Dynamic Network Configuration: An Effective Defensive Protocol for Public Blockchain

Zhengwei Jiang[1(✉)], Chenyang Lv[2(✉)], Bo Zhang[3(✉)], Chao Zhang[4(✉)], Wei Lu[4(✉)], and Shouling Ji[2(✉)]

[1] State Grid Zhejiang Electric Power CO., LTD., Hangzhou, China
jiang_ng_zhengwei@zj.sgcc.com.cn

[2] College of Computer Science and Technology, Zhejiang University, Hangzhou, China
{puppet,sji}@zju.edu.cn

[3] NARI Group Corporation, Beijing, China
zhangbo7@sgepri.sgcc.com.cn

[4] State Grid Hangzhou Power Supply Company, Hangzhou, China
13757118405@139.com, 48618828@qq.com

Abstract. To earning unfair profits, adversaries can attack the legitimate nodes in the Bitcoin network with *selfish mining*, the *eclipse attack* and the *information delaying attack*. In this paper, we study the patterns of the above attacks and then present a scheme to enhance the security of the Bitcoin network. First, we propose a new structure for the Bitcoin network called double-layer dynamic network, which improves the defense capability of Bitcoin against several attacks. Second, we design a new structure for the blocks in the Bitcoin network, which provides a way to store the IP addresses in the blocks. Third, we present a novel network protocol named dynamic network configuration for public blockchain. Our protocol pushes the updating of IP addresses in the blocks and changes the construction of the network periodically. From theoretical analysis and simulated evaluation, we find that under our protocol, the Bitcoin network can defend against the *selfish mining*, the *eclipse attack* and the *information delaying attack* effectively.

Keywords: Bitcoin · Defensive protocol · P2P network

1 Introduction

Bitcoin [1] has received much more attention and trust than any other digital cryptocurrency recently. One of the key reasons for the success of Bitcoin is that it uses a distributed database called blockchain rather than a central bank. Most nodes in the Bitcoin network work on the longest blockchain. The blockchain implements a hash-based Proof of Work (PoW) mechanism and uses blocks to store transactions. By using the PoW mechanism, the blockchain ensures that the recorded transactions are unmodifiable. Therefore, Bitcoin provides more reliable currencies than other digital cryptocurrencies.

The Bitcoin network has an open network environment. It allows nodes to join and leave the network freely. While the open network environment also brings several security risks. Adversaries can join the network and gain unfair profits by attacking legitimate nodes using typical attacks such as *double spending* and *selfish mining*. Many researches explain the attacks on the Bitcoin network [2,3,5,9,12–14].

To defend against the above attacks, a few research has been conducted [4,15]. However, to our knowledge, there is no research that can defend against multiple intractable attacks for the Bitcoin network.

Therefore, in this paper, we introduce a novel defensive protocol to enhance the security of the Bitcoin network and defend against *selfish mining*, the *eclipse attack*, and the *information delaying attack*. The contributions of our work in this paper can be summarized as follows:

- We analyze the vulnerabilities of the Bitcoin network by discussing the implementations of *selfish mining*, *eclipse attack* and *information delaying attack*. Based on the analysis, we propose a novel protocol named Dynamic Network Configuration (DNC) and a new structure of the overlay network called Double-layer Dynamic Network (DDN) to defend against above mentioned attacks.
- We analyze the defensive capability of the DNC protocol both theoretically and experimentally. Our protocol is very effective against *selfish mining*, *eclipse attack* and *information delaying attack* in the Bitcoin network.

2 Background

In this section, we introduce the basic concepts of the Bitcoin network to facilitate our discussion. A complete explanation of Bitcoin can be found in [6].

Nodes. The nodes in the Bitcoin network are identified by IP addresses and can trade with other nodes. Each node has at most 8 outgoing TCP connections and 117 incoming TCP connections. We model the Bitcoin network by a graph $G = (V, E)$, where $V = \{A_1, A_2, ...\}$ and $E = \{e_{ij} | A_i, A_j \in V\}$ characterize the set of nodes and the set of connections among the nodes respectively. Let $outaddr(A_k)$ be the set of peers that are connected by the outgoing connections of node A_k, and $inaddr(A_k)$ be the set of peers that connect to the node A_k by their outgoing connections. $tried(A_k)$ denotes the set of IP addresses in the *tried table* of node A_k. Let $|V| = n$, i.e., the number of nodes in Bitcoin is n.

Inv Messages. Nodes in the Bitcoin network use the *inv* messages to reduce the cost of spreading messages. The *inv* message only contains the type and the hash of the entire message. Hence, the size of an *inv* message is much smaller than that of a complete message. The *inv* messages are saved in the First In First Out (FIFO) buffer of each node.

The New Table. Each node maintains a *new table* which contains the IP addresses of a node that it has not yet established successful connections with.

