Asian Journal of Convergence in Technology *Volume VIII and Issue I*
ISSN NO: 2350-1146 I.F-5.11

A Random and Scalable Blockchain Consensus Mechanism

Davut Çulha,
ASELSAN

***Abstract*— The central piece of blockchain technologies is the consensus algorithm. The consensus is reached via consensus algorithms in the distributed network of the blockchain. The consensus becomes stronger if nearly all the nodes in the blockchain network take part in building the blockchain. Even if some nodes misbehave or malfunction, the consensus should not break down then. Since blockchain networks are distributed, even if some regions of the network or some nodes of the network leave, the blockchain network should not malfunction and it should be consistent. Therefore, the consistency of the blockchain transactions should be assured by all the nodes or nearly all the nodes, or most of the nodes. In other words, the consensus should be the issue of all the nodes. In this work, a novel consensus algorithm is presented to diffuse the mission of building the blockchain to all the nodes. In other words, the consensus should be the issue of all the nodes. In this work, a novel consensus algorithm is presented to diffuse the mission of building the blockchain to all the nodes. The algorithm increases the randomness of nodes and enforces the blockchain network to be more decentralized. Randomness is realized by employing the power of cryptography, especially by using public keys as a characteristic for miners, which are also called signers. Moreover, since the algorithm is realized with a few operations, it contributes to the scalability of the blockchain. Furthermore, digital signatures improve the security level and consistency of the blockchain.**

***Keywords*— blockchain; consensus algorithm; decentralization; randomnes;, scalable; public key; hash value; digital signature; cryptography**

I. Introduction

Blockchain networks are distributed, and there should be consensus in the blockchain transactions. For the consensus, there are many methods, and which are known as consensus algorithms.

Since blockchain networks are distributed, even if some regions of the network or some nodes of the network leave, the blockchain network should not malfunction and it should be consistent. Therefore, the consistency of the blockchain transactions should be assured by all the nodes or nearly all the nodes, or the majority of all the nodes. In other words, the consensus should be the issue of all the nodes. Naturally, some nodes may not take part in the consensus, worse some may behave against consensus. Even in these conditions, a consensus should be reached. This well-known and tough problem is known as The Byzantine Generals Problem. In short, the consensus of the network is under the responsibility of all the honest nodes. The number of honest nodes should be maximized to maximize the strength of consistency. This is decentralization. This means that there should not be central authorities. If the consistency is controlled by some central authorities, worse by only one authority, this is the centralization problem. If such central authorities misbehave, they can change the transactions, or create new transactions in favor of themselves. In brief, decentralization should be maximized in a blockchain network. In this work, a consensus algorithm is designed mainly for this purpose.

Against centralization, some consensus algorithms are developed. In some of them, the structure of the transactions of the blockchain is redesigned, and some graph-based structure is proposed. In this work, decentralization is empowered by increasing the randomness of the nodes which take part in building the blockchain.

In blockchain technologies, although security is empowered by using cryptography, security is very important for blockchain technologies. Especially, private keys should be kept safe from threats. In this paper, the randomness of consensus is realized by using the public keys of miners as their special characteristics. These unique attributes are used in the mining process, and in the earning of mining rewards. For miners, instead of maintaining keys with different unique attributes, it is easier to sign suitable blocks in the network. In other words, a collaboration of network nodes can complete the mining operation fastest. Fastest completions bring mining rewards fastest.

Pool mining is an indicator of centralization in blockchain systems. The proposed model eliminates pool mining because mining rewards are distributed among signers directly. In other words, solo mining, and collaboration with the blockchain network is the most efficient way to do mining.

A big problem in blockchain technologies is scalability. In other words, the number of transactions processed per second is not sufficient for online businesses usually. Therefore, to increase scalability, new consensus algorithms are proposed day-by-day, and existing algorithms are revised. In the proposed consensus algorithm, a few operations are needed for the consensus, and this will contribute to increasing scalability.

In the next section, the related work is presented. After presenting the methodology, the proposed consensus algorithm comes. The experimental results are presented. Lastly, the conclusion is made.

II. Related Work

Blockchain networks are distributed networks, and in distributed networks, a consensus is reached via special algorithms. These are known as consensus algorithms. In the following, some important consensus algorithms and their decentralization characteristics are examined.

Proof-of-Work (PoW): The PoW algorithm requires some amount of work for consensus. This work is proved using hash functions. Hash functions are one-way functions, which means that its application to an input is easy, and the result is got immediately, however, from the result it is impossible to find the input. This property of hash functions is used to find an output with the desired pattern. The found output guarantees that enough effort has been spent. In Bitcoin, this algorithm is used for mining. A new block is mined nearly in 10 minutes according to the adjusting of the difficulty in the algorithm. Then the miner gets the mining reward for its effort. And the miner propagates the block to other nodes in the blockchain network, and they confirm the correctness of the block easily with the help of the one-way property of hash functions. After confirmation, the other nodes add the block to their local blockchain with confidence. The PoW algorithm provides consensus in distributed nodes. However, it requires a huge computational power, and this power increases exponentially in time. Thus, it wastes such an amount of computational power and becomes very expensive economically. The huge computational power causes miners to form mining pools. The disadvantage of mining pools is that blockchain becomes centralized. The mining pools may use their powers to dictate their own rules to the blockchain network stakeholders and demolish the consistency of the blockchain.

