Computer Systems Science & Engineering
DOI:10.32604/csse.2021.014630
Article

Tech Science Press

Blockchain Consistency Check Protocol for Improved Reliability

Mohammed Alwabel and Youngmi Kwon*

Department of Radio and Information Communications Engineering, Chungnam National University, Daejeon, Korea
*Corresponding Author: Youngmi Kwon. Email: ymkwon@cnu.ac.kr
Received: 04 October 2020; Accepted: 11 November 2020

Abstract: Blockchain is a technology that provides security features that can be used for more than just cryptocurrencies. Blockchain achieves security by saving the information of one block in the next block. Changing the information of one block will require changes to all the next block in order for that change to take effect. Which makes it unfeasible for such an attack to happen. However, the structure of how blockchain works makes the last block always vulnerable for attacks, given that its information is not saved yet in any block. This allows malicious node to change the information of the last block and generate a new block and broadcast it to the network. Given that the nodes always follow the longer chain wins rule, the malicious node will win given that it has the longest chain in the network. This paper suggests a solution to this issue by making the nodes send consistency check messages before broadcasting a block. If the nodes manage to successfully verify that the node that generated a new block hasn't tampered with the blockchain than that block will be broadcasted. The results of the simulation show suggested protocol provided better security compared to the regular blockchain.

Keywords: Blockchain; security; forking; blockchain consistency check (BCC); protocol

1 Introduction

Blockchain technology was proposed in 2008 and implemented in 2009 by Satoshi Nakamoto from Japan, and since then the technology has been gaining more attention every year [1]. Blockchain is a distributed ledger that contains the history of all transactions made in the network. The ledger is kept by volunteer nodes. This is done by using a consensus mechanism, which means that the state of the shared ledger is updated by achieving an agreement using consensus algorithms [2]. Since its implementation started, blockchain technology has been received positively by many for providing a reliable security mechanism, allowing the ability to get rid of the middleman in transactions, and providing anonymity for users. The reliability of the system comes from the consensus algorithms. These algorithms run all over the network in every node. The algorithm guarantees that all nodes in the network have the same ledger and any updates added to that ledger are added to every node in a reliable way. Hence, achieving consensus across the network, and making it hard to tamper with the ledger. Bitcoin, for example, uses proof of work consensus algorithm. In this algorithm, the nodes try to find a targeted value in order to be

allowed to build the next block and be rewarded for its work to find the targeted value. Blocks are added to the blockchain in a timely fashion. These blocks contain transactions made by other nodes in the network and the hash of the previous block. This means that until the next block is added to the blockchain, the last block is it a risk of being tampered with. The remainder of this paper covers the mechanism of how blockchain works, how its mechanism can be exploited, and the proposed work to mend this issue.

2 Background and Related Works

2.1 Blockchain Operation

Blockchain is a technology that allows multiple nodes to update a shared ledger in a peer-to-peer fashion. This technology allows the participating nodes to make transactions with one another in a secure and confidential manner. The use of P2P helps in getting rid of any intermediaries by allowing the nodes to communicate directly with each other. Blockchain is considered a shared database with an append-only functionality that cannot be changed. Any new entry to the database in one node gets reflected across the network, which ensures that all nodes have the same entry. Making all the nodes in the network maintain the same database.

Blockchain was designed to work as a decentralized system. A decentralized system is a system that does not have a single authority, and all nodes in the system have equal authority. Decentralized systems have a lack-of-trust problem because there is not any governing party. And in the case of blockchain, there is no third party to ensure that the sender and the receiver of transactions are safe. However, blockchain inherently has trust integrated into its network [3].

2.2 Background

Forking is a very important aspect of blockchain. Forking means that the blockchain is branching into two or more branches, and each branch has its own set of blocks [4]. The branches continue growing until one of them gets longer than the other(s). The network will continue with the longer branch and drop all the other branches [5]. Fig. 1 shows an example of a fork.

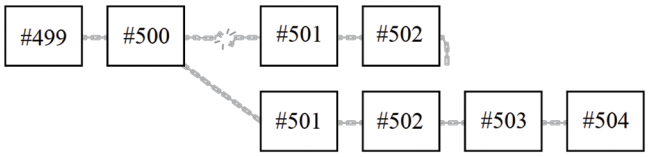

Figure 1: Example of forking in blockchain

Forks happen when two miners or more generate a block nearly at the same time; this is called an accidental fork [6]. However, malicious nodes can cause an accidental fork on purpose in order to manipulate the blockchain by either engaging in a process called selfish mining or trying to manipulate the last block. Selfish mining attack is an attack where a node holds the block that it had generated and wait until it generates more blocks, and then broadcast these blocks to the network at once. This will cause the network to fork and create a new branch. In case the branch that was caused by the selfish node was longer, that branch will be used by the network and the other honest as the main branch. And the short branch will be discarded. This is due to the longest chain rule used by the blockchain to resolve any forks that might occur [7].

