International Journal of Grid and Distributed Computing
Vol. 10, No. 12 (2017), pp.93-106
http://dx.doi.org/10.14257/ijgdc.2017.10.12.09

Blockchain Consensus Rule Based Dynamic Blind Voting for Non-Dependency Transaction

Soohwan Cho[1], Soo Yong Park[†] and Sang Rok Lee [2]

Dept. of Computer Science & Engineering, Sogang University, Seoul, South Korea soohwancho@outlook.com, sypark@sogang.ac.kr, vhdys251@gmail.com

Abstract

As in 2009 the first encrypted electronic currency Bitcoin is introduced, the block chain technology has received attention. Bitcoin is a currency issued without a trusted third party, in peer-to-peer (Peer-to-Peer) network and gives a guarantee of the reliability of money transactions, ensures the integrity of transactions. There is a back end-technique called block-chain that guarantees these without trusted third party. Block chain is a public ledger that all nodes to verify transactions and maintain integrity with the same ledger. Block chain technology applies not only electronic currency but also in various industries such as contract document integrity, cloud storage service, internet of thigns, supply chain for transaction integrity. However, there are problems to be solved if the bitcoin block chain consensus rule is applied to these industry. First, there is problem of low transaction throughput and block generation rate by an inefficient consensus rule. Second, There is problem of tampered block broadcast with non-dependency (internet of things data, food history data) transactions when applying the block chain consensus rule, which is proof-of-work of bitcoin's competitive system.

To solve these problems, we propose new blockchain consensus algorithm for non-dependency transactions. In this paper, the blockchain transactions targeted internet of things data and food history. We propose consensus rule based voting for the efficiency, also propose method of imposing difficulty of block generation according to the voting result in order to prevent malicious node attacks. We provide experimental results on attack malicious node in the block chain implemented by the proposed consensus algorithms and the result is analyzed

Keywords: *Blockchain, Consensus Rule, internet of things, Supply chain, Cold chain*

1. Introduction

Bitcoin blockchain technology applies various industries [2][3] such as contract document integrity, cloud storage service, internet of thigns, supply chain for transaction integrity. There is consensus algorithm [1] to maintain the transaction integrity of the bitcoin. The consensus rule of the bitcoin blockchain is an agreement on which blocks the nodes select on the network. To maintain the network, the nodes must select blocks that are not to be tampered. In the bitcoin, the nodes compete for the mining of the blocks. Therefore, in order for malicious node to generate new block, it must incur more opportunity costs than other nodes. When new block is sent to the network, each node validates the block before it is broadcast to connected nodes. only valid blocks broadcast over the network. In order for malicious node to tampered transaction (double spending, generate money), it must consume enormous power to create blocks and tampered all the blocks of more than 51% of the nodes participating in the network. If more than 51% of the good nodes exist, the same block (integrity of the transaction) can be maintained all over the bitcoin network

Received (September 7, 2017), Review Result (December 2, 2017), Accepted (December 7, 2017)

However, if bitcoin blockchain consensus rule is applied to the internet of things and supply chain industries based on the non-dependency transaction, there are the following problems. First, there are problems of low transaction throughput and block generation rate by inefficient consensus rule. This is not appropriate for internet of thigns and supply chain industry, which need to process transactions at high performance and low computational resources. Second, non-dependency transactions (internet of things and food history transactions) is problem of the block broadcast including the tampered transactions. Case of electronic coin transaction, it is dependency transaction that can verify the next transaction with the previous transaction. Therefore, even if malicious node generates a block containing tampered transaction, other nodes can independently perform block verification through their transaction history. If there are more than 51% of the good nodes in network, the integrity of the transaction is maintained.

However, in most cases of internet of things and food history transactions are non-dependency transaction, so the next transaction cannot be verified from the previous transaction. The [Figure 1] below shows the problem of bitcoin consensus rule of non-dependency transactions.

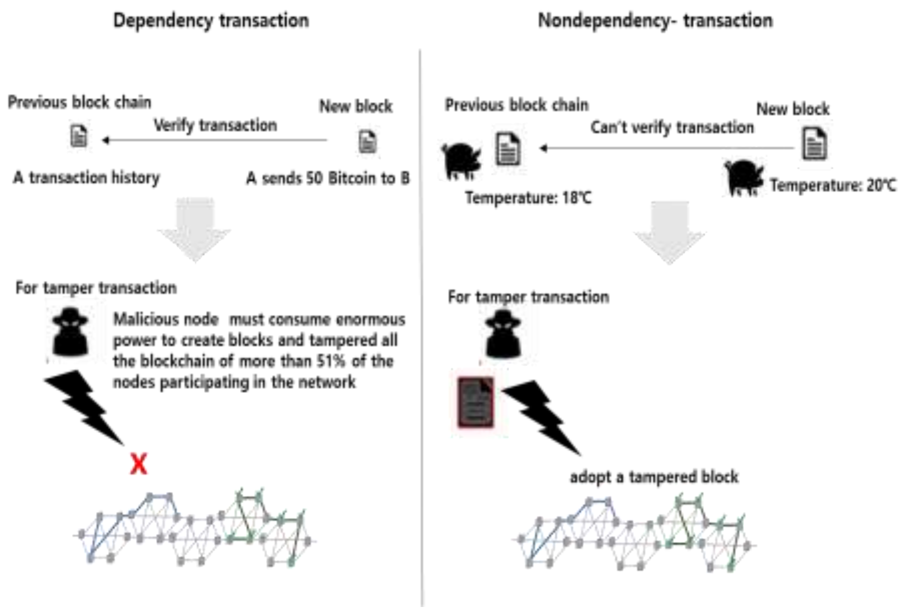


Figure 1. Problem Bitcoin Consensue Rule

In the case of internet of thigns, it can't verify the current temperature and humidity measurement values with the previous temperature and humidity measurement values. Because non-depency transactions cannot verify the next transaction, the consensus rule of proof-of-work is problematic in the validation of the block. If malicious node generates a tampered block. Since other nodes cannot verify the transaction of the tampered block, they adopt the tampered block. In bitcoin consensue rule for non-dependency transaction, even if there is more than 51% of good nodes, the integrity of the data cannot be maintained.