The Tried Table. Each node in Bitcoin maintains a *tried table* which contains 4,096 IP addresses. When a node has successfully established outgoing or incoming connections with peers, it stores the IP addresses in its *tried table*. Each node keeps the timestamps of the latest successful connections to the peers.

Transactions. Bitcoin uses the transaction-to-transaction payments to implement the trades among the nodes. Transactions are the messages to transfer the ownerships of coins between the nodes in Bitcoin.

Blocks. The construction of a block has two parts: the transaction data part and the block header. Transactions are stored as the leaf nodes of the merkle trees in the transaction data part of blocks. After hash computing, the merkle roots will be stored in the block headers. The block header also stores the hash of the previous block's header. Due to the high computational cost of hash computing, it is usually assumed that the messages on the blocks are inalterable.

Miners. Miners are a kind of nodes that have computing power. Miners collect transactions and mine blocks to make the blockchain get longer. A miner mines a block means that the miner calculates the hash of the previous block header that satisfies the requirements and obtains a new block. The miner spreads the new block in the Bitcoin network and will be rewarded with coins.

Blockchain. The Bitcoin network uses a decentralized database called blockchain, where transactions are stored in the blocks of the blockchain. Since the block header stores the hash of the previous block's header, blocks are logically organized as a chain. All the nodes in the Bitcoin network reach a consensus on the longest blockchain by downloading from the first block to the latest block. Then, they keep working on the longest blockchain. The height of a block is the number of blocks between this block and the first Bitcoin block in the blockchain.

3 Vulnerability Analysis

3.1 Selfish Mining

Eyal and Sirer showed that a mining pool controlling more than 33% of the computing power in Bitcoin can increase the mining advantages by withholding its mined blocks [5]. The pool releases the hidden blocks to get rewards when new blocks are found by other miners.

From this perspective, we consider the nodes of adversaries with the following strategy: nodes of adversaries are randomly distributed in the Bitcoin network. Their outgoing connections connect to legitimate IP addresses randomly. Meanwhile, the nodes of adversaries mine blocks cooperatively. When a node of adversaries mines a new block, the node transmits the block to other nodes of adversaries and they withhold the block. When legitimate miners mine another block and spread it in the Bitcoin network, sooner or later one or more nodes of adversaries will receive the block. Then, these nodes inform the rest of adversaries' nodes and all they spread the withheld block in order to make other nodes work on adversaries' hidden block. We explore how many nodes will receive the hidden block first and work on the adversaries' block eventually through this strategy via simulation. The result is shown in Fig. 1.

From Fig. 1, even if adversaries control 10% of all the nodes in the Bitcoin network, there are nearly 60% nodes receiving the hidden block of adversaries first and working on the hidden block under this strategy.

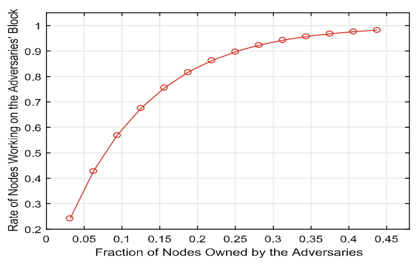

Fig. 1. The number of nodes eventually working on the adversaries' hidden block versus the number of adversaries' nodes.

The result indicates that if adversaries have enough nodes, most nodes will work on their hidden block. The hidden block will be the member of the longest blockchain with much higher probability than the blocks mined by legitimate miners. Thus, adversaries can accumulate mining advantages by always withholding the blocks and releasing one of the hidden blocks when legitimate nodes publish a new block. We come to a conclusion that adversaries can do *selfish mining* without the risk of losing the rewards under this strategy.

3.2 Eclipse Attack

Heilman et al. showed that by monopolizing the outgoing connections and incoming connections of victims and controlling the messages that victims receive and send, adversaries can utilize the computing power of victims for their own use [2]. We call this kind of attack the *eclipse attack*.

There are two main vulnerabilities utilized by *eclipse attack*. (1) Adversaries can attack the *tried table* and *new table* to improve the success rate of monopolizing the victims' outgoing connections. As shown in [2], when the resources are enough, adversaries can monopolize the outgoing connections of victims with a quite high probability. (2) It is difficult for victims to realize that they have already been attacked by adversaries. This is because victims cannot be conscious of that their input and output messages are filtered by adversaries.

3.3 Information Delaying Attack

Gervais et al. showed that by sending victims *inv* messages and ignoring the requests for the entire message, adversaries can delay the victims for receiving specific messages for nearly 20 min [3]. Because of such latency, adversaries can implement double-spending attack more easily and improve the reward of *selfish mining*. We call such attack the *information delaying attack*.

Specifically, *information delaying attack* utilizes the vulnerability that each node in the Bitcoin network has to request the sender of the *inv* message for the entire message in sequence. However, the Bitcoin network does not punish the senders who do not respond the requests. We deduce the reasons as follows: first, there are many reasons that receivers do not receive the entire message. For example, the senders do not receive the requests of the receivers, the entire message is lost during the transmission and so on. Second, adversaries may utilize the punishment mechanism if there is one to harm the profits of legitimate senders. Thus, it is improper to punish the overtime senders, we consider modifying the advertisement-based request management system to defend against the *information delaying attack*.

