The CSI Journal on
Computer Science and Engineering
Vol. 19, No. 2, 2025
Pages 33-38
doi.org/10.22034/jcse.2025.225914
Regular Paper

Analysis of incentive mechanism in Repchain

Mojdeh Hemati[1] Mehdi Shajari[1]

[1] Department of Computer Engineering Amirkabir University of Technology

Abstract

In recent years, the blockchain that is the basis of Bitcoin has received much attention. However, the blockchain also faces many challenges, such as security and scalability, which have been the subject of recent researches. Much work has been done to solve the scalability problem in blockchain; one of these methods is sharding. This method is based on dividing the network into different groups and validating transactions in parallel. These methods use traditional consensus algorithms. One of the problems in this regard is the incentive that should be provided for nodes to participate in these consensus algorithms. In this paper, Repchain, one of the existing methods in this field, is examined, and the problems that this method has is analyzed. Next, it is proved that the proposed method causes the network nodes not to follow the protocol and also causes collusion between network nodes.

Keywords: blockchain, bitcoin, consensus algorithm, sharing, incentive.

1. Introduction

The blockchain that was introduced by Bitcoin [1] is a peer-to-peer and distributed technology that does not require any third party to manage. Blockchain is a distributed database or a public ledger in which all transactions and data are recorded. All data in the blockchain is stored in blocks. The length of this chain increases over time, and network nodes must update the version they maintain of this public ledger over time. Any information recorded in this database cannot be changed or deleted in any way. Asymmetric cryptography and consensus algorithms are used to maintain this public ledger's security and compatibility [2].

Blockchain technology generally has the characteristics of decentralization, stability and compatibility, anonymity and reliability; also, it has a significant role in reducing costs and increasing efficiency. [3]

Consensus algorithms are one of the primary and important components in blockchain distributed networks. These algorithms are the backbone of distributed networks that provide a proper implementation for cryptocurrency systems and other applications. The network nodes can achieve an agreement in an environment with no trust and trusted third party by these algorithms. [4]

In Bitcoin, all network nodes must agree on the block's validity for adding a block to the blockchain. For this goal, a consensus algorithm called proof of work is used. In the proof-of-work method, network nodes should solve a complicated problem that needs much effort, and finally, the solution to this problem is used for creating a block. Proof of work was a successful solution used in Satoshi Nakamoto's paper in 2008, but it also has many weaknesses that many studies have been done to solve them. However, these issues are not entirely solved. The problems of the proof-of-work algorithm are as the following: [4]:

1. Energy consumption
2. Scalability

Section 2 explains each of these problems and mentions the solutions proposed to solve each of these problems. Section 3 talks about the Sharding method, which is one of the scalability methods. This section also talks about the incentive mechanism in this solution. In this section we explain Repchain, a sharding method that has tried to establish the correct incentive mechanism. Section 4 analyzes the Repchain incentive mechanism and shows that this mechanism has some drawbacks. Section 5 talks about other sharding solutions that have similar drawback to Repchain. Section 6

talks about future work, and finally, section 7 concludes the paper.

2. Weaknesses of proof of work

2.1. Energy consumption

One of the significant problems with the proof of work algorithm is that it requires a lot of energy for solving a hash problem that has no use and application in the real world. The only use of this energy consumption is to prove that a certain amount of work has been done. Therefore, it is said that the proof-of-work algorithm wastes much energy that has no real application and wastes a large number of sources such as electrical energy.

Much research has been done to solve this problem, in all of which attempts have been made to replace an unapplied mathematical problem (hash function) with a useful alternative problem. A problem that has the feature of hash function such as difficulty and the result of solving this problem has an application in the real world. Therefore, the energy that needs to prove a certain amount of work is spent on solving a useful problem.

The first attempts to prove useful works were two protocols known as Primecoin [5] and Permacoin [6]. In Primecoin, an attempt is made to find a series of specific prime numbers which are called Merssene number that has many applications in various fields. In Permacoin, an attempt is made to expend energy on storing huge files in a distributed environment.

Another work done in this direction is proving useful work in which a series of applied mathematical problems such as orthogonal vectors have been tried to be used instead of using the hash function [7].

2.2. Scalability

Another problem with the proof-of-work algorithm is scalability. Scalability includes both the space required to store transactions and the speed of transactions. In Bitcoin, transactions are generated at a rate of 7 transactions per second, which is a very small amount; we need several thousand transactions per second. In addition, a large volume of transactions is generated daily and added to the blockchain. This causes the blockchain storage to increase significantly on a day.