Proof-of-stake (PoS): PoS algorithms are designed to prevent resource wastes of PoW algorithms. In PoS, the new block is mined by a selected miner who has enough stake in terms of cryptocurrencies in the blockchain. The miner selection is done proportional to the stake of the miner. This would cause undesirable centralization on the side of the highest stake owners. To prevent this, a randomization method can be applied. In addition to this, the coin-age method can be applied. In the coin-age method, not only the stakes are used but also the ages of the stakes are considered to select the miner. The selected miner uses its digital signature to prove the ownership of the stake. The basic PoS suffers from the Nothing-at-Stake problem. The selected miner may create multiple blocks to gain more fees, and no one prevents this. PoS may cause the blockchain network centralized because the highest stake owners can direct the blockchain.

Delegated Proof of Stake (DPoS): DPoS is a version of the PoS algorithm. PoS resembles a direct democratic election whereas DPoS resembles a representative democratic election. In DPoS, stakeholders select their delegates for mining operations. Then, these delegates confirm transactions and blocks, and mine new blocks. This representative democracy in consensus algorithm makes the blockchain more centralized. Moreover, the delegates may behave dishonestly in DPoS. To prevent this dishonesty, the cheating delegates can be voted out.

Leased Proof of Stake (LPoS): LPoS is a version of the PoS algorithm. Nodes with low balances increase their chances of mining new blocks by leasing stakes of other nodes with high balances. Naturally, the leased amount of stakes are possessed by original wealth owners during leasing, however, it increases the chance of mining new blocks for the leasers. If the leaser wins to mine a block, the mining reward will be shared among the leaser and the original wealth owners proportionally to their stakes. LPoS consensus method increases the decentralization of blockchain and makes it more secure.

Proof of Elapsed Time (PoET): PoET consensus algorithm is used in IntelLedger which is developed by Intel as a blockchain platform. PoET runs in a Trusted Execution Environment (TEE), such as Software Guard Extensions (SGX) by Intel. Adding new blocks is done by randomly selected nodes. Nodes request wait times from TEE which produces random wait times. The node with minimum wait time is selected as a miner. The TEE also produces the proof for the election, and it is easily verified by other nodes. The randomness and safety of this consensus algorithm rely on specialized hardware such as SGX. This type of device guarantees that their executions cannot be tampered with externally. The main disadvantage of this algorithm is the reliance on such devices, and this conflicts with the philosophy of the decentralized blockchain.

The Practical Byzantine Fault Tolerance (PBFT): PBFT is a Byzantine fault tolerance (BFT) consensus protocol. It is the first practical solution to Byzantine failures. It uses replicated state machines for consensus. In this method, a new block is added to the blockchain if more than two-thirds of all nodes agree on that block. The PBFT can tolerate the malicious activities of one-third of all nodes. In this method, the consensus is reached faster and more economically than the PoW algorithm. Moreover, nodes without coins can add new blocks to the blockchain, unlike the PoS algorithm. PBFT is suitable for permissioned blockchains like Hyperledger Fabric because the tolerance of PBFT to malicious activities is low, and that may prevent reaching consensus. Hyperledger Fabric also uses another variant of PBFT called SIEVE for chaincode execution.

Cross-Fault Tolerance (XFT): XFT is a variant of the BFT protocol. XFT assumes that a powerful adversary cannot easily take the control of the entire network by partitioning it into manageable subnetworks that do not communicate with each other. Therefore, XFT relaxes the BFT approach and simplifies the state machine replication problem. If most replicated systems work synchronously and correctly, a consensus is reached in XFT. XFT increases the decentralization of the blockchain. However, it becomes centralized as the probability of malicious attacks is decreased.

Ripple: Ripple is a consensus algorithm that depends on the Byzantine fault tolerance model. Ripple requires 80% of the agreement for consensus in collectively trusted subnetworks. Each trusted subnetwork is managed by a special node named as server, and the server keeps the list of nodes in its subnetwork called Unique Node List (UNL). If a transaction triggered by a node is agreed by 80% of the nodes in the UNL, then it is added to the distributed ledger. The ledger will be consistent if faulty nodes do not exceed 20% of the nodes. Decreasing the probability of malicious attacks results in centralization in the blockchain.

Delegated Byzantine Fault Tolerance (dBFT): dBFT is like PBFT except for delegation. In PBFT, all nodes take a role in adding a new block to the ledger, whereas in dBFT, some nodes are chosen as delegates and only they take a role in adding new blocks. Therefore, dBFT is more scalable than PBFT, but it loses from the decentralization of the network. dBFT algorithm is used in NEO cryptocurrency.

Federated Byzantine Fault Tolerance (FBFT): FBFT algorithm is a variant of the BFT algorithm. In FBFT, large numbers of participants can reach a consensus easily. Each participant trusts a limited number of other participants. Therefore, there are many different groups of participants in the network. Each group reaches a consensus internally. If there are overlapping transactions in the groups, a global consensus should be required. FBFT is more decentralized than the PBFT algorithm. However, decentralization is decreased with an increase in the security.

Stellar Consensus Protocol (SCP): SCP is designed for micro-finance services. SCP is a variant of a Federated Byzantine Fault Tolerance (FBFT) protocol. In FBFT, each node selects its trusted partners. A group of trusted nodes forms a quorum slice. In SCP, a node can place in more than one quorum slices concurrently. Therefore, quorum slices may overlap and build quorums. A quorum is a group of nodes that is required to reach an agreement. A consensus is reached if a quorum agrees on a statement. SCP has two steps for consensus: nomination and ballot. Firstly, the nomination step is executed. In this step, candidate values are proposed and sent to all the nodes in the quorum for agreement. Each node votes for a single candidate value. At the end, the values which are selected unanimously go to the ballot step in SCP. In the ballot step, federated voting takes place and unanimously selected values are aborted or accepted. Finally, the aborted values are discarded. If the consensus is not reached in the ballot step, a new ballot step is initiated using higher values. SCP has high throughput and low latency, which is suitable for IoT applications; however, its latency should be reduced to milliseconds to be used efficiently in IoT applications. SCP like other BFT algorithms is sensitive to malicious attacks. Therefore, empowering the security of SCP yields centralization.