Proof of stake(PoS) based algorithms have recently been introduced that improve the inefficiency bitcoin blockchain consensus rule. Proof of stake has the advantage of being able to make immediate consensus without wasting energy because voting through stakes. These proof-of-stake based consensus are the Delegated proof of stake(DPoS) and Tendermint. DpoS is consensus rule in which stakeholder vote to elect block verification delegated. Stakeholders keep their security because they have assets. TenderMint is consensus rule that allows each node to receive stake base on the total amount of money held and thus to make decisions. If node performs verification with flase, it guarantees integrity by depriving all the stakes of that node. However, Pos-based consensus algorithms have the following problems in non-dependency transactions. It is difficult to prove that non-dependncy transaction is stake because it is not transaction of electronic currency. In the case of proof of stake, the stakeholder has the right to make decisions. This makes it possible to attack from a malicious node by tracking stake.

In this paper, we propose the high performance block chain consensus rule that maintains integrity as long as there are 51% of good nodes for non-dependency transactions. For these goals, propose voting-based consensus rule. Generally voting-based consensus has certain central decision-making nodes, or there exists round-robin way of communicating information between nodes. If there is specific central node for decision, there is possibility that the tampered of the data by decision node, and the malicious node to attack the decision authority. To do this, we propose dynamic blind voting. Also the malicious node attempts to tampered block. To prevent this, we propose way of generating a block that adjusts the proof-of-work of the work by voting results.

We implement the block chain applying this consensus rule and verify the consensus rule according to the increase of the malicious node.

The rest of the paper is organized as follows. In Section II refer related work, in section III propose consensus algorithm is described. In Section IV the implemented system model is described and the provide experimental results on attack malicious node in the block chain implemented by the proposed consensus algorithms and the result is analyzed. Finally, conclusions are drawn in Section V.

2. Related Work

2.1. Delegated Proof of Stake

With the problems mentioned in Chapter 1, PoS based consensus rule [6] is not appropriate to apply the blockchain consensus rule of the bitcoin to maintain the security of the non-dependency transaction. Delegated Proof of Stake is used in BitShares, one of the block-chain applications, to solve the problems of power consumption and block generation time in the proof-of-work.

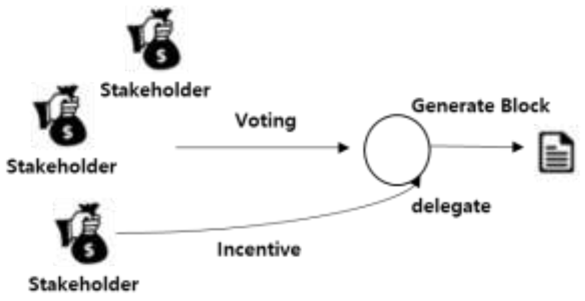


Figure 2. Delegated Proof of Stake

Figure 2 shows dekegated proof-of-stake(DPoS). DpoS is consensus rule in which stakeholder vote to elect block verification delegated. Stakeholders keep their security because they have assets. The delegates selected by proof of stake are only authorized to verify the entire transaction and to generate blocks. Stakeholders give the delegate an incentive to generate blocks, and delegates create more blocks honestly for incentives.

However, DpoS consensus algorithms have the following problems in non-dependency transactions. It is difficult to prove that non-dependncy transaction is stake because it is not transaction of electronic currency. If the node storing the most non-dependable transactions becomes stakeholder, there is problem that the integrity of the data is violated if some nodes with high stake collide. Also, if a malicious hacker attacks a node with the largest stake, the network will be destroyed

2.2. Tendermint

The Tender Mint allows each node to receive a stake based on the total amount of money held and thus to have the right to make decisions. At this time, if node performs verification with false, it guarantees integrity by depriving all the stake of that node.

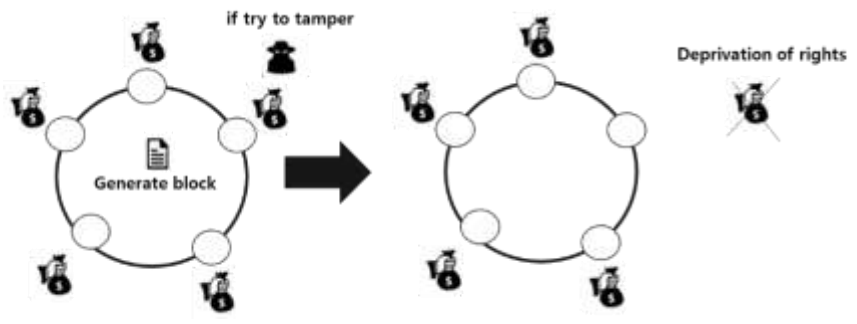
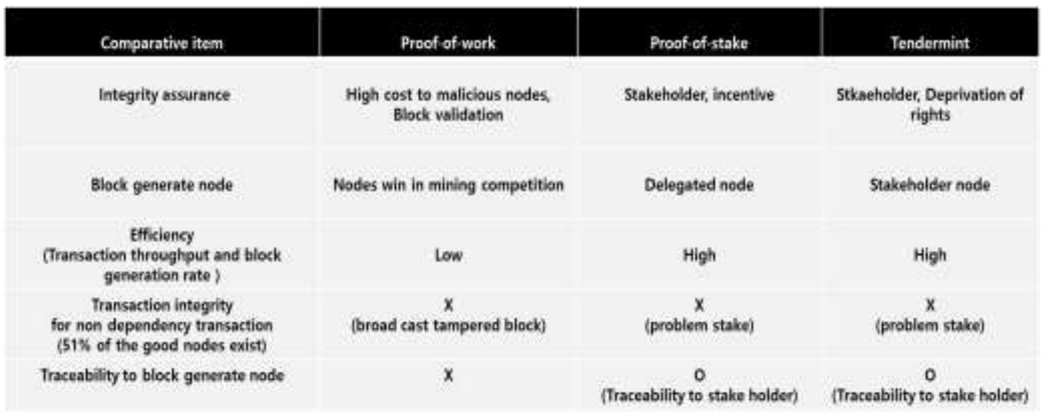


Figure 3. Tendermint Consensus Rule

This consensus rule remove the proof-of-work and improves overall computing resource efficiency through stake-based agreements. Integrity is maintained through punitive processing of tamper and falsification. Like DpoS, It is difficult to prove that non-dependncy transaction is stake because it is not transaction of electronic currency. If the node storing the most non-dependable transactions becomes stakeholder, there is problem that the integrity of the data is violated if some nodes with high stake collide

2.3. Comparative Related Work

Table 1. Comparative Blockchain Consensus Rule



Comparative item	Proof-of-work	Proof-of-stake	Tendermint
Integrity assurance	High cost to malicious nodes, Block validation	Stakeholder, incentive	Stkaeholder, Deprivation of rights
Block generate node	Nodes win in mining competition	Delegated node	Stakeholder node
Efficiency (Transaction throughput and block generation rate)	Low	High	High
Transaction integrity for non dependency transaction (51% of the good nodes exist)	X (broad cast tampered block)	X (problem stake)	X (problem stake)
Traceability to block generate node	X	O (Traceability to stake holder)	O (Traceability to stake holder)

In Section 2 we looked at the research on improvement of block chain consensus rule. [Table 1] compares the bit coin blockchain with the consensus rule of related work.

