional Bitcoin protocol with $l \geq 2$, the fraction of synced nodes almost never changes because of the continuous requests to the first response neighbors. On the other hand, in case of the proposed method with $l \geq 2$, we can confirm that honest nodes speedily succeed in avoiding the attacks by learning the download rate from retrieval nodes through past block diffusion events. In particular, the proposed method enables almost all the nodes to retrieve blocks when $l \geq 4$. Note that 0.5% of honest nodes cannot retrieve blocks even in the range of $l \geq 4$ because they are surrounded by adversaries. To tackle this problem, we plan to propose a dynamic updating method of neighboring nodes in future work.

We also find that the proposed method is adaptive to the bandwidth fluctuation because the fraction of synced nodes with $\gamma = 0.1, 0.3$ shows almost the same result as that with $\gamma = 0$.

In Figure 7, we only focus on the fraction of synced nodes, which can retrieve the block within the default timeout value, 10 [min], at each block diffusion event. On the other hand, considering the speedy block diffusion, we also need to focus on the block retrieval time of each node. Figure 8 illustrates how the average block retrieval time changes with the progress of the block diffusion events. In case of $\gamma = 0$, we observe that the average block retrieval time becomes 16–17 (resp. 17) [s] in the proposed method (resp. conventional Bitcoin protocol), and thus the performance difference seems to be limited. If the bandwidth fluctuation increases ($\gamma = 0.3$), we confirm that the average block retrieval time of the proposed method slightly increases and becomes unstable.

From the above results, we can confirm that the proposed method enables almost all the honest nodes to retrieve blocks normally and speedily in the progress of continuous block diffusion events.

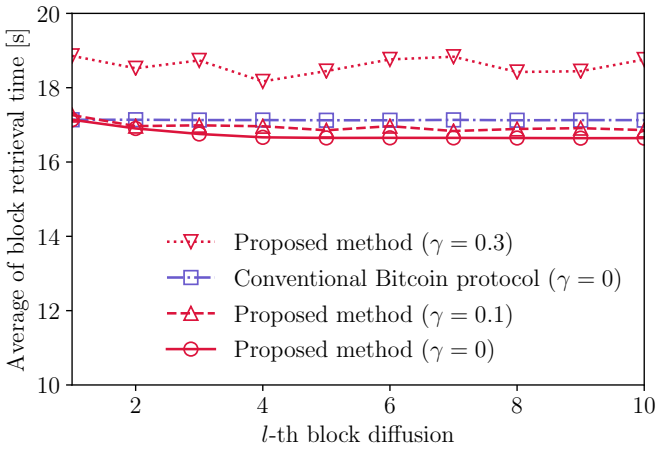


FIGURE 8 Impact of the number of block diffusion, l, on average of block retrieval time (location pattern of attackers: degree-based).

6 | RELATED WORK

Attacks exploiting the Bitcoin protocol are classified into two types: computation-based attacks[21,22,23,24,25] and network-based attacks.[9,13,26,27] There are several survey papers that summarize the attacks and their countermeasures.[6,7,8] The computation-based attacks can be 51% attack,[23] double spending,[21,22] and selfish mining.[24,25] These attacks require a huge amount of computing power, and thus it is not easy to realize them.

On the other hand, the network-based attacks exploit the vulnerabilities of the Bitcoin network. In addition, it has also been pointed out that network-based attacks would increase the feasibility of the computation-based attacks.[9,13,26,28] Heilman et al. proposed Eclipse attack where an adversary monopolizes all connections of a victim node to isolate it from the Bitcoin network.[13] Nayak et al. proposed an attack combining Eclipse attack and selfish mining.[28] Apostolaki et al. proposed Border Gateway Protocol (BGP)[29] hijacking attack where adversaries advertise incorrect BGP routing information in the underlying IP network, i.e., the Internet, to disturb the information propagation over the Bitcoin network.[26]

Gervais et al. pointed out the risk of block propagation delay attack where an adversary can delay the block propagation to its neighboring node by exploiting the regular timeout mechanism for block transfer.[9] Walck et al. proposed TendrilStaller, which is the combination of the Eclipse attack and the block propagation delay attack.[27] Recently, the interruption risk of generalized competitive information diffusion was modeled[30], with the help of the mathematical epidemiology[31]. These studies also mentioned the possibilities of some countermeasures against the block propagation delay attack, e.g., shortening block download timeout and neighbor selection, but they did not give the detail mechanisms.