There are several solutions to the problem of scalability in the blockchain, which can be divided into the following two categories [8]:

1. First layer solutions

These solutions require changing blockchain properties such as block size, block structure and consensus algorithm. Sharding techniques [8] and Segwit [9] are two solutions in this category. In the following, we will examine the sharding more.

2. Second layer solutions:

These solutions try to increase the blockchain's scalability by using of some off-chain methods, such as creating a network channel separate from the blockchain network to reduce the overhead that there is on the main blockchain. Solutions such as plasma [10], lightning [11], Raiden [12] are among these category's solutions.

3. Sharding methods and its challenges

The main idea of these methods is to divide all the network computing power into groups called committees, and to divide all network transactions into separate groups and give each group of transactions to a committee for processing. In this way, the transactions will be processed in parallel. Finally, a final block of transactions approved by various committees is created and distributed on the network. The general method of sharding can be seen in Figure 1.

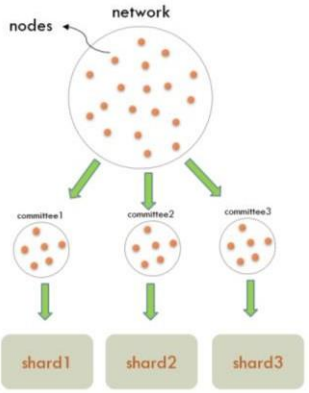

Fig. 1. The sharding method

In these methods, the main purpose is to validate transactions in parallel by network validators. This method will increase the speed of confirmation and registration of the transaction. Elastico [13], Omniledger [14], and Rapidchain
[15] are some of the basic and important solutions for sharding.

Since in these systems, network nodes are divided into different groups, and each of these groups has a limited number of nodes, the use of traditional consensus algorithms such as PBFT [15] and Raft [16], in each of these groups is possible. Traditional algorithms have high speeds, but because they require peer-to-peer communication with network nodes, it is not possible to use them in networks with a large number of nodes. But in sharding systems where we deal with small groups of nodes, it is possible to use these Algorithms.

Many systems have been proposed in the Sharding discussion, each of which has used one of the traditional algorithms to solve a group's consensus. For example, Elastico
[13] uses PBFT, while Repchain [17] uses a concurrent raft-based algorithm. A point that is not considered in all of these systems is providing an incentive mechanism. In the bitcoin system, the incentive system is properly considered, and each person acquires a bitcoin amount by mining a block. In the case of sharding, this has not been done correctly. In the traditional consensus algorithms used in sharding methods, the algorithm participants were considered to be two groups, honest and malicious. Honest nodes follow the protocol under any circumstances and do not violate it. Malicious nodes can take any arbitrary action to damage the system. These assumptions cannot be true for a system such as a blockchain because the nodes have no incentive to enter the network and participate in the consensus algorithm. In fact, not all blockchain nodes can be considered honest and follow the protocol under any circumstances, but they are rational.

Rational means that they may violate the protocol to increase their benefit. In this way, a rational node does everything to increase his benefit, even if it is a protocol violation. When we do not provide an incentive for nodes to participate in the consensus algorithm, the participants do not benefit from it and do not participate. Some solutions have been proposed to create an incentive system in various sharding methods, but in none of these methods consider the role of the leader in the traditional consensus algorithm. In the traditional consensus algorithm, a leader has an important role in involving other nodes in the consensus algorithms. The incentive mechanism in these methods must encourage the leader to involve the maximum number of nodes in the consensus algorithm. This issue is essential for the security of the system.

One of these solutions that tried to solve sharding's incentive mechanism is Repchain [14], which is explained in the following and is shown that this method cannot properly provide an incentive mechanism.

3.1. Repchain

In this system, the leader of each group is selected based on reputation, and finally, the reward received for building each block is divided among them based on the reputation of the nodes. In this system, in each group, the consensus algorithm is based on the Raft algorithm, an algorithm for synchronous environments. Besides achieving consensus on transactions in each epoch, an agreement also must be used for the reputation of groups nodes and recorded in the reputation blockchain. In the following, we will explain this algorithm more comprehensively.