Based on the aforementioned discussion, we design DNC to patch the vulnerabilities and improve the construction of the Bitcoin network.

4 Dynamic Network Configuration

In this section, we give the details of our DNC protocol. The main idea of DNC is as follows: Different from the traditional design that nodes maintain the long term connections, DNC is designed to build a dynamic structure of a communication network for Bitcoin. Nodes periodically reconnect to different peers on their own initiative. We introduce the primary modules of DNC below:

(1) **Preprocessing Module.** To implement DNC, first, leveraging the Bitcoin network, we logically construct DDN, which contains High Layer Dynamic Network (HLDN) and a Low Layer Dynamic Network (LLDN). Second, we design a new block structure to store the IP addresses for the nodes in HLDN.
(2) **Outgoing Connection Maintenance Module.** We redesign the rules for determining outgoing connections to ensure the defensive capability of the DNC protocol and the structure of DDN. Each node in the Bitcoin network has to follow the rules when the replacement period starts, when a node connects to peers and when a node disconnects from peers.
(3) **Substitutive Peer Selection Module.** The nodes in HLDN are updated after a fixed replacement period. By using the substitutive peer selection module, each node in HLDN will choose a substitutive peer and announce its IP address in the Bitcoin network periodically.

To start the DNC protocol, the community of Bitcoin run the preprocessing module. Then, each node in Bitcoin runs the outgoing connection maintenance module and the substitutive peer selection module to maintain DNC.

4.1 Preprocessing Module

In DNC, we first construct a DDN for the Bitcoin network to transmit information. To implement DDN, we first select a small part of nodes from Bitcoin network to construct HLDN. The remaining nodes form the sub-network called LLDN. The illustration of DDN is shown in Fig. 2.

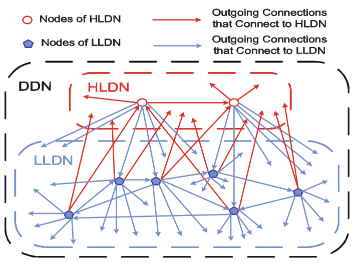
Nodes of HLDN
Nodes of LLDN
Outgoing Connections that Connect to HLDN
Outgoing Connections that Connect to LLDN
DDN
HLDN
LLDN

Fig. 2. The illustration of DDN.

Let $V_H = \{A_i, A_j, A_k...\}$ and $V_L = \{A_m, A_n, A_p, ...\}$ characterize the set of the nodes in HLDN and in LLDN respectively. Let $|V_H| = n_H$ and $|V_L| = n_L$.

HLDN has the following properties. First, the number of nodes in HLDN is small. Second, the IP addresses of the nodes in HLDN will be stored in the blocks with a new structure. Therefore, each node can obtain the IP addresses of all the nodes in HLDN from the latest block. Third, each node in DDN is only allowed to use two outgoing connections to connect to HLDN. Fourth, under the DNC protocol, the nodes in HLDN can transmit the entire messages directly without sending *inv* messages first and waiting for the requests from the receivers. Therefore, the nodes in HLDN will expedite the spread speed of messages. Once a node receives an unsolicited message from an incoming connection, it checks whether the IP address that sends the message is in the latest block. If yes, the node receives the message and checks the content; it abandons the message otherwise.

The nature of DDN is indicated by the mechanism that each node in HLDN will select a substitutive peer after a period of replacement (Sect. 4.3).

After constructing DDN, the Bitcoin network also has to implement our new construction of blocks. The IP addresses of the nodes in HLDN will form a merkle tree in the address part of the new blocks and will be stored in the leaf nodes of the merkle tree. Miners obtain a merkle root by hash computing and store the root in the block header. Thus, the IP addresses of the nodes in HLDN will influence the hash computation of the block header. In addition, miners cannot mine the block without the IP addresses of the nodes in HLDN. In particular, The new block structure is shown in Fig. 3.

Note that the new block structure can be implemented without modifying the block structure of the existing blockchain. Since existing blocks are not influenced by the later blocks, the blocks with the new structure can still store the hash

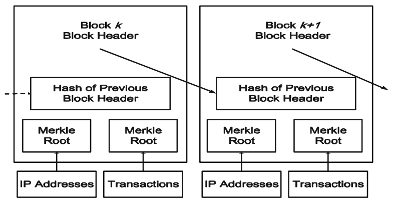

Fig. 3. The new block structure.

headers of the existing blocks. Therefore, the Bitcoin network can implement the new block structure whenever the need arises.