Proof of Authority (PoA): PoA is designed for permissioned blockchain for high-performance needs. In PoA, there are N trusted nodes, which are called authorities. Each authority has a unique ID, and it is assumed that most of the authorities are honest. In other words, N/2+1 authorities are assumed as honest. The responsibility of block creation is distributed among authorities using a rotation schema. For each authority in PoA, there is a special time slice to create a new block. There are two implementations of PoA: Aura and Clique. PoA is centralized because the authorities are predetermined.

Aura: Aura is an implementation of the PoA algorithm. Aura is implemented in Parity Ethereum clients. In Aura, the network should be synchronous, and all authorities should be synchronized with the same clock. Each authority has a special time slice in a round of mining blocks. In that special time slices, authorities add transactions to the new block and broadcast it to other authorities. If all authorities accept the proposed block, the new block is added to the blockchain. Because of the predetermined authorities, Aura loses decentralization characteristic.

Clique: Clique is an implementation of PoA algorithm. Clique is implemented in Geth, which is a GoLang based Ethereum client. In Clique, block creations are done in epochs. When an epoch, which is a time period, starts, a particular block is broadcasted, which specifies the set of authorities. In Clique, epochs and their related leaders and authorities are calculated using a formula. In each epoch, authorities other than leaders can also propose blocks for creation but blocks proposed by non-leaders are delayed. Therefore, the block proposed by the leader in the current epoch will be probably accepted first for maintaining the blockchain. Clique is centralized because there are a few authorities.

Proof-of-Capacity (PoC): PoC is, which is also known as Proof of Space (PoSpace), is like PoW. PoW depends on computing power whereas PoC depends on computer storage. Miners in PoC increase their chance to mine new blocks by storing immense data sets known as plots. This algorithm was used in Burstcoin firstly. In Burstcoin, the consensus algorithm is implemented in two stages. The first stage is called plotting. In this stage, miners create nonces, which are repeated hashing of data and miner's ID using Shabal hash algorithm. Calculating nonces using the Shabal algorithm is very hard. Therefore, miners calculate nonces in advance and store them in the hard disks. For each block creation in the network, a puzzle is started. The winner is the miner, who has the closest nonce. As miners increase the space used for plotting, they get more nonces and increase their chance to obtain rewards. PoC is suitable for mining pools. Therefore, it becomes centralized.

Proof-of-Burn (PoB): In PoB, miners send some coins to a verifiable and unspendable address, which is called the burning of coins. The used addresses are called eater addresses, and the coins sent there are unrecoverable and cannot be spent again. Those addresses are generated randomly and there is no associated private key for them. Miners win to mine new blocks proportional to the amounts of coins they burned. Therefore, malicious miners will not be in the mining of blocks because they will not want to spend coins. This method is used in Slimcoin cryptocurrency. The more the miners burn coins, the more they have chances to win the mining rewards. In PoB, miners will not earn more even if they are joined. Therefore, PoB supports decentralization. PoB is not suitable for IoT applications because it does not have a monetary framework. PoB is centralized because the mining ability of nodes increases with wealth.

Proof of Importance (PoI): PoI can be accepted as a variant of PoS. In PoS, miners have more chances to add new blocks if they hold more coins in their accounts. Similarly, in PoI, the chance of adding a new block to the blockchain is proportional to the importance of the miners. PoI depends on the importance of nodes in the network. The importance of a node is determined according to the productive activities of that node in addition to its account balance. For example, the number of transactions that occurred to or from that node can show the importance of the node. Miners increase their chance to mine new blocks proportional to their importance. PoI is firstly used in NEM cryptocurrency. PoI improves decentralization of the blockchain network, and it is resistant to Sybil-style attacks. PoI has high throughput and comparatively low latency. Therefore, it is suitable for IoT applications however it depends on a monetary framework, which is not compatible with IoT applications. PoI is suitable for mining pools. Therefore, it tends to be centralized.

Proof of Activity (PoAc): PoAc is a combination of PoW and PoS algorithms. First, a hash problem is solved by miners as in the PoW algorithm. The hash solution is related only to the header of the block not to the transactions. Transactions are added to the solved block header later. After adding transactions to the block, some validators sign the block to reach a consensus. The last part of PoAc is done using the PoS algorithm. Since PoAc is a combination of PoW and PoS, it is more secure against practical attacks. In other words, 51% of attacks drop to nearly zero in PoAc, because that kind of attack requires 51% of mining power and 51% of all coins at the same time. The cryptocurrencies Decred and Espers use the PoAc algorithm in their blockchain. Since PoAc has high latency, it is not suitable for IoT applications. PoW could lead to centralization of the blockchain network, while PoAc tends to decentralize because centralized mining power should be reduced to obtain rewards proportional to miner stakes. PoW and PoS can be considered as centralized. Therefore, PoAc can be considered as centralized.

Casper: Casper is a PoS consensus algorithm for transferring the consensus algorithm of Ethereum from PoW to PoS. Casper punishes malicious validators by decreasing their stakes to solve the problem of Nothing at Stake. Moreover, a chain selection rule like the longest chain in Bitcoin blockchain is applied for the forks. The selection rule is the Greedy Heaviest Observed Subtree (GHOST). In GHOST, the fork with the heaviest subtree among the forks is selected as the main chain. Casper is a PoS algorithm. Therefore, it tends to become centralized.