As shown in the table, each consensus algorithm guarantees transactional integrity of the electronic currency system. Bit coin shows low efficiency, and Pos-based consensus algorithms show high efficiency. However, transactional integrity in non-dependency transactions, In the case of bit coin, there is a problem of tampered block broadcast. Pos-based algorithms have difficulty in proof of stake. Traceability was unknown in the case of bit coin, Pos-based algorithms have the problem that malicious hackers can track with equity. In conclusion, the consensus algorithm for non-dependency transactions should be highly efficient, maintain transaction integrity, and have no traceability to the block generator.

3. Approach

3.1. Inter-Node Blinding Votes Using the Final Block Vote Hash Value

As noted in Section 1, the majority decision of the central decision-making node, round-robin, is not appropriate because of the possibility of tampered.

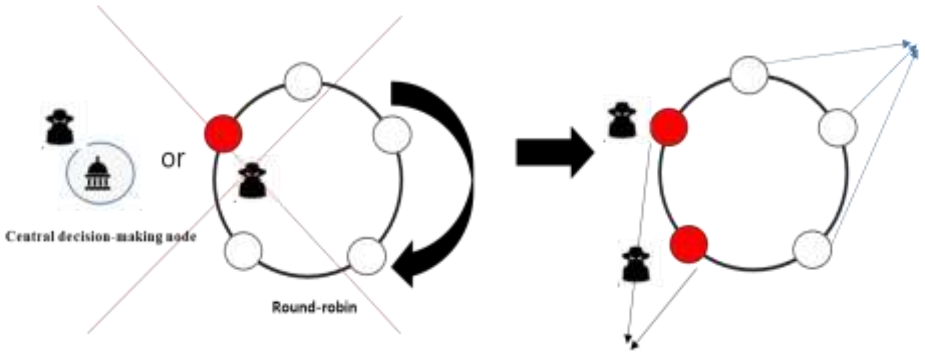
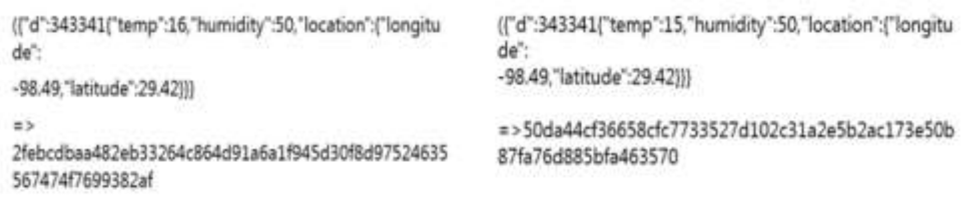


Figure 4. Concept Blinding Vote

As shown in [Figure 4], in this paper, we propose blind voting that each node is independent, between nodes using a block hash and a merkle tree where a large number of decisions occur dynamically, without knowing which node to select. All nodes store the transferred transactions in a transaction pool according to certain even number. For each transaction, hash with SHA256 function. If the internet of things transaction is hashed, it is represented by the 256-bit number as shown in [Figure 5]



```
({"d":343341{"temp":16,"humidity":50,"location":{"longitu
de":
-98.49,"latitude":29.42}}}
=>
2febcdbaa482eb33264c864d91a6a1f945d30f8d97524635
567474f7699382af

({"d":343341{"temp":15,"humidity":50,"location":{"longitu
de":
-98.49,"latitude":29.42}}}
=>50da44cf36658cfc7733527d102c31a2e5b2ac173e50b
87fa76d885bfa463570
```

Figure 5. Hash Result of Internet of Things

As shown in [Figure 5], the actual hash function changes completely when the input value changes. The only way to find the input value with the result value is to assign the number of all cases. After the hash is done, the hash is done once again in even number of pairs.

Repeating this way, the Merkle tree shown in [Figure 6] is completed. If all the nodes have the same transaction using the merkel tree feature, the merkel tree root value will be the same for all nodes.

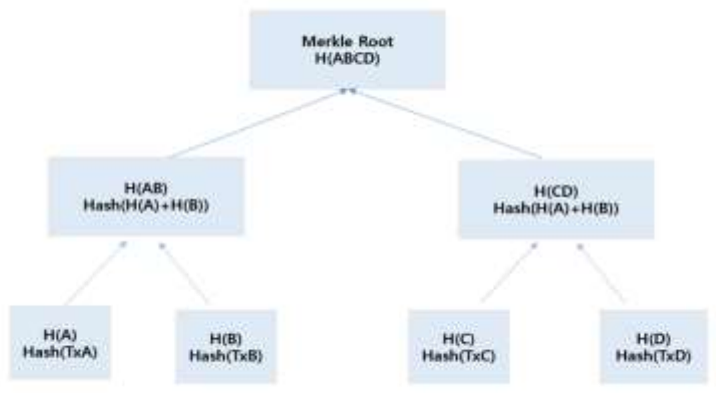


Figure 6. Merkle Tree

That is, if transaction is tampered by a malicious attacker, the merkle tree root hash value will be incorrect. Each node hashes the SHA-256 once again with the value of the last block chain hash added to the merkle root hash value for the transaction it has. This hash value is called the final block vote hash value in this paper. After that, it performs the mod operators as many as the number of nodes participating in the block chain.

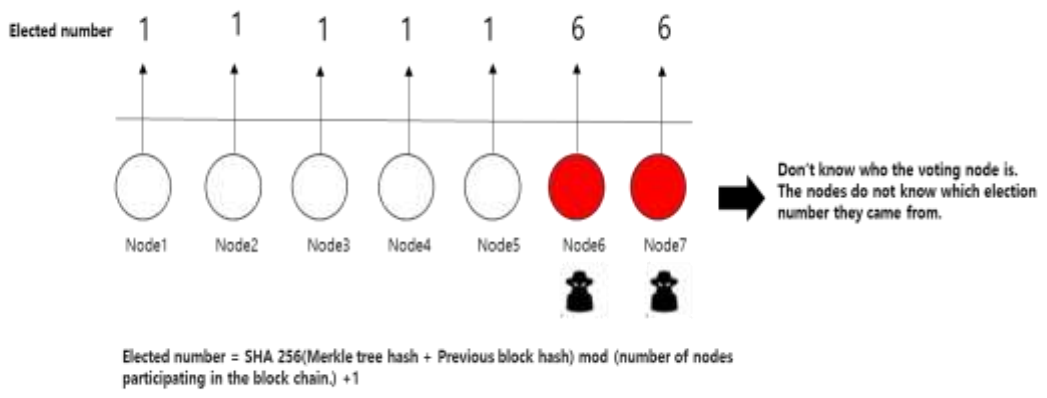

Elected number
1
1
1
1
1
6
6
Node1
Node2
Node3
Node4
Node5
Node6
Node7
Don't know who the voting node is.
The nodes do not know which election number they came from.
Elected number = SHA 256(Merkle tree hash + Previous block hash) mod (number of nodes participating in the block chain.) +1