A different method to fork the chain intentionally is manipulating the last block. This method can be done by leveraging on the fact that blockchain achieves security by saving the information of the previous block in the next block. Thus, it can be concluded that the last block is always vulnerable, given

that its information has not been stored in any block yet. This means that a malicious node can manipulate the last block (block #500 for example) and generate a new block (block #501) and broadcast it to the network. Now given that the malicious node has the longest chain, a chain with 501 blocks, that chain will win because of the longest chain rule [8]. More explanation on the last block manipulation in Chapter 3.

2.3 Related Works

The authors of Eyal et al. [9] suggested the most straightforward solution to amend the last block vulnerability, suggested that a node should randomly choose a fork to extend whenever it finds multiple forks of the same length. If all the nodes in the network randomly choose what fork to extend, the probability of extending the fork that was caused by the malicious node will decrease. The writers of Heilman [10] introduced the concept of *Freshness Preferred* (FP), which places the unforgeable timestamps in blocks and prefer blocks with recent timestamps. This approach uses Random Beacons to stop miners from using timestamps from the future. As selfish mining uses a strategic block withholding technique, the proposed strategy will decrease the incentives for selfish mining because withheld blocks will lose block races against newly minted or *fresh* blocks. A similar solution for selfish mining that requires no changes in the existing Bitcoin protocol was proposed in Zhang et al. [11]. The authors suggested a fork-resolving policy that selectively neglects blocks that were not published in time, and it appreciates blocks that include a pointer to competing blocks of their predecessors. Therefore, if the secretly mined block was not published in the network until a competing block was published, it will contribute to neither or both branches. Hence, it will not get benefits in winning the fork race. The writers of Zhang et al. [12] proposed another defense against selfish mining, in which miners need to publish intermediate blocks (or in-blocks). These blocks will only reward the miners who do a lot of work. Although miners who didn't do a lot of work can generate blocks, they won't be rewarded. When a fork happens, miners adopt the branch with the largest total amount of work, rather than the longest chain.

Unlike most of the aforementioned solutions against malicious forking, the writers of Solat et al. [13] proposed a timestamp-free prevention of block withholding attack called *ZeroBlock*. In ZeroBlock, if a selfish miner keeps a mined block private for more than a specified interval called *mat*, than later when this block is published on the network, it will be rejected by the honest miners. The key idea is that each consecutive block must be published in the network, and it should be received by honest miners within a predefined maximum acceptable time for receiving a new block (i.e., *mat* interval). In particular, an honest miner either receives or publishes the next block in the network within the *mat* interval. Otherwise, to prevent the block withholding, the miner itself generates a specific dummy block called *Zeroblock*. These signed dummy Zeroblocks will accompany the solved blocks to prove that the block is witnessed by the network and that a competing block is absent before miners can work on it. In Courtois et al. [14], Bahack suggested that the only viable option to countermeasure a block withholding attack launched within a pool is that the pool managers should involve *ONLY* miners who are personally known to them. Hence, they can be trusted. The pool manager should simply dissolve and close a pool as soon as the earning of the pool goes lower than expected from its computational effort.

While both selfish mining and manipulating the last block in the blockchain cause the blockchain to fork and stealing the efforts of honest nodes, it is important to note that these two attacks are different. Which in turn means that the solutions for them are different. Unlike all the previous works, this paper suggests a solution to prevent manipulating the last block in the blockchain by using a consistency check message between the nodes.

3 The Proposed Work

In blockchain, the hash of the previous block is saved in the next block. This means that the last block is always in danger of being manipulated with (since its information has not been hashed and stored in another block).