The Bitcoin protocol specification has also been reviewed to reduce the block diffusion delay. Bitcoin Improvement Proposals (BIP) 130[32] introduced the "sendheaders message mechanism" where a node sends a `headers` message including the headers of new blocks instead of an `inv` message so as to notify the new block announcement. BIP 152[33] proposed Compact Block Relay (CBR) where the receiver node builds the block rather than retrieving the block itself. However, these modifications still rely on the pull-based diffusion mechanism, and thus the risk of block diffusion delay attack is remaining.

7 | CONCLUSION

In this paper, we have focused on the risk of block diffusion delay attack where one or more attackers colluding with a specific miner aim to disturb the propagation of blocks generated by the competitors of the miner. We have first evaluated how the number of attackers and their locations in the Bitcoin network affect the risk of block diffusion delay attack through simulation experiments considering the features of the actual Bitcoin network. In particular, we have revealed that only 1% adversaries can prevent more than 40% of honest nodes from retrieving blocks within the average block generation interval when they are located at high-degree nodes in the Bitcoin network.

To tackle the block diffusion delay attack, we have proposed two kinds of countermeasures. The first one is a speedy recovery method reception by appropriately adjusting the timeout value. The second one is a block retrieval node selection method based on estimated download rate. We have also shown that these countermeasures effectively alleviate the block diffusion delay risk. In future work, we plan to consider the situation where the Bitcoin network dynamically changes.

DATA AVAILABILITY STATEMENT

Research data are not shared.

ACKNOWLEDGMENTS

This work was supported in part by JSPS KAKENHI (19H01103 and 19KT0045), Japan.

References

1. Nakamoto S. Bitcoin: A Peer-to-Peer Electronic Cash System. available at https://bitcoin.org/bitcoin.pdf; 2008.
2. BitcoinCore. available at https://bitcoincore.org/; .
3. Tschorsch F, Scheuermann B. Bitcoin and Beyond: A Technical Survey on Decentralized Digital Currencies. *IEEE Security and Privacy* 2016; 18(3): 2084–2123.
4. Lin IC, Liao TC. A Survey of Blockchain Security Issues and Challenges. *International Journal of Network Security* 2017; 19: 653-659.
5. Conti M, E SK, Lal C, Ruj S. A Survey on Security and Privacy Issues of Bitcoin. *IEEE Communications Surveys and Tutorials* 2018; 20(4): 3416–3452.
6. Neudecker T, Hartenstein H. Network Layer Aspects of Permissionless Blockchains. *IEEE Communications Surveys and Tutorials* 2019; 21(1): 838–857.
7. Hasanova H, Baek Uj, Shin Mg, Cho K, Kim MS. A Survey on Blockchain Cybersecurity Vulnerabilities and Possible Countermeasures. *International Journal of Network Management* 2019; 29(2): 1–36.
8. Saad M, Spaulding J, Njilla L, et al. Exploring the Attack Surface of Blockchain: A Comprehensive Survey. *IEEE Communications Surveys and Tutorials* 2020; 22(3): 1977–2008.
9. Gervais A, Ritzdorf H, Karame GO, Capkun S. Tampering with the Delivery of Blocks and Transactions in Bitcoin. In: ACM SIGSAC Conference on Computer and Communications Security. ACM. ; 2015: 692–705.
10. Running A Full Node - Bitcoin. available at https://bitcoin.org/en/full-node; .
11. Miller A, Litton J, Pachulski A, et al. Discovering Bitcoin's Public Topology and Influential Nodes. available at http://cs.umd.edu/projects/coinscope/coinscope.pdf; 2015.
12. Essaid M, Park S, Ju HT. Bitcoin's Dynamic Peer-to-Peer Topology. *International Journal of Network Management* 2020; 30(5): 1–21.
13. Heilman E, Kendler A, Zohar A, Goldberg S. Eclipse Attacks on Bitcoin's Peer-to-Peer Network. In: USENIX Security Symposium. USENIX. ; 2015: 129–144.
14. Latora V, Marchiori M. A Measure of Centrality Based on Network Efficiency. *New Journal of Physics* 2007; 9(6): 188–188.
15. Gervais A, Karame GO, Wüst K, Glykantzis V, Ritzdorf H, Capkun S. On the Security and Performance of Proof of Work Blockchains. In: ACM SIGSAC Conference on Computer and Communications Security. ACM. ; 2016: 3–16.