Figure 7. Selected Numbers Using Hash of Final Block Vote Hash Value

As shown in [Figure 7] if the result of the mod operation of the number of nodes in the final block voting hash value is 0, the election number is 1. The malicious node is 5, so the number is 6. If all the nodes have transactions that are not tampered, they will vote for the same node. If the transaction is tampered by malicious attacker, different election number will appear.

[Figure 8] shows that the voting message is transmitted according to the election results. Each node sends a message to a node with the same index as an independently selected number. Node 1 becomes decision node dynamically, get 5 vote result. Node 6 becomes decision node dynamically, get 2 vote result. The node that received the message becomes the decision node and has the authority to generate the block. The reason for selecting block generation node through this final block vote hash value is that because the merkle tree is created dynamically each time, so it is method for eliminating traceability to block generator and collecting large number of decisions. As long as 51% more of good nodes exist, a large number of decision nodes are always selected. If tamper occurred, there would be block generation node selected from the honest node and block generation node selected from the malicious node.

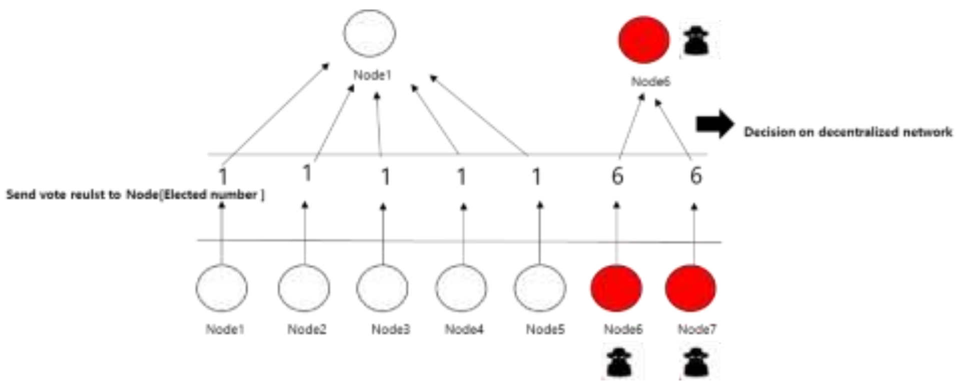

Node1
Node6
Decision on decentralized network
Send vote reulst to Node[Elected number]
1
1
1
1
1
6
6
Node1
Node2
Node3
Node4
Node5
Node6
Node7

Figure 8. Send a Message by Election Number

3.2. Proof-of-Work Difficulty Setting Block Generation by Voting Result

As noted Section 3.1, if no tampered transaction occurred, all nodes would have elected the same node as the block generator, the block generation node would have received as many votes as the number of nodes participating in the network. Decision nodes, which receives large number of votes, broadcast honest block. The problem is that malicious nodes will also attempt to broadcast tampered blocks. To prevent the malicious node from generating a block, set the difficulty of proof of work according to the voting result. In other words, it increases the opportunity cost to the malicious node to prevent the block generation. The decision node is mined through proof-of-work for block generation. Proof-of-work is done as follows.

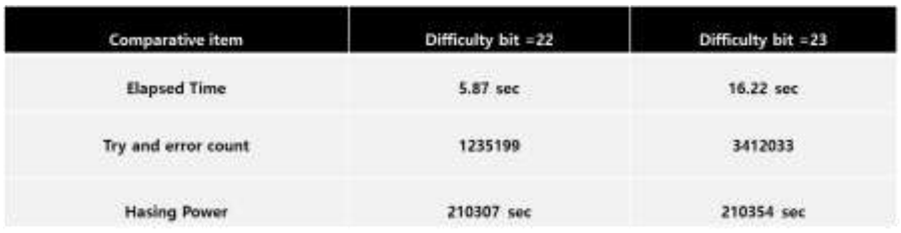
$$Target\ Value = 2^{256-difficulty\ bit}$$

$$Block\ info = previous\_block\_hash + Merkleroot\_hash$$

$$Block\_Hash = SHA\ 256(Block\ info + nonce) <= Target\ Value$$

Difficulty Bits is number that sets the difficulty level for generating blocks. A node that receives a lot of votes will set small difficulty bit, and a node that receives few votes will set large difficulty bit. Block_info is the block's previous block hash (Previous_Hash) add the Merkleoot_hash value. Then, the value obtained by adding Block_info and nonce value is hashed through SAH256 and nonce value that is smaller than Target Value is found.

Table 2. Results According to Difficulty Bits



Comparative item	Difficulty bit =22	Difficulty bit =23
Elapsed Time	5.87 sec	16.22 sec
Try and error count	1235199	3412033
Hasing Power	210307 sec	210354 sec

[Table 2] shows that difficulty bits are compared between 22 and 23. The nonce value was found to be about 2.7 times faster than that for 22bit at 5.87sec and 23.22sec at 23bit, the number of attempts was 1235199 and 3412033, more than twice the difference. Assign low difficulty bit to node that receives large number of vote, and set high difficulty bit for malicious node to suppress block generation. [Figure 9] shows the number of cases in which malicious nodes and honest nodes are elected. Looking at 6: 1 (honest node: malicious node), node 1 received many votes as honest node. In such case, the difficulty bit is set lower than the malicious node and the block generation speed is increased.