3.1 Original Blockchain Behavior

From the above rules regarding the blockchain behavior, it can be concluded that there is a possibility for a malicious node to alter the information of the last block, and generate another block to make its chain longer and broadcasts it to the network. In order to understand blockchain better, a closer look at the following equation written by Satoshi Nakamoto's paper about Bitcoin is needed [1].

p: Probability of an honest node finding the next block.

q: Probability of a malicious node finding the next block.

q_z: Probability of a malicious node catching up to the honest nodes. (z is explained in detail on the next page).

$$q_z = \begin{cases} 1, & if\ p \le q \\ \left(\frac{q}{p}\right)^z, & if\ p > q \end{cases} \tag{1}$$

This means that the malicious node will win and can manipulate the blocks if its probability is higher than or equal to the probability of the honest nodes to mine the next block [1]. (A better probability means that the malicious node has a better hashing power, or the number of the malicious nodes is larger than the honest nodes). However, if the hashing power or the number of the malicious nodes is less than the honest nodes, then the winning probability of the malicious node is smaller (i.e., the manipulation is not guaranteed to happen). The more blocks generated by the honest nodes (dictated by z), the less likely the malicious node will win. The following example explains more:

Assume the following, Node A is an honest node, and Node B is a malicious node. Both nodes have the same hashing power. Node B wants to manipulate the last block, block #500.

In this chain, the value of z is 0. Because right now both the honest and the malicious nodes have the same number of blocks after block #500, which is 0 as seen in Fig. 2.

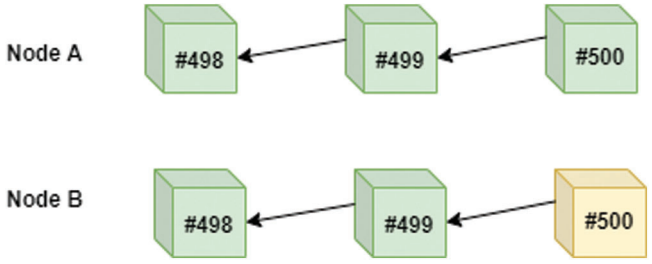

Figure 2: Blockchain example

Suppose now that block #501 was generated by Node A (an honest node) and broadcasted to Node B as seen in Fig. 3. Now the value of z = 1. This means that in order for Node B (the malicious node) to be able to manipulate block #500, it has to manipulate block #500 and generate two more blocks (block #501 and block #502) and broadcast them in order to successfully manipulate the blockchain. Suppose now that block #502 was generated by Node A (an honest node) and broadcasted to Node B as seen in Fig. 4.

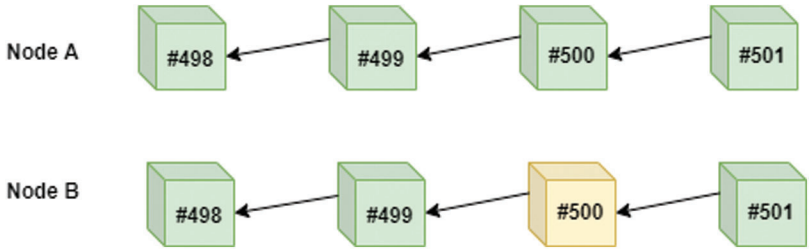

Figure 3: The value of z increments by 1

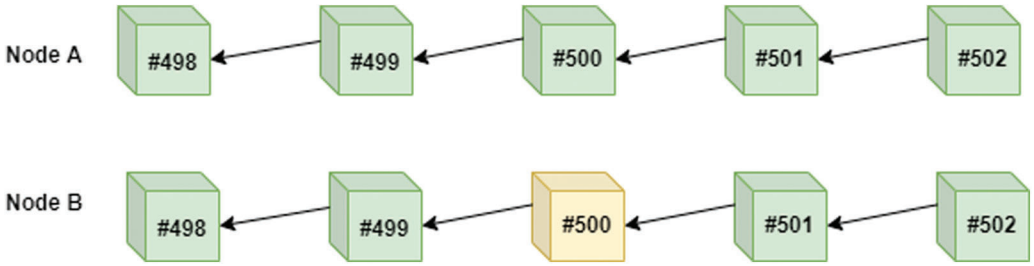

Figure 4: The value of z increments by 2

Now the value of z is 2. This means that in order for Node B (the malicious node) to be able to manipulate block #500, it has to manipulate block #500 and generate three more blocks, block #501, block #502, and block #503 in order to successfully manipulate the blockchain. This logic can be represented in the following equation:

$$z = \begin{cases} z + 1, & \text{if block generated by A} \\ z - 1, & \text{if block generated by B} \end{cases} \tag{2}$$

The bigger the value of z, the less likely B will be able to catch up and manipulate the chain. However, when the value of $z = 0$ (i.e., when the malicious node tries to attack the last block) the malicious node has a higher probability of manipulating the blockchain, and in theory, it can succeed. It can be concluded from this that the last block is always in danger because its hash is not saved yet in any block [15].