Because of the new block structure, each node that receives a new block can get the IP addresses of the nodes in HLDN from the table, which we call the *global table*. Further, since the nodes in HLDN only select the connected IP addresses to store in blocks and the IP addresses in the *global table* are replaced periodically, the *global table* in the latest block provides more reliable IP addresses than the *tried table* and *new table* of each node.

4.2 Outgoing Connection Maintenance Module

Since all the nodes are logically divided into two parts in DDN: the nodes in HLDN and the nodes in LLDN. We design the rules of determining the outgoing connections for each node in DDN to ensure the performance and the defensive capability of DNC.

There are two rules of constructing outgoing connections for each node in DDN: the first is that two outgoing connections have to connect to the IP addresses in the latest *global table*, the second is that the remaining outgoing connections have to connect to the IP addresses which are not in the latest *global table*. In other words, we limit the number of the outgoing connections for each node to connect to HLDN is two. Under the DNC protocol, each node has to maintain the rules when the node receives blocks containing the updated *global table*, when the node disconnects from other peers, and when the node connects to others.

From the above rules, all the outgoing connections are required to connect to IP addresses. This makes DDN more compact, makes messages spread more quickly and increases the difficulty of attacks on transmitting messages. Now, we describe the details of the outgoing connection maintenance module. Let T_G be the set of IP addresses in the *global table* of the latest block. Specifically,

(1) For a node A_k in DDN, if $|outaddr(A_k) \cap T_G|$ is greater than 2, A_k will randomly select an IP address from $(outaddr(A_k) \cap T_G)$ and disconnect from it until $|outaddr(A_k) \cap T_G|$ is equal to 2.

(2) If $|outaddr(A_k) \cap T_G|$ is less than 2, A_k should connect to a random IP address from $(T_G - outaddr(A_k))$ by an unoccupied outgoing connection until $|outaddr(A_k) \cap T_G|$ is equal to 2. If A_k does not have any unoccupied outgoing connection, A_k will randomly select an IP address from $(outaddr(A_k) - T_G)$ and disconnect from that IP address to get an unoccupied outgoing connection.

(3) After the above steps, A_k has satisfied the requirement that $|outaddr(A_k) \cap T_G|$ is equal to two. If A_k still has unoccupied outgoing connections, A_k will randomly select IP addresses from $(tried - T_G - outaddr(A_k))$ with a bias towards addresses with fresher timestamps. Then, A_k will try to connect to them until there is no unoccupied outgoing connections.

4.3 Substitutive Peer Selection Module

Each node in HLDN will select a substitutive peer after a fixed replacement period, which is implemented by the substitutive peer selection module. Note that, if a node A_n in LLDN runs the module and announces the substitutive IP address, all the nodes in the Bitcoin network can check the *global table* to realize that A_n is not the member of HLDN and ignore its replacement message. We give the details of the module below.

The replacement period is counted by the number of blocks. For example, if the period is counted by 10 blocks, every IP address of the nodes in HLDN will be stored in the *global table* for 10 blocks. Then, the IP address in the *global table* will be replaced by the substitutive IP address selected by itself. Each node in LLDN has approximately the same probability to be selected as the member of HLDN (the node that has more outgoing connections can have a higher probability, since more nodes may select it as the substitutive IP address). Specifically,

(1) When the replacement period starts, each node in HLDN will obtain an unoccupied outgoing connection first. For a node A_k in HLDN, If A_k does not have any unoccupied outgoing connection, A_k will randomly select an occupied outgoing connection connecting to LLDN and disconnect it.

(2) Then each node in HLDN will try to randomly select a substitutive IP address from its incoming connections. The substitutive IP address I_i of A_k should satisfy the following requirements: (i) $I_i \notin T_G$; (ii) $I_i \notin outaddr(k)$; (iii) I_i has not been chosen by other nodes in HLDN. A_k uses an unoccupied outgoing connection to connect to I_i. If they establish the connection successfully, A_k will spread the message in the whole network to announce that I_i will be the substitutive IP address of A_k. Then, I_i will be stored in the *global table* of the subsequent blocks.

(3) If the current incoming connections of A_k do not satisfy the requirements or A_k fails too many times to connect to the IP address from $inaddr(A_k)$, A_k will randomly select an appropriate IP address from $tried(A_k)$. After establishing the connection successfully, A_k announces the substitutive IP address.

In the substitutive peer selection module, DNC requires that each node in HLDN has to connect to the substitutive peer with the outgoing connection. There are two reasons for this design. First, two nodes can reach a consensus on the longest blockchain by establishing a connection and exchanging the version messages, which prevents that the substitutive peer works on a shorter fork. Second, the DNC protocol can preferentially guarantees that the nodes in HLDN will have the incoming connections with a higher probability. Since a node in HLDN can directly send an entire message to its peers, it will transmit messages faster than the nodes in LLDN. Without the incoming connections, the nodes in HLDN cannot receive messages or spread the messages from the peers, which wastes the transmittability of HLDN.