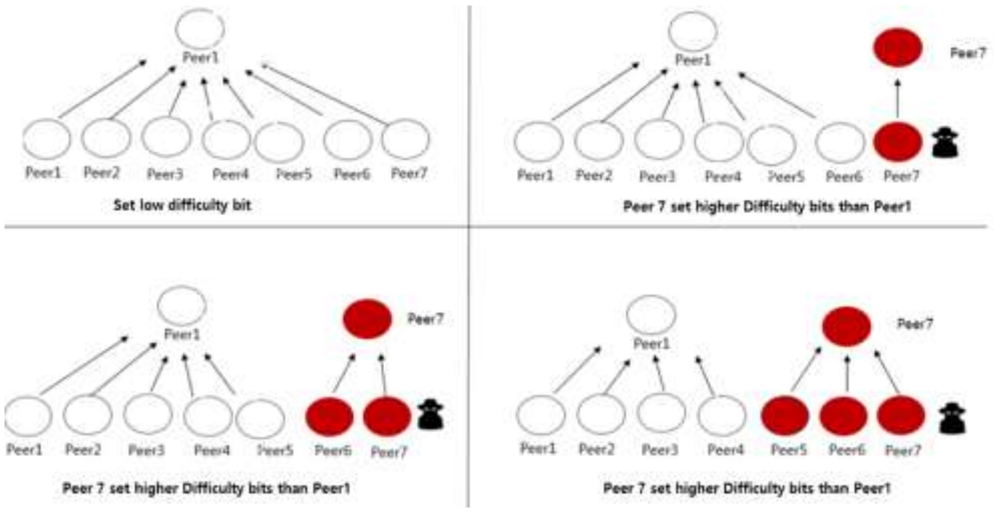

Peer1
Peer1
Peer2
Peer3
Peer4
Peer5
Peer6
Peer7
Set low difficulty bit
Peer7
Peer 7 set higher Difficulty bits than Peer1
Peer 7 set higher Difficulty bits than Peer1
Peer 7 set higher Difficulty bits than Peer1

Figure 9. Assign Difficulty Bit According to Vote Result

The block generation by difficulty control is as shown in [Figure 10]. If malicious attacker consumes a lot of power they can generate a block, resulting in difference in block generation speed. The longest block chain is not tampered, reflecting large number of decisions, short block chain is automatically taken over the network

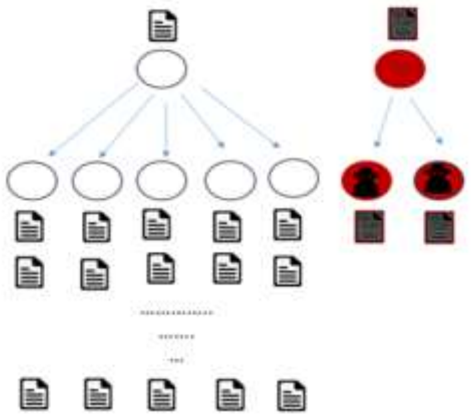
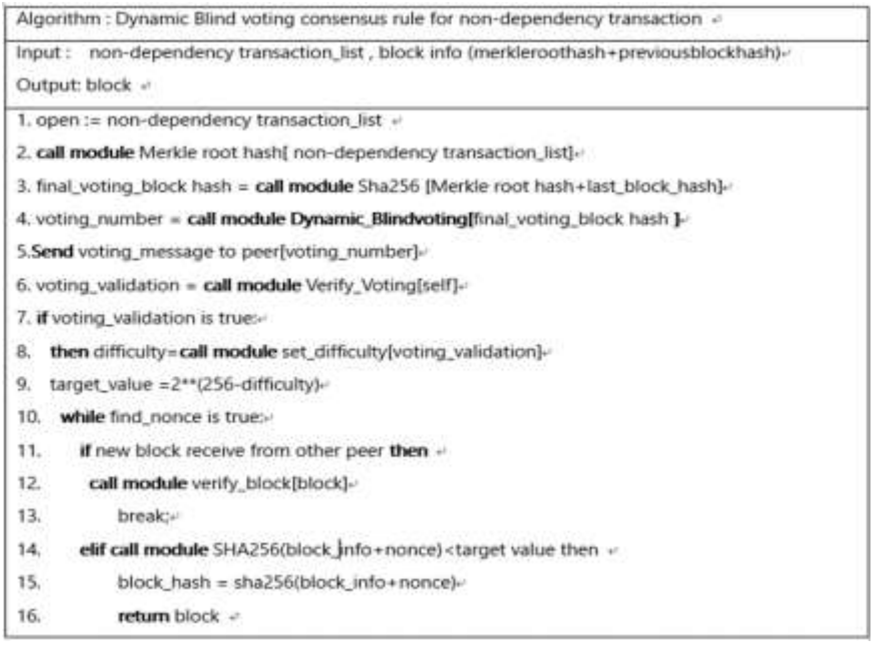


Figure 10. Differences in Block Generation

3.3. Propose consensus Algorithm Pseudocode and Process Description

In the proposed algorithm, the nodes that received large number of decisions dynamically by using the final block voting hash value become the decision node. In order to prevent malicious nodes from generating blocks, we set the difficulty according to the voting result. The following [Figure 11] shows the pseudo code of the proposed algorithm. Each node computes merkel tree value for the transaction. The sum of the hash value of the merkle tree and the value of the last block hash is hashed through the SHA 256 function. This is the final block vote hash value. After that, it performs the mod operators as many as the number of nodes participating in the block chain then add 1. This value is the voting_number. Each node votes independently to the node that has the index corresponding to the voting_number.



```
Algorithm : Dynamic Blind voting consensus rule for non-dependency transaction
Input :   non-dependency transaction_list , block info (merkleroothash+previousblockhash)
Output: block
1. open := non-dependency transaction_list
2. call module Merkle root hash[ non-dependency transaction_list]
3. final_voting_block hash = call module Sha256 [Merkle root hash+last_block_hash]
4. voting_number = call module Dynamic_Blindvoting[final_voting_block hash ]
5.Send voting_message to peer[voting_number]
6. voting_validation = call module Verify_Voting[self]
7. if voting_validation is true:
8.    then difficulty=call module set_difficulty[voting_validation]
9.    target_value =2**(256-difficulty)
10.    while find_nonce is true:
11.       if new block receive from other peer then
12.         call module verify_block[block]
13.            break;
14.       elif call module SHA256(block_info+nonce)<target value then
15.            block_hash = sha256(block_info+nonce)
16.            return block
```

Figure 11. Pseudo Code of Proposed Algorithm

The selected node receives the difficulty according to the voting result, and generates the block.

If there are more than 51% good nodes, then majority of votes will form longest block chain than the malicious node. The node that has propagated the block of the selected node verifies the block by using its own vote and the final block vote hash value. If they do not match, discard them and re-vote them with the next transaction. The verification prevents arbitrary malicious nodes from being selected.

4. System Implementation and Experiment

4.1. Block Chain Experimental Environment and Implementation

In this paper, to verify the consensus rule of the propose, we implemented block chain that stores non-dependency transaction - internet of thigns data and food history data .We measure transaction throughput and block generation rate for the efficiency of the applied consensus algorithm. Then, the integrity verification is performed according to increase in the number of malicious nodes in the network. Section 4 describe the experimental environment, important components of the implemented block chain, experiment and verification. In this paper, we define a transaction in the block chain as one of the information sent from internet of things device and food processing solution. The transactions on the internet of things saved the Rule engine event that needed the most reliability from the Daliworks company's IoT platform and stored the food history data generated by Nongshim's food history data.