Once the malicious node manipulates the last block (block #500) and manages to generate the next block (block #501) the value of z becomes −1. Fig. 5 shows the blockchains in Nodes A and B after Node B broadcasted its new block (#501).

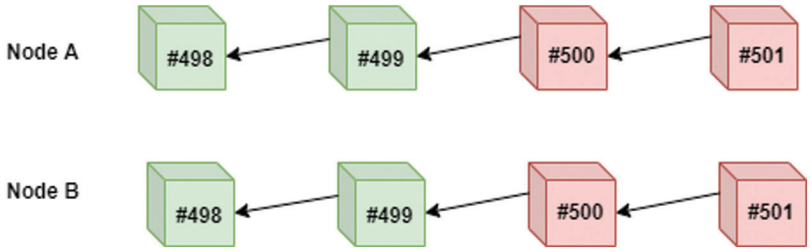

Figure 5: The value of z decrements by 1

3.2 Last Block Consistency Check Protocol

The way of how blockchain works currently does not give a chance for the honest nodes to defend themselves against this attack. Because as mentioned previously, the honest nodes will always trust the longest chain.

The malicious node managed to win in the previous example because the nodes in the network do not question the block and immediately accept it. There is not any form of inspection to ensure that the block hasn't been manipulated. As long as the block was generated by a node that has the longest chain, the other nodes will immediately trust the new block. Therefore, in order to amend this issue, this paper suggests the use of a blockchain consistency check (BCC) protocol. BCC protocol exchanges message between a node and its neighbors before a new block is broadcasted throughout the network. Unlike the regular blockchain where a block is broadcasted once it is generated, in the proposed method, any node that generates a new block as seen in Fig. 6 has to send a consistency check message to its neighbors first in order for them to decide whether the new block should be broadcasted or not.

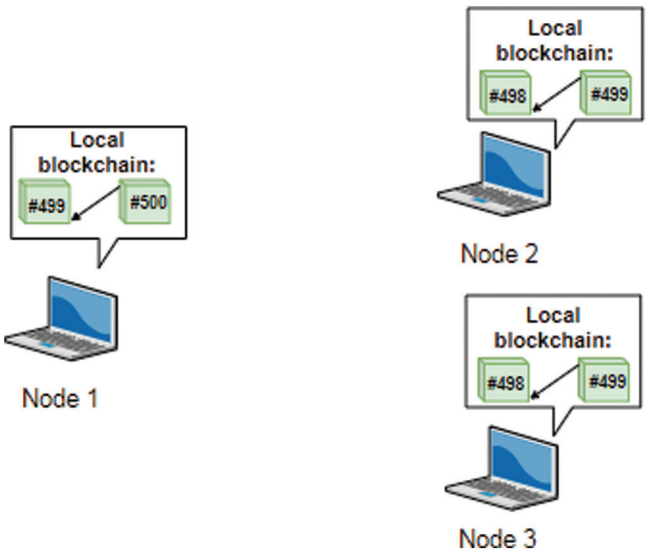

Figure 6: BCC protocol Step 1: Node 1 generated a new block (#500)

The BCC message consists of the two Merkle-Tree hashes of the last two blocks. Which are the Merkle-Tree hash of the second to last block (e.g., block #499), and the Merkle-Tree of the last block that was newly generated (e.g., block #500) as seen in Fig. 7.

Once the message arrives at the neighboring nodes, every node will compare the hash of the second to last block (block #499) in the message to its own local hash as seen in Fig. 8.

Right after that, the node that generated the new block will send the new block (block #500) to its neighbors as seen in Fig. 9.

Once the block arrives at the neighboring nodes, every node will compare the Merkle-Tree hash of the received block (block #500) to the Merkle-Tree hash of block #500 in the consistency check message as seen in Fig. 10.

In case the hash of block #500 was the same after inspecting it, the neighboring nodes (Node 2 and Node 3) will accept the block, and broadcast it. Otherwise, in case a mismatch was detected by the neighboring nodes, the neighboring nodes will not broadcast the new block preventing the malicious node from winning over the network.