Tendermint: Tendermint is a combination of PBFT and PoS algorithms for permissioned blockchain. In PBFT, the voting power of each node is the same, whereas in Tendermint voting powers of the nodes are proportional to their stakes. In Tendermint, the voting process has two steps: pre-vote and pre-commit. If two-thirds of validators pre-vote for the block, the block goes to the pre-commit step. If two-thirds pre-commit the block, the block is added to the blockchain. In Tendermint, validators should lock their coins to vote. If they do malicious activities, they are punished. Tendermint has high throughput and low latency however it is not suitable for IoT applications because it depends on a monetary framework. Since Tendermint depends on PBFT and PoS algorithms, it is a candidate for centralization.

Paxos: Paxos is the first consensus algorithm. The nodes in Paxos are categorized as proposers, acceptors, and learners. In Paxos, a single value is selected from one or more values by the acceptors. If the value is accepted by most of the acceptors, it means that the consensus is reached, and the value is broadcasted to all learners. Paxos is based on voting system like PBFT. Therefore, it may become centralized.

Raft: Raft algorithm is proposed after the Paxos algorithm. Raft is easier to understand and easier to implement than Paxos. Raft is a voting-type algorithm and uses log synchronization for data consistency. Nodes have different roles as leaders, candidates, and followers. All nodes can be candidates. If a candidate gets more than half of the votes, it becomes the leader. At this time, a log is copied to the followers. Each node checks the health of the leader whether it is alive or not. If it is lost, a new leader election is performed. In Raft, there are terms, and, in each term, there is only one leader. During that term, the leader manages all the requests of the clients. Raft can be considered as centralized because it uses a voting mechanism.

Proof-of-Trust (PoT): PoT has four phases for block creation. These are leader election, determining validators, voting, and chaining. In the leader election phase, a leader is selected. In the second phase, the leader determines the validators. In the third phase, the validators vote for the transactions. In the last phase, validated transactions are chained to the blockchain. PoT is scalable and resistant to Sybil-type attacks. Mining pools can control PoT. Therefore, it becomes centralized.

Proof of Weight (PoWeight): PoWeight is a type of PoS algorithm. In PoWeight, each node is assigned a weight. The weights are calculated according to different factors in addition to balances. Filecoin cryptocurrency uses PoWeight, and the weights are calculated according to the amount of IPFS data. There may be an incentivization problem because there is no reward for transaction confirmation. PoWeight is suitable for mining pools. Therefore, it becomes centralized.

Blockchain networks are distributed and should be decentralized. Decentralization means that there should not be central authorities in the distributed network to reach a consensus. In [1], an assessment on decentralization is made. Most of the nodes should be able to take a role in the decisions of the network, namely in the validation of transactions. Moreover, they should be able to take a role to record and update those transactions distributedly. Although all nodes in the network do not need to trust each other, they should reach a consensus in this trustless environment. For this reason, there should be no single-point-of-failure in the distributed network.

In blockchain technologies, centralization is a crucial problem, and it is contradictory to the soul of blockchain technologies. The root cause of centralization is mainly from mining pools. In [2], the increasing power of mining pools and centralization problems are addressed. In bitcoin network, more than 99% of the hash power is produced by mining pools [3]. Worse, the three biggest mining pools have more than 50% of the total hash power [4]. Therefore, instead of decentralization, because of these monopolies, blockchain networks become centralized. Centralization brings security problems as well.

To decentralize the blockchain networks, some consensus algorithms are designed. In [3], against the centralization problem, a graph-based consensus mechanism is proposed. Instead of mining blocks, the transactions are mined using a proof-of-work mechanism, and a transaction verifies two parent transactions and gets mining fees from parent transactions that had posted fees before for verification. Therefore, the newly mined transactions increase confirmations of the parent transactions, and its graph-like structure increases the scalability. This transaction-based mining supports solo miners instead of mining pools and empowers decentralization.

The randomness of nodes can be provided to achieve decentralization. The proposed algorithm uses the randomness of cryptography to incorporate all nodes in the building of blockchains. Security is also an important issue in blockchain networks. Security in blockchain means that the transactions should not be changed by unauthorized ones, and additionally, new transactions can be added by authorized ones. The private keys showing the authorizations should be kept secretly. There should not be external threats, or the blockchain should be safe from those threats.

Nowadays, scalability may be the biggest problem in blockchain technology. With the term of scalability, the performance scalability of a blockchain network is meant. In other words, it means the number of transactions that can be processed in the blockchain per second. In [5], near-linear scalability is reached by secure sharding of transactions. In [6], a highly scalable consensus algorithm is introduced based on sharding. In [7], various solutions to scalability are analyzed. In [8], the inherent conflicts between scalability and decentralization, and key challenges in blockchain decentralizations are identified.

In [9], blockchain consensus algorithms are compared based on scalability, method of the algorithm, and security risks. In [10], for scalability and centralization problems, a new consensus algorithm is proposed based on the BFT algorithm. Advanced cryptographic techniques are used to enhance the BFT algorithm.

Variations of BFT algorithms are used mainly to increase the transaction throughput of blockchain. In [11], a fast and scalable consensus algorithm based on BFT and hardware-assisted secret sharing is proposed. Similarly, in [12], hardware-based security is used to build an efficient consensus algorithm named Proof of Luck. In [13], a consensus algorithm based on the Swirlds hashgraph data structure is presented. It is fast and Byzantine fault-tolerant. In [14], a consensus algorithm based on the PBFT algorithm is presented for scalability, and strong consistency is achieved via collective signing. In [15], a fast and practical consensus algorithm based on BFT is implemented in an asynchronous structure to achieve high throughput. In [16], a scalable Byzantine consensus protocol is designed to add several blocks by building random secure committees. In [17], a BFT consensus algorithm is presented to order transactions in a determined way among nodes to reach enough scalability.