We compare the difference between the current Bitcoin network and that under the DNC protocol. The Bitcoin network at present requires nodes to establish long-range connections. Therefore, the construction of the whole network is static. Our protocol requires nodes to reconnect to different peers periodically by the outgoing connection maintenance module and the substitutive peer selection module. Therefore, the network's construction is dynamic. Since the nodes in DDN connect to the peers randomly and frequently, the distribution of the nodes in DDN will be more balance than the present distribution. In this way, the DNC protocol can improve the security of the Bitcoin network.

5 Defensive Capability Analysis

5.1 Selfish Mining

We first analyze the defensive capability of DNC against *selfish mining*.

Theorem 1. *Legitimate nodes will detect the number of selfish miners' blocks periodically under the DNC protocol.*

Proof. Because of the replacement of nodes in HLDN and the maintenance of outgoing connections, nodes will disconnect from peers and connect to other peers periodically and randomly. If *selfish miners* are connected to other legitimate nodes or are connected by others, they will exchange the version messages containing the length of their blockchains to establish the connection. Then the hidden blocks of *selfish miners* will be exposed to the legitimate nodes, followed by, legitimate nodes can detect the number of selfish miners' blocks periodically.

Because of the same reasons mentioned above, the legitimate nodes whose outgoing connections are connected to *selfish miners* will disconnect from *selfish miners* gradually. If *selfish miners* refuse to establish incoming connections, the number of their incoming connections will be decreased as time goes on. Eventually, *selfish miners* will have no incoming connections.

Then, we consider a worse case that adversaries can forge the version messages to conceal the length of the hidden blockchain and establish connections with legitimate nodes successfully. There is another countermeasure to defend against *selfish mining* as shown in the following theorem.

Theorem 2. *Under the DNC protocol, the mining advantage of selfish miners will be reset when the replacement period starts.*

Proof. When the replacement period starts, the nodes of HLDN will randomly choose the substitutive nodes. All the miners have to wait for the replacement messages of the nodes in HLDN. Adversaries cannot forge the replacement messages or forecast the substitutive IP addresses of all the nodes in HLDN. Without the correct IP addresses, *selfish miners* cannot construct the correct merkle tree, nor calculate the hash of the block header. Thus, *selfish miners* cannot selfishly mine the block containing the updated *global table.* When all the replacement messages are spread in the Bitcoin network, both legitimate miners and *selfish miners* start to mine the subsequent block at the same time. Therefore, the mining advantage of *selfish miners* accumulated before will be reset.

Since *selfish miners* cannot mine the subsequent block and it is unnecessary to wait for the legitimate miners to catch up. If *selfish miners* have hidden blocks and their hidden blockchain is coming up to the block which is going to update the *global table*, *selfish miners* will spread the hidden blocks to get rewards.

By adjusting the number of blocks that each fixed replacement period has, the DNC protocol limits the maximal length of the hidden blockchain. The optimal defensive capability against *selfish mining* is to set the period counted by one block. In other words, the Bitcoin network will update the *global table* when a miner mines a block. In practice, the community of Bitcoin can set an appropriate number of the blocks that one replacement period has. Considering the computing power that adversaries may have and the probability of mining a block by adversaries, the periodic time can be set loosely.

5.2 Eclipse Attack

As we discussed in Sect. 3.2, there are two main vulnerabilities of the Bitcoin network that are utilized by the *eclipse attack.* In DNC, we provide novel ways to patch these two vulnerabilities.

For the first vulnerability, we explain how DNC lowers the success rate that adversaries can monopolize all the outgoing connections of victims.

For each node under the DNC protocol, two outgoing connections connect to HLDN and six outgoing connections connect to LLDN (it is the default setting that each node has eight outgoing connections). When a victim restarts, it will randomly select two IP addresses from the *global table* and randomly select six IP addresses from the *tried table* to connect.

Let H_a be the number of adversaries' nodes in HLDN, and H_n be the number of legitimate nodes in HLDN. Let $P_h(H_a, H_n)$ be the probability that both two outgoing connections connect to the adversaries' nodes in HLDN. Then, we have

$$P_h(H_a, H_n) = \left\{ \frac{C^2_{H_a}}{C^2_{H_a+H_n}} \right\}. \tag{1}$$

Let r be the number of IP addresses that have been rejected so far, and t be the difference between the timestamp of the IP address and the current time.