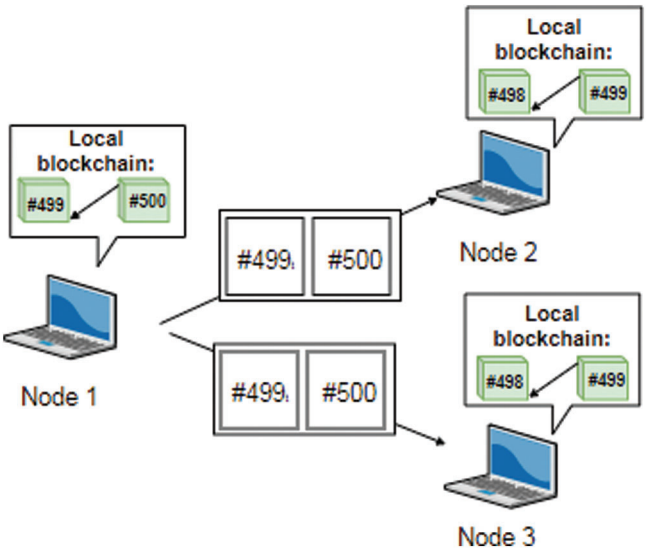

Figure 7: BCC protocol Step 2: Node 1 sends the BCC message to its neighbors

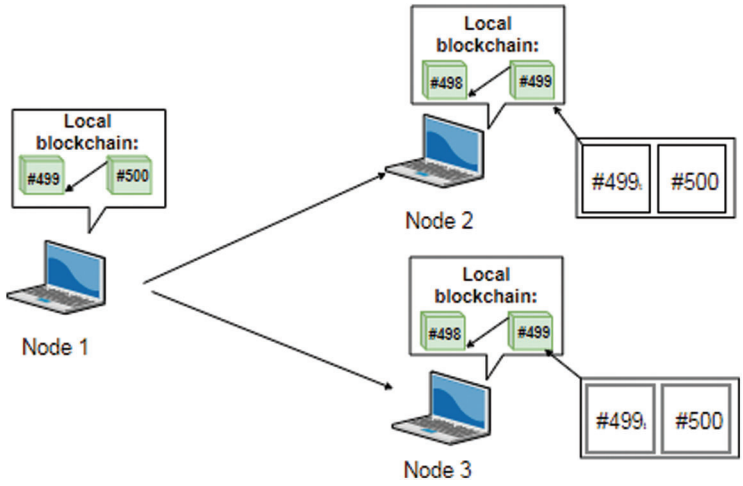

Figure 8: BCC protocol Step 3: The neighboring nodes compare their local hashes to hash included in the CCM

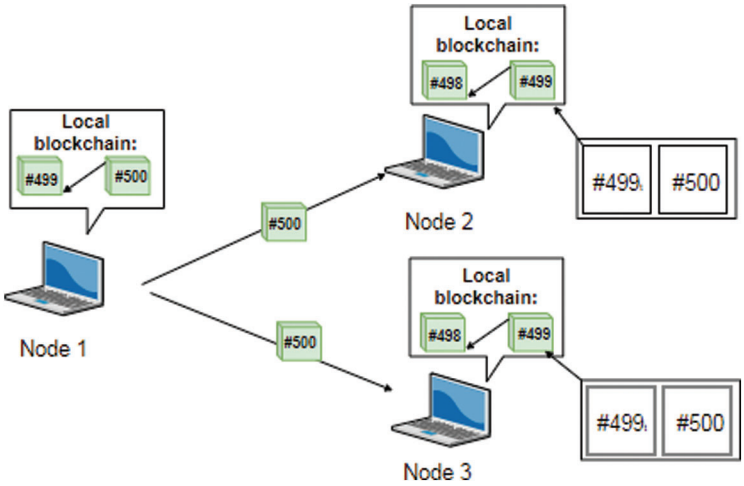

Figure 9: BCC protocol Step 4: Node 1 broadcasts the new block to its neighbors

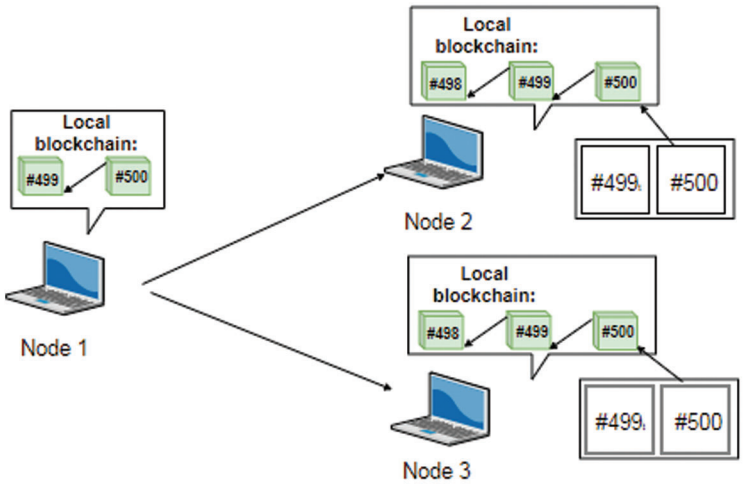

Figure 10: BCC protocol Step 5: The neighboring nodes compare the hash in the new block to the hash that was included in the CCM