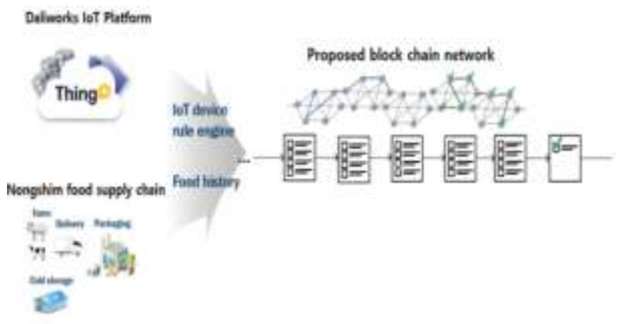
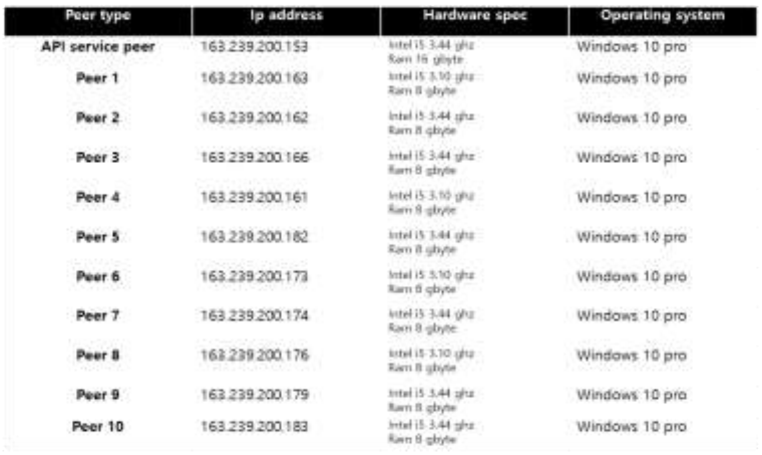


Figure 11. Store Non-Dependency Transaction

When 10 transactions are transferred, the nodes are set to generate blocks. The number of nodes in the block chain is 11 as shown in [Table 4], and consists of nodes with general computing resources.

Table 2. Block Chain Configuration Node



Peer type	Ip address	Hardware spec	Operating system
API service peer	163.239.200.153	Intel i5 3.44 ghz Ram 16 gbyte	Windows 10 pro
Peer 1	163.239.200.163	Intel i5 3.10 ghz Ram 8 gbyte	Windows 10 pro
Peer 2	163.239.200.162	Intel i5 3.44 ghz Ram 8 gbyte	Windows 10 pro
Peer 3	163.239.200.166	Intel i5 3.44 ghz Ram 8 gbyte	Windows 10 pro
Peer 4	163.239.200.161	Intel i5 3.10 ghz Ram 8 gbyte	Windows 10 pro
Peer 5	163.239.200.182	Intel i5 3.44 ghz Ram 8 gbyte	Windows 10 pro
Peer 6	163.239.200.173	Intel i5 3.10 ghz Ram 8 gbyte	Windows 10 pro
Peer 7	163.239.200.174	Intel i5 3.44 ghz Ram 8 gbyte	Windows 10 pro
Peer 8	163.239.200.176	Intel i5 3.10 ghz Ram 8 gbyte	Windows 10 pro
Peer 9	163.239.200.179	Intel i5 3.44 ghz Ram 8 gbyte	Windows 10 pro
Peer 10	163.239.200.183	Intel i5 3.44 ghz Ram 8 gbyte	Windows 10 pro

We implement private block chain (project name: Log chain)using the block consensus rule proposed in this paper. It was developed in the Python language, and the data format was the same as the existing block chain.

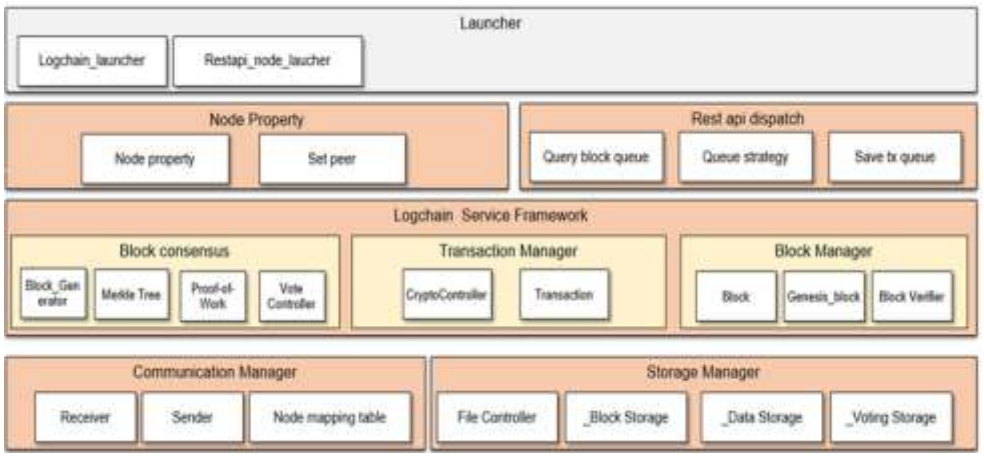


Figure 11. Implemented Block Chain Architecture

As shown in Figure 11, the configuration of package in private block chain consists of elements for node to execute. The main modules are described below.

- Logchian_launcher : Launcher for running node
- Rest_Api_launcher: Node for handling external service requests
- Msg_dispatch: b_type_queue_thread, t_type_queue_thread, v_type_queue_thread
- P2P: peer to peer network manager, Node_mapping_table, Receiver, Sender
- File_Controller: Controller of stored files
- Block Consensus: Propose consensus algorithm

4.2. Experiments and Results

In the 11-node block-chain network, the block generation rate and transaction throughput were measured for efficiency verification. When 10 transactions occur, the transaction throughput and the block generation rate are calculated by the following formula, and the results are shown in the table below.

- Transaction Throughput: When generating a block, the number of transactions per block, number of transactions/ (Generate block/sec)
- Block generation rate: The rate at which a minor generates a block through a block consensus process.