- Steps of Repchain Algorithm

At the beginning of each epoch, the nodes should be placed in different groups according to their reputation, which is recorded in a separate chain. After assigning nodes to different groups, in each group, a leader should be chosen. To elect a leader, only nodes of the group can be chosen as a leader that their reputation is more than the median of all the group nodes reputation. One of the nodes who has the necessary leadership conditions (sufficient reputation) is randomly selected.

Once the leader is elected, a consensus algorithm must be worked out between the group nodes to achieve a consensus and generate a new block. The consensus algorithm that Repchain uses in its system is a Raft-based algorithm. The step of this algorithm is as follows.

The leader first collects a list of transactions, signs it, and sends it to the rest of the group. The group nodes receive the leader's list, check the correctness of the received message, including the correct signature of the leader on the received list, and then confirm each transaction within this list. Each node expresses her opinion on each of the transactions in the list, which can be "accept", "reject", or "unknown". Unknown mode is for cases where a node does not have the computational power to participate in the confirmation of transactions.

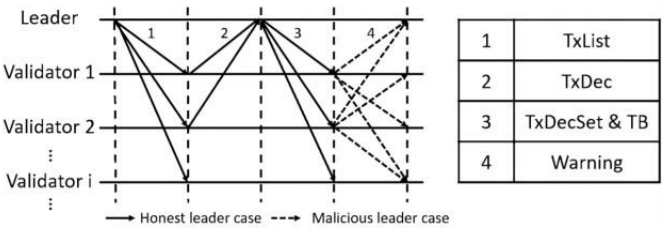

Fig. 2. Raft consensus in Repchain [17]

After each node has commented on the transactions on the list, they sign the list and send it to the leader. The leader examines all the lists received from the network nodes. He selects the transactions that most of the nodes (more than half of the nodes) have agreed to and places them as an acceptable transaction in a block. Then the leader sends all nodes' comments on all transactions to other nodes in the

After registering the block, they must now agree on the reputation of the group's nodes, and the latest reputation of all of the nodes must be registered in the reputation blockchain. To update the reputation in this system, the score of each action gets score as follows:

- Each vote to accept a correct transaction or against a wrong transaction get +1 point.
- Each vote to accept a wrong transaction or against a correct transaction get -1 point.
- Each vote without comment get 0 point.

So, each node updates the reputation of other nodes as relation 1:

$$r_i = \sum_{j=1}^{l} S(j) * T(j) \quad (1)$$

In this relation, *S(j)* is same as the points mentioned in the previous paragraph. *T(j)* is the degree of importance and value of the transaction.

After each node calculates the others' nodes reputation, based on the PBFT consensus algorithm, the group nodes' reputation is recorded in the reputation blockchain.

In the following, we will examine the incentive mechanism of the Repchain method and show that this mechanism cannot work properly and also causes collusion.

3.2. The incentive mechanism in Repchain

Participating in consensus algorithms in Sharding requires an incentive mechanism. As mentioned in the previous section, Repchain has tried to solve this problem. However, an issue that the Repchain does not consider is that in the traditional consensus algorithms, the leader of the group can prevent others from involving in the consensus algorithm to increase his benefit. Since traditional consensus algorithms always require a leader to send the block or message to the other nodes to achieving consensus, at this step, a leader can prevent some nodes of the network from involving in the consensus algorithm by not sending the primary message or block to them. In fact, in the traditional consensus algorithm, more than a defined number of nodes must validate and sign the block for creating a valid block. For example, in the Raft algorithm, more than half of the network nodes must sign the block. So in this system, the leader can prevent some nodes from participating in the consensus algorithm and register a block with the minimum signature.

We will examine the Repchain incentive mechanism and show that this mechanism causes collusion. The leader in this system also tends to involve the minimum number of nodes in the consensus algorithm.

4. Analyze the incentive mechanism in Repchain

As mentioned in the Repchain system, the nodes are first randomly but based on a balanced average reputation belonging to different groups. In each group, based on the reputation of nodes, those who have a reputation above the median of the whole group's reputation can be candidates for becoming a leader. We consider the number of nodes that can become a leader in each epoch equal to L, and n equal to the whole number of nodes of each group.
If we represent the reputation of node i with r_i so that when we arrange the r_{is} in ascending order, we will have a set of nodes reputation as relation 2.

$$r = \{r_1, r_2, \dots, r_n\} \tag{2}$$

Which in this set, the value of median is equal to relation 3

$$m = r_{\lceil \frac{n}{2} \rceil} \tag{3}$$

A node can be a leader candidate if $r_i > m$ If we consider x the number of r_{is} that are equal to m and $i > \left\lceil \frac{n}{2} \right\rceil$, we can say that the number of candidates in each stage is equal to relation 4.