4 Experiments and Results

The environment of the simulation was developed using Python programming language. Anaconda environment was used to build the network in order to send and receive requests between the nodes. The simulation implementation has been performed in an Ubuntu operating system with Core i5-8500U CPU 3 GHz. The installed RAM is 4.00 GB. In order to implement Proof of Work, SHA-256 cryptographic hashing algorithm was used. The first block (also known as the genesis block) which does not contain any transactions or previous hash was hardcoded. The tests were carried out on three different networks, that had different topologies and a different number of nodes. 50 nodes, 200 nodes, and 500 nodes were used accordingly.

On the small network, 100 tests were made. 50 tests on the original blockchain, and 50 tests on the proposed blockchain. On the medium size network, 100 tests were made. 50 tests on the original blockchain, and 50 tests on the proposed blockchain. Lastly, on the large network, 100 tests were made. 50 tests on the original blockchain, and 50 tests on the proposed blockchain.

When it comes to block generating, the blockchain was generating 1 block every 10 min. This is done by adjusting the difficulty of the network, also known as the number of leading zeros in a hash. Bitcoin network adjusts the difficulty by comparing how long it took to mine the last 2016 blocks (every two weeks), and adjust the difficulty based on the result. If mining the last 2016 blocks took more than two weeks, then, the network will make the target value easier and vice versa.

After multiple experiments in selecting the network difficulty, it has been decided that the best choice for this simulation's environment was to produce a hash with leading 11 zeros or more. Lastly, in order to give a better probability for the malicious node to generate a block, it was given less strict rules for generating a block. The malicious node was allowed to produce a block by generating a hash that starts with 9 zeros or more. The reason this decision was made is to save time in order to carry more experiments. However, in order to ensure that there is a balance in the network the malicious node was allowed to attack once when the value of z from Eq. (2) equals zero. Remember that if the value of z is 0. Once the value of z became 1, the simulation round was over, indicating that the malicious node did not succeed in attacking the network. In theory, this should balance out the network's performance. Over time, the malicious node will fall behind and never catch up to the honest nodes who form the majority. Giving the scope of the

simulation, every node is connected to 8 other nodes maximum, and 1 one node minimum. The metrics measured in the simulation are:

- The number of times the malicious node generated a block before any honest node.
- The number of times the attack has succeeded.
- The number of exchanged messages.

4.1 The Number of Time the Malicious Node Successfully Generated a Block before Any Honest Node

Tab. 1 shows that the malicious node in both the original blockchain and the proposed blockchain managed to successfully generate a block within the given condition in order for the malicious node to carry out the simulation. Although the number of times for the malicious node to succeed is purely based on hashing, and has nothing to do with the differences between the original blockchain and the suggested one. However, generating a block does not necessarily mean that the malicious node will be able to successfully manipulate the network.

Table 1: Metrics used in the simulation

	Regular blockchain (50 nodes)	Proposed blockchain (50 nodes)	Regular blockchain (200 nodes)	Proposed blockchain (200 nodes)	Regular blockchain (500 nodes)	Proposed blockchain (500 nodes)
The Number of Time the Malicious Node Successfully Generated a Block Before Any Honest Node	6	5	6	7	4	4
The Number of Times an Attack Has Succeeded	6	0	6	0	3	0
The Percentage of successful attacks	100%	0%	100%	0%	75%	0%

4.2 The Number of Times an Attack Has Succeeded

- The 50 nodes network: Tab. 1 shows the number of times the proposed blockchain managed to prevent an attack on the network compared to the original blockchain. It can be seen that the original blockchain didn't defend itself against any attack out of the 6 attacks. While in the proposed blockchain, the network managed to defend itself against all attacks.
- The 200 nodes network: Tab. 1 shows the number of times a successful attack has happened. It can be seen that the original blockchain didn't defend itself against any attack out of the 6 attacks. While in the proposed blockchain, the network managed to defend itself against all attacks.
- The 500 nodes network: Tab. 1 shows the number of times a successful attack has happened. Given the large number of nodes in the large network, the original blockchain network managed to defend itself against one attack. The network managed to defend itself because an honest node generated a new block at the same time as the attack was happening, and the block generated by the honest node propagated to more nodes in the network before the block that was generated by the malicious node. Meaning that the original blockchain network managed to defend itself against 1 attack out of 4 as seen in Fig. 11. While in the proposed blockchain, the network managed to defend itself against all attacks.