The scalability of blockchain networks is increased using additional chains to the main blockchain. In [19], different consensus protocols run in satellite chains in parallel to increase scalability. In [20], a graph-based consensus algorithm is presented, which uses a dual-blockchain to increase confirmation efficiency.

Another approach to the scalability problem of blockchain is a functional deconstruction of a building new block. In [18], a fast consensus algorithm based on the PoW algorithm is implemented by deconstructing the blockchain into its basic functionalities.

III. Methodology

Blockchain provides trust and consensus to desired systems. To efficiently implement trust and consensus in blockchain systems, decentralization of blockchain systems should be ensured and perpetuated. In this paper, a novel solution is proposed, which depends on the randomness of public keys, for decentralization. The steps of the applied methodology are the following:

- Maximizing decentralization of blockchain systems is examined
- Unique attributes that can be used for decentralization are determined
- A blockchain model is designed with the unique attributes
- The blockchain model is proved using simulations

The most important problems in blockchain systems are related to security, scalability, and decentralization. At the heart of the blockchain systems, there are consensus algorithms. Consensus algorithms are adjusted to optimize properties in security, scalability, and decentralization of the blockchain systems.

PoW favors security but it is not compatible with scalability and decentralization. On the other hand, PoS is a more scalable and decentralized algorithm, but it loses from security. The combination of these algorithms is PoAc, which tries to optimize all these properties.

In PBFT, decentralization is supported because all the nodes vote for maintaining the blockchain. However, it is less scalable because all the nodes are included in the chaining process. Therefore, other approaches are applied to increase scalability. In dBFT, the voters are limited to a group of delegates.

In this work, decentralization is taken as the most important issue for blockchain systems. Without decentralization, the meaning of blockchain will be lost.

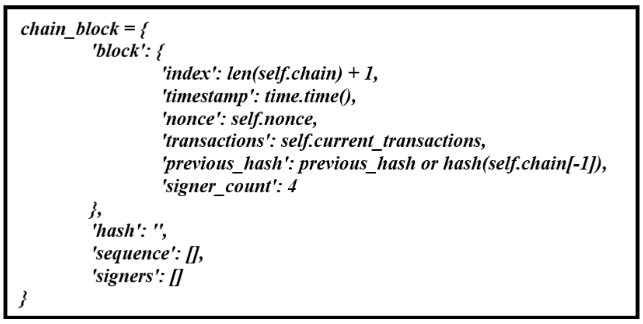
Decentralization is improved if the number of nodes in the chaining process is increased. If distinct properties of the nodes are included in the consensus mechanism, it will support decentralization because it will disfavor the joining of nodes.

Distinct features of the nodes may be their IDs, their hash values according to some formulas, their private or public keys. Public keys can be used to differentiate nodes. Public keys, hash values, and signatures are random values because no one can predict their values. Therefore, hash values of public keys produce random values. This randomness property can be used to support decentralization.

In this work, the randomness property of hash values of public keys is considered to find a solution to the decentralization problem of blockchain systems. After finding the solution, simulations are performed to prove the correctness of the solution.

IV. The Proposed Consensus Mechanism

The proposed algorithm works on a usual blockchain. In other words, blocks are chained to the previous blocks with the previous block header hashes until to the genesis block. Block headers and transactions are kept separately. The structure of the blockchain is shown in Figure 1 as a code snippet.



```
chain_block = {
        'block': {
                'index': len(self.chain) + 1,
                'timestamp': time.time(),
                'nonce': self.nonce,
                'transactions': self.current_transactions,
                'previous_hash': previous_hash or hash(self.chain[-1]),
                'signer_count': 4
        },
        'hash': '',
        'sequence': [],
        'signers': []
}
```

Fig.1. Structure of a block header

In the block header, four elements are defined as “block”, “hash”, “sequence”, and “signers”. The first element is a structure named “block”. In the structure, there are ordinary block header elements. The element “index” is used to keep the order of the block in the blockchain. The element “timestamp” is used to keep the creation time of the block. The element “nonce” is added to change the hash of the block header. The element “transactions” keeps the pointer to the transactions of the block. The element “previous_hash” is used to point to the previous block header. The element “signer_count” is special to this proposed algorithm. It shows the number of signers who should sign the block header. Here, this value is set to 4. It means that each block header should be signed by 4 signers. The signer count can be incremented to contribute to the strength of security. The second element of the block header is named “hash”, and it keeps the hash value of the ordinary block header. The third element is named as “sequence”, which keeps public keys and hash values of the combination of previous hash values and public keys. The last element of the block header is named “signers”, and it keeps the signatures of the signers who signed the block header.

```
Build a valid block with valid transactions;
Take the initial hash value of the block header;
Take the hash value of the combination of the initial hash value and the public key of the first signer;
If the hash value ends with bits “00”;
    Sign the last hash value with the private key of the first signer;
    Calculate the next hash value by adding the public key of the signer;
    If the hash value ends with bits “01”;
        Sign the last hash value with the private key of the second signer;
        Calculate the next hash value by adding the public key of the signer;
        If the hash value ends with bits “10”;
            Sign the last hash value with the private key of the third signer;
            Calculate the next hash value by adding the public key of the signer;
            If the hash value ends with bits “11”;
                Sign the last hash value with the private key of the fourth signer;
                Calculate the hash value of the updated block header;
                Add the updated block header to the blockchain;
```

Fig.2. Adding a valid block to the blockchain

In Figure 2, the pseudocode of adding a valid block to the blockchain is depicted. The pseudocode shows that for each block, ordered hash values are expected. Assume that each node in the blockchain network has only one private key. And a node prepares a valid block with valid transactions and propagates it to the blockchain network. Each node in the network tries to sign the block. Only one-fourth of them will be able to sign the block because the end pattern of the resulting hash value should be compatible with the consecutive signing order. Only one-fourth of the nodes will have the ability to sign because the algorithm is designed for 4 signers. In other words, 4 trials are needed for a signing step. Therefore, this mechanism randomizes the signing node. For each block, this operation will be repeated 4 times. At each time, other random nodes will be able to sign the block. Briefly, to add a valid block to the blockchain, 3 random nodes should sign the block after the block creation.