Table 3. Test Result

Transaction Throughput	96.00117188930528
Block generation rate	0.312496185303 sec

Actual test results are the result of 10 transactions and 16 bits of difficulty bits in a lot of decisions. When the low difficulty bit is given, the block generation rate converges to zero and the transaction throughput cannot be calculated. If the number of transactions in the block generation and the difigiculty bits are set lower than 16 bits for a large number of decisions, the proposed algorithm can be more efficient.

We measured,10-minute block generation count, number of attempts per block, time required depending on the increase in malicious nodes (9: 1.8: 2, 7: 3). The Diffculty bit decreased by 2 bits in 10bit, 8bit, and 6bit as the number of malicious nodes increased.

- When the malicious node is 1

Among 10 nodes, when the malicious node is 1, the Difficulty bit difference is set to 10 bits, 17 bits are assigned to 9 peers, and 27 bits are allocated to 1 peer.

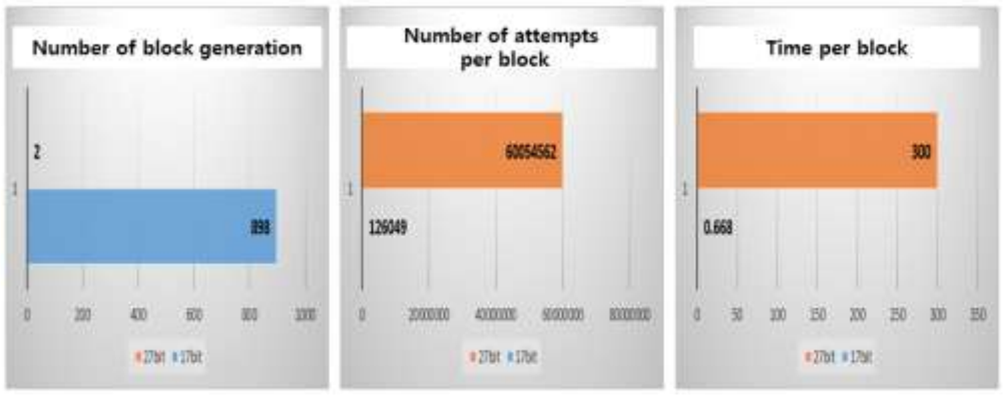


Figure 11. Experiment Result

[Figure 11] shows the results of the experiment. The number of blocks generated is 898 for good nodes and 2 for malicious nodes. The good node generated about 449 times as many blocks. Malicious node could make a block only if it had to try about 476 times. The time required per block was 300 seconds for malicious nodes and 0.668 seconds for good nodes. The time taken per block took about 499 times the time it takes for a malicious node to generate a block.

• When the malicious node is 2
Among 10 nodes, when the malicious node is 2, the Difficulty bit difference is set to 8 bits, 18 bits are assigned to 8 peers, and 26 bits are allocated to 2 peer.

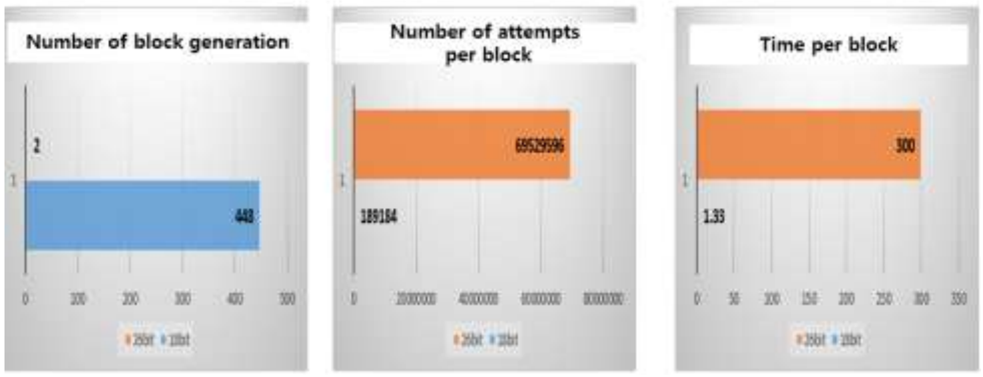


Figure 12. Experiment Result

[Figure 12] shows the results of the experiment. The number of blocks generated is 448 for good nodes and 2 for malicious nodes. The good node generated about 224 times as many blocks. Malicious node could make a block only if it had to try about 367 times. The time required per block was 300 seconds for malicious nodes and 1.33 seconds for good nodes. The time taken per block took about 225 times the time it takes for a malicious node to generate a block

• When the malicious node is 3
Among 10 nodes, when the malicious node is3, the Difficulty bit difference is set to 6 bits, 19 bits are assigned to 7 peers, and 25 bits are allocated to 3 peer.

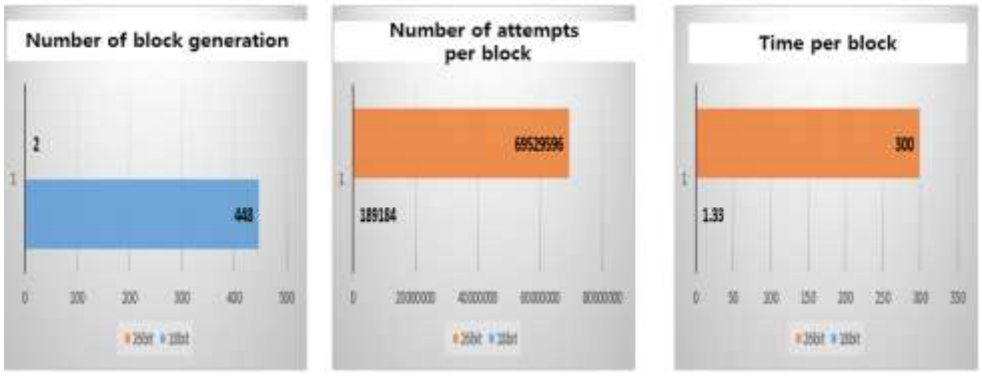


Figure 13. Experiment Result

[Figure 13] shows the results of the experiment. The number of blocks generated is 197 for good nodes and 6 for malicious nodes. The good node generated about 32 times as many blocks. Malicious node could make a block only if it had to try about