$p(r,t)$ denotes the function counting the probability that a node uses this IP address rather than reject it. Then, we have

$$p(r,t) = min\left\{1, \frac{1.2^r}{1+t}\right\}. \tag{2}$$

Let t_a be the difference between the timestamp of adversaries' IP addresses and the current time, and t_n be the difference between the timestamp of legitimate IP addresses and the current time. The adversaries' IP addresses occupy α fraction of all the IP addresses in the *tried table*. $F(i,\alpha,t_a,t_n)$ denotes the function counting the probability that the i^{th} node rejects an IP address. Then, we have

$$F(i,\alpha,t_a,t_n) = [1 - p(i-1,t_a)] \cdot \alpha + [1 - p(i-1,t_n)] \cdot (1-\alpha). \tag{3}$$

Let $P_l(k,\alpha,t_a,t_n)$ be the probability that a node rejects an IP address $(k-1)^{th}$ times and at the k^{th} time the node connects to the IP address of adversaries in LLDN. We have

$$P_l(k,\alpha,t_a,t_n) = \alpha \cdot p(k-1,t_a) \cdot \prod_{i=1}^{k-1} \cdot F(i,\alpha,t_a,t_n). \tag{4}$$

$P_m(H_a,H_n,k,\alpha,t_a,t_n)$ denotes the probability that adversaries monopolize all the outgoing connections of a victim. We have

$$P_m(H_a,H_n,k,\alpha,t_a,t_n) = P_h(H_a,H_n) \cdot \prod_{i=1}^{6} P_l(k_i,\alpha,t_a,t_n). \tag{5}$$

We can learn from $P_m(H_a,H_n,k,\alpha,t_a,t_n)$ that adversaries can improve $P_l(k,\alpha,t_a,t_n)$ with the *eclipse attack*. However, it is difficult for adversaries to improve $P_h(H_a,H_n)$. Let P_{Ha} be the proportion of the number of adversaries' nodes in HLDN to the number of all the nodes in HLDN. P_{Aa} denotes the proportion of the number of adversaries' nodes to the number of nodes in the network. Since each node in HLDN selects the substitutive IP address randomly, we conjecture that P_{Ha} should be nearly equal to P_{Aa}.

The growth rate of $P_h(H_a,H_n)$ is pretty low. Even if P_{Aa} is 40%, the expected value of $P_h(H_a,H_n)$ is only 16%. Considering the success rate of monopolizing six outgoing connections that connect to LLDN, the expected value of the final success rate P_m is lower than 16%. Therefore, our protocol significantly lowers the success rate of monopolizing a victim's all the outgoing connections.

We also do simulated experiments to verify the defensive capability of the DNC protocol against monopolizing all the eight outgoing connections by the *eclipse attack*. In the simulation, the number of nodes in the Bitcoin network is 12,288. Each node maintains its *tried table* that can store 4,096 IP addresses at most. We use different fractions of nodes owned by adversaries to attack legitimate nodes and test the success rate of monopolizing all the eight outgoing connections when the timestamps difference of adversaries' nodes and legitimate nodes is 2 h. In the simulation, *num* is the number of nodes owned by adversaries. Let $P_{AinTried}$ be the average proportion of victims' *tried tables* that occupied by adversaries. In addition, $P_{success}$ is the success rate that adversaries monopolize all the outgoing connections of victims.

The results are shown in Fig. 4 and Table 1, which confirm that DNC can significantly lower the success rate of monopolizing all the eight outgoing connections of a victim. Even if adversaries own 40.625% of all the nodes in the Bitcoin network, their success rate of monopolizing all the eight outgoing connections of a victim is less than 16.5%. We can further learn from Table 1 that although adversaries can still occupy the most items in the *tried table* of a victim, it is much harder for them to increase the number of their IP addresses in the *global table*.

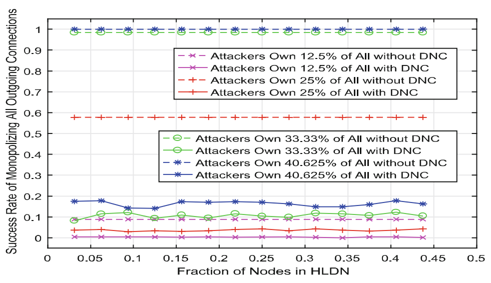

Fig. 4. The success rate of monopolizing all the eight outgoing connections of a victim by the *eclipse attack* with and without the DNC protocol.

Because each node in HLDN randomly selects the substitutive IP address, adversaries' nodes in HLDN should be only influenced by the total number of adversaries' nodes. We can learn from Table 1 that P_{Ha} is nearly equal to P_{Aa}. We can also find that P_{HtoAll} does not affect P_{Ha}. These results confirm our conjecture.

Table 1 also indicates that an effective way for adversaries to improve $P_h(H_a, H_n)$ is to add more nodes to the network, which requires more resources. However, if adversaries put lots of computing resources into the Bitcoin network, they can earn more coins even by following the rules of Bitcoin. If adversaries attack the Bitcoin network, Bitcoin may lose the trust of users. Then, coins may lose their value. As a result, adversaries may waste their computing resources. Thus, the more resources adversaries put into Bitcoin, the less likely they will attack the Bitcoin network.