4.3 The Number of Exchanged Messages

The number of exchanged messages in the proposed blockchain is a little bit higher than the original blockchain. This is due to the fact that the nodes in the suggested blockchain exchange CCM messages every time a new block is generated. Denoting the number of nodes in the network by n, the nodes in the

original blockchain exchanged n-1 messages over a 10 minutes period of time (the average time it takes to generate a block). All of the messages were block broadcasting messages. In the proposed blockchain, however, Denoting the neighbors of a node by m, the nodes in the network exchanged (n−1) + (m*CCM) messages over the same period of time as seen in Fig. 12.

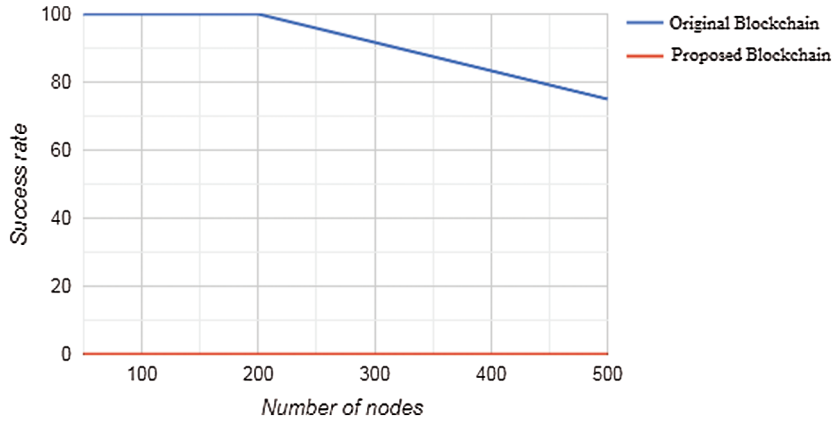

Figure 11: The success rate of the malicious node attack on the network

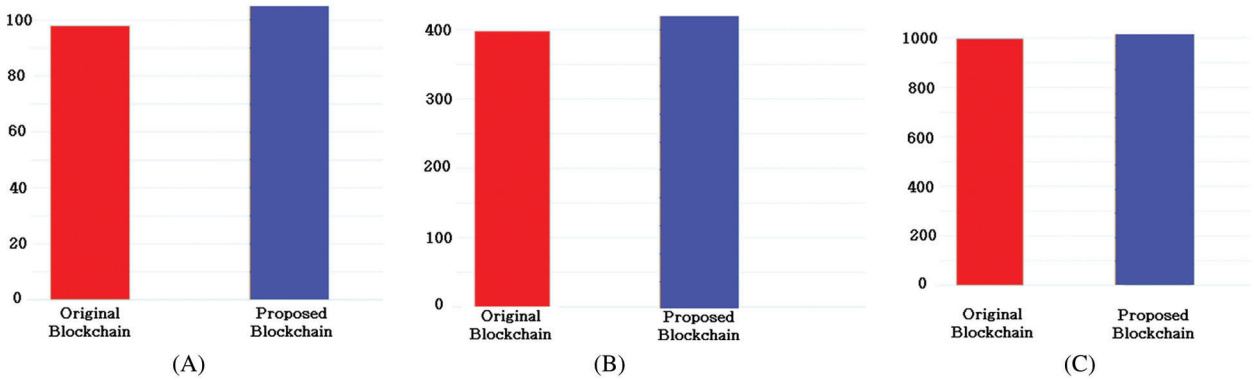

Figure 12: (A) The number of exchanged messages over a 10 minute period in the 50 nodes network (B) The number of exchanged messages over a 10 minute period in the 200 nodes network (C) The number of exchanged messages over a 10 minute period in the 500 nodes network

Fig. 12 shows the number of exchanged messages over a 10 min period in the three networks. (A) show that the nodes in the small network exchanged 98 messages in the original blockchain and 105 messages in the proposed blockchain. While (B) shows that the nodes in the medium network exchanged 398 messages over 10 minutes period in the original blockchain and 403 in the proposed blockchain. Lastly, (C) shows that the nodes exchanged 998 in the original blockchain in the large network, and 1002 messages in the proposed blockchain.