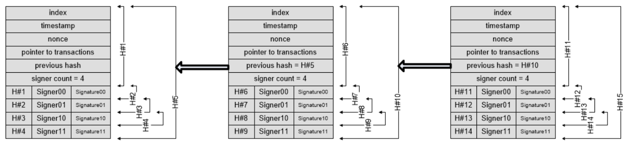


Fig.3. The formation of the blockchain

In Figure 3, the structure of the blockchain is shown. First, the basic elements of the block header are formed. The block header should be valid with valid transactions. Then, its hash value is the first hash value and used by the 4 signers consecutively. Each signer will be able to sign the block if its public key is compatible with the block header hash. The hash algorithm SHA256 is used to get the new hash values. The following formula shows the calculation of the next hash values:

$$HN+1 = SHA256(HN\ SK) \qquad (1)$$

where

HN : The previous hash value,

HN+1 : The next hash value,

SK : The public key of the Kth signer.

Each signer calculates the next hash values by providing its public key to Formula 1. The resulting hash value should be compatible with the order of the signer. If it is compatible, it signs the resulting hash value and adds to the block header. If it is not compatible, a suitable signer is needed to sign the block header hash value. In this blockchain, 4 signers are needed to complete the mining process of the blocks. The public keys of the signers are named as “Signer00”, “Signer01”, “Signer10”, and “Signer11” according to the binary representation of their order numbers. In each signing process, the resulting hash values should end with the binary representation of the signer orders. Therefore, for a signer to sign a block, its public key should produce a compatible resulting hash value. In other words, to take role in the signing process, a characteristic of the signer is considered. Each signer has exactly the probability of ¼ to contribute to the signing process. After completing the signing process, the block is added to the blockchain. This block is easily verified by all the nodes because it has valid transactions, it points to the previous block, its header is signed by 4 proper random signers, and their signatures are verifiable with their public keys. After successfully adding the block to the blockchain, the nodes are prepared to add a new valid block to the blockchain. In this case, the previous hash is the hash of the updated block header of the last block in the blockchain lastly signed by the 4th random signer.

The incentive mechanism of the proposed model depends on the collaboration of the nodes. Nodes are signers, and they are also miners. They earn mining rewards. First, a signer builds a block of transactions and completes the first signing operation. Then, it propagates the signed block to the network. All signers listen to the network for the signed blocks. If a signed block comes, they try to complete the consecutive signing operation. With or without signing, they also propagate the signed block to the network. By this way, all signing operations are completed. The last signer also has a special task for the signed block. After completing the signing operation, it adds reward transaction to the signed block. The reward transaction has the account addresses of the signers with the equally divided reward amount among the signers. The account addresses are derived from the public keys of the signers which are included in the block. Moreover, the last signer sign the complete block to preserve the consistency of the block. Then, the last signer adds the block to its local blockchain and also propagates the block to the network. All signers listen to the network, and they accept blocks. If they are valid, they add them to their local blockchains and also propagate them to the network.

The first signers that are also block creators will not try to complete all signing operations because each additional step in the signing process takes as much as the signer count trials. For the 4-signer case, it is 4 operations. For the remaining 3 signing steps, 64 trials are needed. This number also increases exponentially with an increase in the signer count. The following formula shows the number of trials needed to complete signing steps after block creation:

$$T = c^{c-1} \qquad (2)$$

where

T : The total number of trials,

c : The signer count

The remaining signing steps can be completed in 3 operations in the network if there are enough number of different public keys for the 4-signer case. On the other hand, to complete all the remaining signing steps may take up to the number of trials in Formula 2. Therefore, each node in the network would prefer to do mining in collaboration with the network. In other words, they want to be decentralized.

In the incentive mechanism, the reward account addresses are not changeable. They are dependent to the signers of the block. Therefore, mining pools cannot be formed because mining rewards go directly to the relevant signers.

In the block header, there is a nonce element. This element is added to the header because it can be used if it takes a very long time to add a new valid block to the blockchain. At that time, the nonce can be incremented, and the signing process begins from scratch. This time, the signing process probably will take the usual signing time.