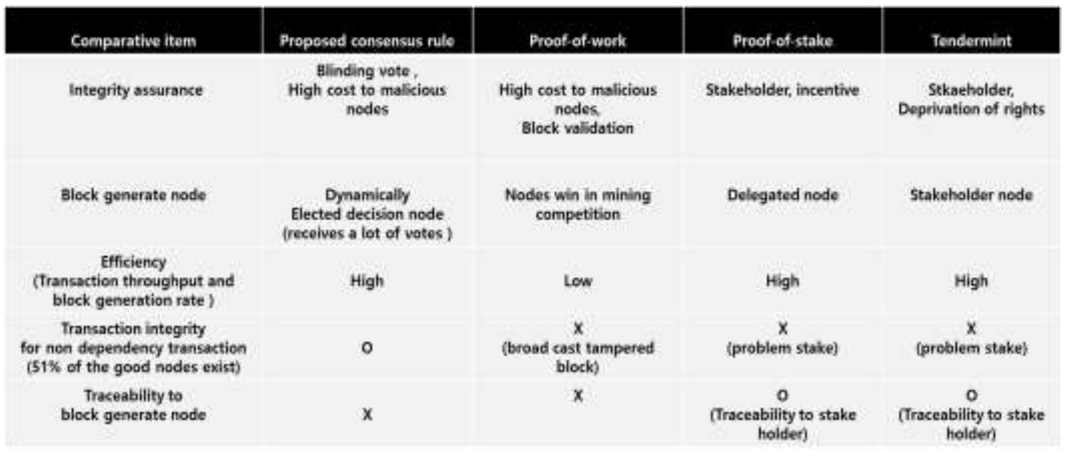
34 times. The time required per block was 100 seconds for malicious nodes and 3.04 seconds for good nodes. The time taken per block took about 32 times the time it takes for a malicious node to generate a block.

For verify the proposed algorithm, we measured,10-minute block generation count, number of attempts per block, time required depending on the increase in malicious nodes.

Even if the number of malicious nodes increases, the good node generates many blocks according to the blinding voting result. This showed that malicious nodes were taken over the network and showed that if more than 50% of the nodes were not tampered, the integrity of the data was maintained.

5. Conclusion

Bitcoin and Pos based of block chains consensus rule is not appropriate for non-dependency- transaction. In this paper, we propose block mining method by blind voting between nodes for large number of decision. The proposed algorithm was verified by implementing private block chain. In the final block vote, the block generator is dynamically selected by performing the mod operation on the hash value, and the result of the voting is different from one transaction even if it is tampered or modified. The result of the vote raised the difficulty of block mining. Honest nodes formed longest block chain, malicious nodes have been taken away from the network because of the opportunity cost of block generation. If there are more than 51% of honest nodes, ensuring the integrity of non-dependent transactions. [Table 4] compares the proposed consensus rule with the related work. If 51% of the honest nodes were exist in the network, integrity was maintained for the non-dependency transaction. The efficiency was high and there was no trace of the block generator. However, this method has problem when a dynamically generated arbitrary node is accidentally attacked. Future research, in order to improve the proposed algorithm, is required for collaborating nodes according to the result of voting. In addition, the above-mentioned non-dependency transaction is also an issue to be solved for the problem of information origin (tagging issue).



Comparative item	Proposed consensus rule	Proof-of-work	Proof-of-stake	Tendermint
Integrity assurance	Blinding vote , High cost to malicious nodes	High cost to malicious nodes, Block validation	Stakeholder, incentive	Stkaeholder, Deprivation of rights
Block generate node	Dynamically Elected decision node (receives a lot of votes)	Nodes win in mining competition	Delegated node	Stakeholder node
Efficiency (Transaction throughput and block generation rate)	High	Low	High	High
Transaction integrity for non dependency transaction (51% of the good nodes exist)	O	X (broad cast tampered block)	X (problem stake)	X (problem stake)
Traceability to block generate node	X	X	O (Traceability to stake holder)	O (Traceability to stake holder)

Table 4. Comparative Blockchain Consensus Rule

Acknowledgments

*This work was partly supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (No.2017-0-00325,blockchain based IoT reliability mangement system development) and (No. 2015-0-00910, National Program for Excellence in SW)

Reference

[1] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system”, Consulted, vol. 1, no. 2012, **(2008)**, pp. 28.
[2] S. Wilkinson, “Storj A Peer-to-Peer Cloud Storage Network”, http://storj.io/storj.pdf.
[3] “Ibm Hyprt ledger white paper”, http://www.the-blockchain.com/docs/Hyperledger%20Whitepaper.pdf.
[4] V. Buterin, “A Next-Generation Smart Contract and Decentralized Application Platform”, https://github.com/ethereum/wiki/wiki/White-Paper.
[5] Rogaway and T. Shrimpton, “Cryptographic hash-function basics: Definitions, implications and separations for preimage resistance, second-preimage resistance and collision resistance”, Springer Berlin Heidelberg, **(2004)**.
[6] “Bit Shares Delegated Proof of Stake”, https://bitshares.org/technology/delegated-proof-of-stakeconsensus/Jae Kwon,Tendermint:ConsensuswithoutMininghttp://tendermint.com/docs/tendermint.pdf.
[7] M. Matsumoto and T. Nishimura, “Mersenne twister: a 623-dimensionally equidistributed uniform pseudorandom number generator”, ACM Trans. Model. Comput. Simul., vol. 8, no. 3, **(1998)**.
[8] J. Barkatullah and T. Hanke, “Goldstrike 1: CoinTerra's First Generation Crypto-currency Mining Processor for Bitcoin”, **(2015)**.
[9] M. Castro and B. Liskov, “Practical Byzantine fault tolerance and proactive recovery”, ACM Trans. Comput. Syst., vol. 20, no. 4, **(2002)**.
[10] S. Meiklejohn, and C. Orlandi, “Privacy-enhancing overlays in bitcoin”, International Conference on Financial Cryptography and Data Security. Springer Berlin Heidelberg, **(2015)**.
[11] J. H. Ziegeldorf, “Coinparty: Secure multi-party mixing of bitcoins”, Proceedings of the 5th ACM Conference on Data and Application Security and Privacy. ACM, **(2015)**.
[12] I. Eyal, “Bitcoin-NG: A scalable blockchain protocol”, 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16), **(2016)**.
[13] K. Croman, “On scaling decentralized blockchains”, Proc. 3rd Workshop on Bitcoin and Blockchain Research, **(2016)**.
[14] J. Herrera-Joancomartí and C. Pérez-Solà, “Privacy in Bitcoin Transactions: New Challenges from Blockchain Scalability Solutions”, Modeling Decisions for Artificial Intelligence. Springer International Publishing, **(2016)**.

Authors

Soo Hwan Cho, he received his Ph.D course in Blockchain and software engineering lab Sogang University. His research includes Blockchain, IoT, Consensus algorithm and Supply chain.

Soo Young Park, he is A Professor in Blockchain and software engineering lab in Sogang University. His research includes lockchain, IoT and Intelligent block chain

Sang Rok Lee, he is an undergraduate in Blockchain and software engineering lab Sogang University. His researche is Adaptive blcockchain system.