16. Bitnodes. available at https://bitnodes.earn.com; .

17. NetworkX. available at https://networkx.github.io/documentation/stable/index.html; .

18. Blockchain.info. available at https://www.blockchain.com/ja/pools; .

19. Chaudhari SS, Biradar RC. Survey of Bandwidth Estimation Techniques in Communication Networks. *Wireless Personal Communications* 2015; 83(2): 1425–1476.

20. Aoki Y, Shudo K. Proximity Neighbor Selection in Blockchain Networks. In: IEEE International Conference on Blockchain. IEEE. ; 2019: 52-58.

21. Karame GO, Androulaki E, Capkun S. Double-Spending Fast Payments in Bitcoin. In: ACM Conference on Computer and Communications Security. ACM. ; 2012: 906–917.

22. Karame GO, Androulaki E, Roeschlin M, Gervais A, Capkun S. Misbehavior in Bitcoin: A Study of Double-Spending and Accountability. *ACM Transactions on Information and System Security* 2015; 18: 1-32.

23. Kroll JA, Davey IC, Felten EW. The Economics of Bitcoin Mining, or Bitcoin in the Presence of Adversaries. In: Workshop on the Economics of Information Security. Georgetown University. ; 2013: 1-21.

24. Eyal I, Sirer EG. Majority Is Not Enough: Bitcoin Mining Is Vulnerable. In: Christin N, Safavi-Naini R., eds. *Financial Cryptography and Data Security*Berlin Heidelberg: Springer. 2014 (pp. 436–454).

25. Eyal I, Sirer EG. Majority Is Not Enough: Bitcoin Mining Is Vulnerable. *Communications of The ACM* 2018; 61(7): 95–102.

26. Apostolaki M, Zohar A, Vanbever L. Hijacking Bitcoin: Routing Attacks on Cryptocurrencies. In: IEEE Symposium on Security and Privacy. IEEE. ; 2017: 375–392.

27. Walck M, Wang K, Kim HS. TendrilStaller: Block Delay Attack in Bitcoin. In: IEEE International Conference on Blockchain. IEEE. ; 2019: 1-9.

28. Nayak K, Kumar S, Miller A, Shi E. Stubborn Mining: Generalizing Selfish Mining and Combining with an Eclipse Attack. In: IEEE European Symposium on Security and Privacy. IEEE. ; 2016: 305–320.

29. RFC 4271 - A Border Gateway Protocol 4. available at https://tools.ietf.org/html/rfc4271; .

30. Sasabe M. Mathematical Epidemiological Analysis of Dynamics of Delay Attacks on Pull-Based Competitive Information Diffusion. *Computer Networks* 2020; 180: 1–9.

31. Brauer F, Driessche v. dP, Wu J., eds.*Mathematical Epidemiology* . 2008.

32. BIP130. available at https://github.com/bitcoin/bips/blob/master/bip-0130.mediawiki; .

33. BIP152. available at https://github.com/bitcoin/bips/blob/master/bip-0152.mediawiki; .

How to cite this article: M. Sasabe, M. Yamamoto, Y. Zhang, S. Kasahara (2021), Block diffusion delay attack and its countermeasures in a Bitcoin network, *Int J Network Mgmt.*, .