5 Conclusion

The original structure of the blockchain is not secure enough. Because in blockchain security is achieved by saving the transaction information of a block in the next block. However, this means that the last block will be always vulnerable and open for manipulation by malicious nodes. There is a lot of room for such an

attack to happen since the nodes in the network contact each other only when a new block is generated. Moreover, the "longest chain wins" rule forces the nodes to trust whatever node has the longest chain, even if the longest chain contained fake blocks.

This paper proposes an algorithm that further enhances the security of the blockchain and ensures that the last block in the blockchain remains secure even when a malicious node tries to manipulate it. The proposed algorithm makes the nodes send a consistency check message to its neighbors every time a new block is generated. This message contains the Merkle-Tree hashes of the last two blocks in the blockchain. The neighbors of a node will investigate the hashes included in the CCM and based on the result, the neighbors will decide whether or not to broadcast the new block.

The simulation results show that the proposed blockchain managed to prevent 100% of the attacks in the small size network. And managed to prevent 100% of the attacks in the medium-size network. And managed to prevent 100% of the attacks in the large size network

While it is been proven that the proposed method offers more security to the network, this security comes at the price of the number of exchanged messages. Because in the proposed method the nodes send an extra message before broadcasting a block, while in the original blockchain the nodes only contact each other when a new block needs to be broadcasted.

Acknowledgement: We would like to express our very great appreciation to the Chungnam National University department of Radio and Information Communication Engineering for providing the necessary tools for this project.

Funding Statement: This work was supported by research fund of Chungnam National University.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] S. Nakamoto, "Bitcoin whitepaper," 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf-(Дата обращения: 17.07. 2019).

[2] B. Singhal, G. Dhameja and P. S. Panda, *Beginning Blockchain*. Berlin, Germany: Apress publications, 2018.

[3] M. Nofer, "Blockchain," *Business & Information Systems Engineering*, vol. 59, no. 3, pp. 183–187, 2017.

[4] J. Abadi and M. Brunnermeier, "Blockchain economics," *National Bureau of Economic Research*, 2018.

[5] M. Pilkington, "Blockchain technology: principles and applications," Cheltenham, UK: Edward Elgar Publishing, 2016. [Online]. Available: https://doi.org/10.4337/9781784717766.00019.

[6] https://en.wikipedia.org/wiki/Fork_(blockchain). 2020.

[7] I. Lin and T. Liao, "A survey of blockchain security issues and challenges," *IJ Network Security*, vol. 19, no. 5, pp. 653–659, 2017.

[8] C. Natoli and V. Gramoli, "The blockchain anomaly," in *2016 IEEE 15th Int. Sym. on Network Computing and Applications (NCA)*, IEEE, Cambridge, MA, USA, pp. 310–317, 2016.

[9] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," in *Int. Conf. on Financial Cryptography and Data Security.* Berlin, Heidelberg: Springer, pp. 436–454, 2014.

[10] E. Heilman, "One weird trick to stop selfish miners: Fresh Bitcoins a solution for the honest miner," *Financial Cryptography and Data Security*, Christ Church, Barbados, pp. 161–162, 2014.

[11] R. Zhang and B. Preneel, "Publish or perish: A backward-compatible defense against selfish mining in bitcoin," in *Cryptographers' Track at the RSA Conf.*, Cham: Springer, pp. 277–292, 2017.

[12] R. Zhang and B. Preneel, "Broadcasting intermediate blocks as a defense mechanism against selfish-mine in bitcoin," in *IACR Cryptol. ePrint Archive*, 2015. [Online]. Available: https://eprint.iacr.org/2015/518.pdf.

[13] S. Solat and M. Potop-Butucaru, "Brief announcement: zeroblock: Timestamp-free prevention of block-withholding attack in bitcoin," in *Int. Sym. on Stabilization, Safety, and Security of Distributed Systems*, Cham: Springer, pp. 356–360, 2017.

[14] N. T. Courtois and L. Bahack, "On subversive miner strategies and block withholding attack in bitcoin digital currency," in *arXiv preprint arXiv*,1402.1718, 2014.

[15] M. Crosby, "Blockchain technology: Beyond bitcoin," *Applied Innovation*, no. 2, pp. 6–19, 2016.