The proposed consensus algorithm is designed with 4 signers. However, the algorithm is flexible to any positive number of signers like 5 or 8. For example, if the signer count is 5, the first signer should produce a hash value ending with “000”. Respectively, the other signers should produce hash values ending with “001”, “010”, “011”, and “100”. In this algorithm, the comparison is done in bits. In other words, binary numbers up to signer count are used for ending patterns. Therefore, if the power of 2 is used, the comparisons are done with all the values of the power of 2. For 8 signers, the power of 2 is 3 exactly, then the signers will produce hash values ending with “000”, “001”, “010”, “011”, “100”, “101”, “110”, and “111”.

```
def try_proof(self, candidateBlock):
    if candidateBlock == None:
        return None
    block = candidateBlock['block']
    if block == None:
        return None
    signer_count = block['signer_count']
    signers = candidateBlock['signers']
    signer_position = len(signers)
    signer_bits_length = math.log2(signer_count)
    signer_bits_length = math.ceil(signer_bits_length)
    sequence = candidateBlock['sequence']
    consistency = sequence[-1] + self.public_key_hex
    consistency_hash = hashlib.sha256(consistency.encode()).digest()
    consistency_hash_hex = consistency_hash.hex()
    signature = self.privateKey.sign(consistency_hash)
    signature_bits = bitstring.Bits(consistency_hash)[-1*signer_bits_length:]
    compare_bits = bitstring.Bits(uint=signer_position, length=signer_bits_length)
    if compare_bits == signature_bits:
        signer = {
            'public_key': self.public_key_hex,
            'signature': signature.hex()
        }
        proof_block = {
            'block': block,
            'hash': consistency_hash_hex,
            'sequence': sequence.copy(),
            'signers': signers.copy()
        }
        proof_block['signers'].append(signer)
        proof_block['sequence'].append(self.public_key_hex)
        proof_block['sequence'].append(consistency_hash_hex)
        return proof_block
    else:
        return None
```

Fig. 4. A flexible number of signers.

In Figure 4, the code snippet for a flexible number of signers is given. The variable “signer_position” is the order of the signer. The count of signers is kept in the variable “signer_count”. The logarithm of the count of signers is taken according to base 2, and it is ceiled to the nearest big integer to find the length of the comparison bits. The variable “signer_position” in bits is compared with the last bits of the hash value according to the found length of comparison bits. If they are equal, it means that the signer produces the corresponding pattern, and its hash value is valid.

V. Results And Discussion

The proposed algorithm is implemented with a python program, and the program is executed for 100 miners. Each miner is represented as a thread in the program. The program is executed until 10000 blocks are mined in the blockchain. In Table 1, the mining counts of the 100 miners are shown. Mining count means the number of blocks mined. In the first row, the counts of the Miners between 0 and 9 are displayed. In the second row, the related counts of Miners 10 to 19 are displayed. In other words, each row displays the mining counts of 10 miners up to 100 miners.

TABLE I. Mining Counts Of 100 Miners

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Miner 0+	123	115	103	108	123	88	85	90	103	120
Miner 10+	98	123	98	85	115	108	90	90	90	88
Miner 20+	113	118	75	63	93	133	70	105	137	100
Miner 30+	65	75	113	116	65	93	98	105	98	115
Miner 40+	126	93	74	120	128	88	61	95	100	90
Miner 50+	98	75	85	135	90	110	88	115	132	105
Miner 60+	80	135	105	134	105	100	130	85	80	108
Miner 70+	138	115	133	90	105	113	82	100	113	103
Miner 80+	109	69	90	103	75	80	118	118	90	93
Miner 90+	113	107	81	78	75	85	75	70	113	103

The mining counts show that each miner mined at least 61 blocks. The maximum mining count is 138, which is done by Miner 70. The standard deviation of the counts is 19.0. These counts and the standard deviation show that each miner mined around the average 100 blocks. Therefore, all the nodes in the blockchain network sufficiently took part in building the blockchain. Namely, the randomness of nodes is accomplished, and decentralization is realized.

TABLE II. MINING COUNTS OF CENTRALIZED 100 MINERS

	0	1	2	3	4	5	6	7	8	9
Miner 0+	52	36	53	40	44	47	56	56	46	40
Miner 10+	53	47	54	49	47	48	40	52	36	41
Miner 20+	57	60	44	48	53	46	46	50	51	47
Miner 30+	46	49	52	44	41	50	59	35	48	50
Miner 40+	68	48	50	46	42	41	52	59	44	46
Miner 50+	172	150	145	177	135	138	145	157	144	157
Miner 60+	156	156	144	159	172	169	153	139	170	135
Miner 70+	153	138	186	146	144	153	128	143	166	172
Miner 80+	157	130	147	134	161	154	150	149	157	150
Miner 90+	171	155	156	141	163	144	135	140	154	141

In Table 2, a centralized version is executed again for 10000 blocks. In this case, the first half of the miners are taken as a group to show the effects of the centralization. In other words, half of the miners form a pool, which is simulated by assigning the same private key to the first half of the miners. The performance of the first half of the miners decreased in the execution. The total number of blocks added by the pool is 2409. The remaining 7591 blocks are added by the solo miners. Therefore, the proposed consensus algorithm supports the decentralization of blockchain.

Decentralization is the most important property among the major properties like security and scalability in blockchain systems. If the blockchain system goes away from decentralization, the blockchain can probably be controlled by central authorities. In short, decentralization should be empowered in the consensus algorithms.

Private keys should be generated randomly to improve their security. However, there is an alternative to select directly. Therefore, it is not random. On the other hand, the public keys are derived from private keys, which are completely random. Similarly, the hash values of some data are completely random. In addition to them, digital signatures are also completely random values. In this work, the complete randomness property of hash values is used in the proposed consensus algorithm. The nodes with different private keys will have more chances in the mining process and will help the decentralization of the blockchain.

The proposed consensus algorithm tries to maximize decentralization. On the other hand, the well-known consensus algorithms are variant of PoW, PoS, and BFT algorithms. PoW-like algorithms are suitable for pool mining. Therefore, they tend to become centralized. PoS-like algorithms depend on the proportion of wealth. Therefore, they tend to become centralized. BFT-like algorithms depend on voting mechanisms. In order to empower them against attacks, they tend to become centralized. In brief, other algorithms converge to centralization in one of three dimensions. On the other hand, the proposed consensus algorithm tends to converge decentralization with an increase in the signer count.