The above analysis shows that DNC can significantly lower the success rate of monopolizing all the eight outgoing connections. Next, we consider the worst case that all the outgoing connections of a victim have connected to the IP addresses of adversaries. We have the following lemma and theorem to show that the DNC protocol can help victims detect the *eclipse attack*.

Table 1. Simulation results of the eclipse attack.

P_{Aa}	num	P_{HtoAll}	P_{Ha}	$P_{AinTried}$	$P_{success}$	P_{Aa}	num	P_{HtoAll}	P_{Ha}	$P_{AinTried}$	$P_{success}$
12.5%	1,536	0		37.49%	8.82%	25%	3,072	0		74.99%	57.75%
12.5%	1,536	4/32	12.53%	38.18%	0.36%	25%	3,072	4/32	25.01%	75.85%	3.37%
12.5%	1,536	8/32	12.44%	38.65%	0.36%	25%	3,072	8/32	24.93%	76.04%	4.32%
12.5%	1,536	12/32	12.99%	40.47%	0.36%	25%	3,072	12/32	25.49%	77.37%	3.25%
33.3%	4,096	0		99.00%	98.39%	40.625%	4,992	0		100%	100%
33.3%	4,096	4/32	33.24%	99.59%	9.38%	40.625%	4,992	4/32	40.50%	99.94%	14.06%
33.3%	4,096	8/32	33.16%	99.71%	10.33%	40.625%	4,992	8/32	40.46%	99.95%	15.99%
33.3%	4,096	12/32	33.81%	99.77%	10.70%	40.625%	4,992	12/32	41.07%	99.97%	15.62%

Lemma 1. *If an adversary monopolizes all the incoming connections of a victim, the adversary cannot transmit blocks to the victim after the global table updates.*

Proof. If a victim receives a block containing the updated *global table*, in order to follow the rules of outgoing connections, the victim may disconnect from the IP addresses of the adversary and randomly select IP addresses from the *global table* or *tried table* to reconnect. Thus, it is possible that this victim connects to legitimate nodes. As a conclusion, adversaries cannot transmit blocks to victims.

Lemma 1 implies that an adversary cannot continuously utilize the computing power of a victim to selfishly mine or implement other attacks.

Theorem 3. *An adversary cannot monopolize the outgoing connections of a victim for a long time under the DNC protocol.*

Proof. Consider whether an adversary can monopolize all the incoming connections of a victim:

(i) If not, the victim can receive blocks and replacement messages from the Bitcoin network under the DNC protocol. Then, there is a certain probability that the victim's outgoing connections disconnect from the IP addresses of the adversary and connect to legitimate nodes.
(ii) If yes, according to Lemma 1, an adversary cannot transmit blocks to a victim. When the adversary do not transmit blocks, it is easy for the victim to figure out the abnormal condition. Since the victim cannot increase the incoming connections of different IP addresses on its own, it will change the IP addresses of outgoing connections. The victim can obtain IP addresses from the *global table* in the previous blocks or ask *DNS seeds* for IP addresses and connect to them.

We can learn from Theorem 3 that there is a dilemmatic circumstance when an adversary monopolize all the incoming connections of a victim. No matter the adversary sends the block with the updated *global table* or not, the victim will try to reconnect to different IP addresses with the outgoing connections. In other words, the victim can recover from the condition that all its outgoing connections are monopolized by the adversary under the DNC protocol.

5.3 Information Delaying Attack

DNC stipulates that nodes of HLDN can spread an entire message without spreading *inv* message first. Only if there is no outgoing connections of the legitimate nodes in HLDN connecting to victims, can adversaries delay the delivery of blocks and transactions successfully. Therefore, DNC lowers the success rate of the *information delaying attack*. Let $P_{dl}(H_n, n_L)$ be the success possibility of the *information delaying attack* to nodes in LLDN. Then, we have

$$P_{dl}(H_n, n_L) = \left\{ \frac{C^6_{n_L - 1}}{C^6_{n_L}} \right\}^{H_n}. \tag{6}$$

$P_{dh}(H_n, H_a)$ denotes the success possibility of the *information delaying attack* to the nodes in HLDN. Then, we have

$$P_{dh}(H_n, H_a) = \left\{ \frac{C^2_{H_n + H_a - 1}}{C^2_{H_n + H_a}} \right\}^{H_n}. \tag{7}$$

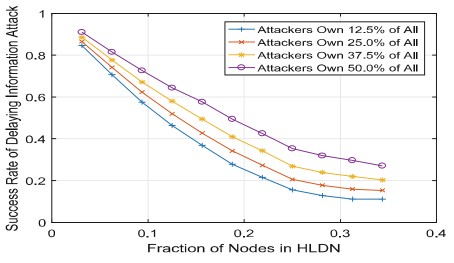

Fig. 5. The success rate of the *information delaying attack* is influenced by P_{HtoAll} and the number of nodes which are owned by adversaries.