VI. CONCLUSION

Blockchain networks suffer from centralization problems. Centralization is contradictory to the spirit of blockchain technologies. This centralization problem weakens the blockchain and brings security problems as well. Therefore, to strengthen the blockchain and make it more secure, decentralization should be maximized. In this work, decentralization is achieved using the randomness power of hash values.

In blockchain networks, a consensus is reached via consensus algorithms. In this work, decentralization is attained by implementing a consensus algorithm.

The proposed novel consensus algorithm depends on a random selection of mining nodes. The randomization is realized by signing the block header with ordered hash values. For mining a new block, four random signers are needed to obtain the desired ordered hash values. In this work, four is taken as the number of signers. However, this algorithm can be executed with a different number of signers. If the number of signers is increased, it will increase the security of the blockchain.

The proposed model tries to maximize decentralization because solo mining is more efficient with the collaboration of other solo miners in the blockchain network than solo mining without collaboration or pool mining. Solo mining without collaboration requires many operations, and pool mining is restricted by distributing mining rewards to the solo miners directly.

The proposed algorithm needs a few operations for mining a block. Therefore, it is a scalable consensus algorithm. Moreover, since this algorithm uses digital signatures, the consistency of blockchain is increased.

Public keys that are unique and random are used as unique properties of miners for decentralization. In brief, the proposed novel consensus algorithm is a scalable consensus algorithm that increases the decentralization of blockchain networks with an ordered list of random hash values.

References

[1] Gencer, A. E., Basu, S., Eyal, I., van Renesse, R., & Sirer, E. G. (2018). Decentralization in bitcoin and ethereum networks. arXiv preprint arXiv:1801.03998.

[2] Sheehan, D., Gleasure, R., Feller, J., O'Reilly, P., Li, S., & Cristiforo, J. (2017, August). Does Miner Pooling Impact Bitcoin's Ability to Stay Decentralized?. In Proceedings of the 13th International Symposium on Open Collaboration (p. 25). ACM.

[3] Boyen, X., Carr, C., & Haines, T. (2016). Blockchain-free cryptocurrencies: A framework for truly decentralised fast transactions. Cryptology ePrint Archive, Report 2016/871.

[4] Bitcoinchain. (2019), https://bitcoinchain.com/pools

[5] Luu, L., Narayanan, V., Zheng, C., Baweja, K., Gilbert, S., & Saxena, P. (2016, October). A secure sharding protocol for open blockchains. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (pp. 17-30). ACM.

[6] Zamani, M., Movahedi, M., & Raykova, M. (2018, October). RapidChain: scaling blockchain via full sharding. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (pp. 931-948). ACM.

[7] Chauhan, A., Malviya, O. P., Verma, M., & Mor, T. S. (2018, July). Blockchain and scalability. In 2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C) (pp. 122-128). IEEE.

[8] Chu, S., & Wang, S. (2018). The Curses of Blockchain Decentralization. arXiv preprint arXiv:1810.02937.

[9] Bach, L. M., Mihaljevic, B., & Zagar, M. (2018, May). Comparative analysis of blockchain consensus algorithms. In 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO) (pp. 1545-1550). IEEE.

[10] Gueta, G. G., Abraham, I., Grossman, S., Malkhi, D., Pinkas, B., Reiter, M. K., and Tomescu, A. (2018). SBFT: a Scalable Decentralized Trust Infrastructure for Blockchains. arXiv preprint arXiv:1804.01626.

[11] Liu, J., Li, W., Karame, G., & Asokan, N. (2018). Scalable byzantine consensus via hardware-assisted secret sharing. IEEE Transactions on Computers.

[12] Milutinovic, M., He, W., Wu, H., & Kanwal, M. (2016, December). Proof of luck: An efficient blockchain consensus protocol. In Proceedings of the 1st Workshop on System Software for Trusted Execution (p. 2). ACM.

[13] Baird, L. (2016). The swirlds hashgraph consensus algorithm: Fair, fast, byzantine fault tolerance. Swirlds, Inc. Technical Report SWIRLDS-TR-2016, 1.

[14] Kogias, E. K., Jovanovic, P., Gailly, N., Khoffi, I., Gasser, L., & Ford, B. (2016, August). Enhancing bitcoin security and performance with strong consistency via collective signing. In 25th USENIX Security Symposium (USENIX Security 16) (pp. 279-296).

[15] Miller, A., Xia, Y., Croman, K., Shi, E., & Song, D. (2016, October). The honey badger of BFT protocols. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (pp. 31-42). ACM.

[16] Luu, L., Narayanan, V., Baweja, K., Zheng, C., Gilbert, S., & Saxena, P. (2015). SCP: A Computationally-Scalable Byzantine Consensus Protocol for Blockchains. IACR Cryptology ePrint Archive, 2015, 1168.

[17] Asayag, A., Cohen, G., Grayevsky, I., Leshkowitz, M., Rottenstreich, O., Tamari, R., & Yakira, D. (2018). Helix: a scalable and fair consensus algorithm (pp. 2-1). Technical report, Orbs Research.

[18] Bagaria, V., Kannan, S., Tse, D., Fanti, G., & Viswanath, P. (2018). Deconstructing the Blockchain to Approach Physical Limits. arXiv preprint arXiv:1810.08092.

[19] Li, W., Sforzin, A., Fedorov, S., & Karame, G. O. (2017, April). Towards scalable and private industrial blockchains. In Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts (pp. 9-14). ACM.

[20] Lee, J. (2018). The Chain of Antichains, Box Protocol: the Dual-Blockchain and a Stablecoin. arXiv preprint arXiv:1810.11871.