We perform a simulation to evaluate the influence of DNC on the success rate of *information delaying attack*. The result is shown in Fig. 5, from which we can learn the following observation: if there are more nodes in HLDN, the success rate of the *information delaying attack* decreases. This is because there are more legitimate nodes in HLDN to deliver messages without sending *inv* messages. We also figure out that if adversaries own more nodes, P_{Ha} will increase. Then, there will be less legitimate nodes in HLDN.

6 Related Work

A number of works analyzed the security of Bitcoin [7,8,10,11]. Eyal and Sirer showed that if a mining pool withholds blocks on purpose, it can earn rewards in exceed of it is supposed to get [5]. These studies indicate that Bitcoin's countermeasures are not very effective to defend against *selfish mining*.

Karame et al. indicated that the measures from Bitcoin developers are not always effective in defending against double-spending [9]. There are some works [2,12–14] showed that *eclipse attack* can isolate the victims from other peers in a peer-to-peer network. Gervais et al. showed that adversaries can delay specific messages delivery to victims [3]. These demonstrate the importance of timely transmitting information in Bitcoin, which is a motivation of our research.

Ruffing et al. introduced a cryptographic primitive for preventing double-spending [4]. Schrijvers et al. introduced a game-theoretic model for reward functions of mining [16]. Bag et al. presented a new scheme for the PoW mechanism [15]. Different from the above researches, our study focused on improving the structure of the Bitcoin network. Our protocol is designed to enhance the security of transmission and to effectively defend against multiple attacks.

7 Conclusion

In this paper, we introduced a novel defensive protocol called Dynamic Network Configuration (DNC) to defend against *selfish mining*, the *eclipse attack* and the *information delaying attack* for the Bitcoin network. At first, we analyzed the vulnerabilities of the present Bitcoin network utilized by the aforementioned attacks. Based on our analysis, we presented the DNC protocol and then analyzed its defensive capability. From theoretical analysis and simulated experiments, we made the conclusions that the DNC protocol is effective against *selfish miners*, *eclipse attack* and *information delaying attack*.

Acknowledgement. This work was partly supported by NSFC under No. 61772466, the Zhejiang Provincial Natural Science Foundation for Distinguished Young Scholars under No. R19F020013, the Provincial Key Research and Development Program of Zhejiang, China under No. 2017C01055, the Fundamental Research Funds for the Central Universities, and the Alibaba-ZJU Joint Research Institute of Frontier Technologies. Technology Project of State Grid Zhejiang Electric Power co. LTD under NO. 5211HZ17000J.

References

1. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008)
2. Heilman, E., Kendler, A., Zohar, A., et al.: Eclipse attacks on bitcoin's peer-to-peer network. In: USENIX, pp. 129–144 (2015)
3. Gervais, A., Ritzdorf, H., Karame, G.O., et al.: Tampering with the delivery of blocks and transactions in bitcoin. In: CCS, pp. 692–705 (2015)
4. Ruffing, T., Kate, A., Schröder, D.: Liar, liar, coins on fire!: penalizing equivocation by loss of bitcoins. In: CCS, pp. 219–230 (2015)
5. Eyal, I., Sirer, E.G.: Majority is not enough: bitcoin mining is vulnerable. In: Financial Cryptography, pp. 436–454 (2014)
6. Bitcoin Developer Guide. https://bitcoin.org/en/developer-guide
7. Decker, C., Wattenhofer, R.: Information propagation in the bitcoin network (2013)
8. Bonneau, J., Miller, A., Clark, J., et al.: Sok: research perspectives and challenges for bitcoin and cryptocurrencies. In: S&P, pp. 104–121 (2015)

9. Karame, G.O., Androulaki, E., Capkun, S.: Double-spending fast payments in bitcoin. In: CCS, pp. 906–917 (2012)
10. Courtois, N.T., Bahack, L.: On subversive miner strategies and block withholding attack in bitcoin digital currency. arXiv preprint arXiv:1402.1718 (2014)
11. Gervais, A., Karame, G., Capkun, S., et al.: Is bitcoin a decentralized currency? S&P **12**(3), 54–60 (2014)
12. Singh, A., Ngan, T.W., Druschel, P., et al.: Eclipse attacks on overlay networks: threats and defenses. In: INFOCOM, pp. 1–12 (2006)
13. Sit, E., Morris, R.: Security considerations for peer-to-peer distributed hash tables. In: IPTPS, vol. 2429, pp. 261–269 (2002)
14. Castro, M., Druschel, P., Ganesh, A., et al.: Secure routing for structured peer-to-peer overlay networks. OSDI **36**(SI), 299–314 (2002)
15. Bag, S., Ruj, S., Sakurai, K.: Bitcoin block withholding attack: analysis and mitigation. TIFS **12**(8), 1967–1978 (2017)
16. Schrijvers, O., Bonneau, J., Dan, B., et al.: Incentive compatibility of bitcoin mining pool reward functions. In: Financial Cryptography, pp. 477–498 (2016)