$$L = \left\lceil \frac{n}{2} \right\rceil - x \quad 0 \le x \le \left\lfloor \frac{n}{2} \right\rfloor \tag{4}$$

For each node, we define the utility function as relation 5.

$$U = B - C \tag{5}$$

In this relation C is the cost of participating in the consensus algorithm. B, which is the reward received by creating a block, is calculated as relation 6.

$$B_i = \begin{cases} \left(\frac{R}{2}\right)\left(\frac{r_i}{\sum_{j=1}^{k} r_j}\right) & \text{i = nonleader node} \\ \frac{R}{2} & \text{i = leader node} \end{cases}$$

In this regard, r_i is the reputation of the node i, k is the number of nodes who participate in the consensus algorithm of this epoch. The leader gets half of the reward R and the other half of R is divided between other nodes according to their reputation. In the following, we will examine the incorrectness of this mechanism in two scenarios. In the first scenario, we assume that there is no collusion between the group nodes. In other scenario we consider that the nodes in the group can form a collusion.

4.1. First Scenario: without any collusion

Suppose node i is selected as the leader at this current epoch. In this case, the reward that this leader will receive from participating in the consensus algorithm in this epoch and the next epoch, if he completely follows the algorithm, can be expressed as relation 7:

$$U_i = \frac{R}{2} + \left(\frac{1}{L}\left(\frac{R}{2}\right) + \left(1 - \frac{1}{L}\right)(R_i')\right) \tag{7}$$

In this respect, R is equal to the total reward received from the creation a block by the group. L is equal to the number of nodes that can candidate for leadership and is equal to the reward that node i will obtain if he will be a non-leader node in next epoch. Is as relation 8:

$$R_i' = \left(\frac{R}{2}\right)\left(\frac{r_i'}{\sum_{j=1}^{k} r_j'}\right) \tag{8}$$

In fact, as mentioned before, half of the block reward that remains for other nodes in the network is divided according to the reputation of the nodes among those who participate in the algorithm.
In the traditional consensus algorithm that is used in Repchain, the leader can prevent some nodes from participating in the consensus algorithm; This means that the leader does not send a block to these nodes, or if these nodes send a comment on the current block to the leader, the leader ignores their comments. By this work, the leader can decrease their reputation and, in turn, remove someone from the list of leadership candidates for the next epoch. For example, suppose the leader decreases a node's reputation with a proper condition for being a candidate in the next epoch to a reputation equal to or lower than the median m. In that case, he can successfully remove this node from the list of leadership candidates in the next epoch. So we can calculate the utility function for this node in this epoch and next epoch as relation 9:

$$U_i' = \frac{R}{2} + \left(\frac{R}{2(L-1)} + \left(1 - \frac{1}{L-1}\right)R_i'\right) \tag{9}$$

We should prove that $u_i < u'_i$, for this purpose we assume that $u_i < u'_i$ is correct, so we have the relation 10:

$$U_i < U_i'$$

$$\Rightarrow \left(\frac{1}{L}\left(\frac{R}{2}\right) + \left(1 - \frac{1}{L}\right)(R_i')\right) < \left(\frac{1}{L-1}\left(\frac{R}{2}\right) + \left(1 - \frac{1}{L-1}\right)(R_i')\right)$$

$$\Rightarrow \left(\frac{1}{L-1} - \frac{1}{L}\right)R_i' < \left(\frac{1}{L-1} - \frac{1}{L}\right)\frac{R}{2}$$

$$\Rightarrow R_i' < \frac{R}{2} \tag{10}$$

It is always true that the leader reward is more than reward of a non-leader nods. So the above relationship is true. As a result, since we said that network nodes are rational, so they do everything to increase their profit. Here we saw that the violation of the protocol for a rational node is beneficial.

4.2. First Scenario: without any collusion

This mechanism will also increase the motivation to create collusion because the rewards are divided among the other nodes based on their reputation. If we assume that T nodes want to collude, each of this T node if is chosen as the leader,

tends to reduce the reputation of the other nodes in the network to increase the reputation of his group. So he tries to involve the least number of nodes in the consensus algorithm and register the block with the minimum available signature.
The first case assumes that the leader follows the protocol and involving k node in the consensus algorithm. In this case, the utility function for the leader and his colluding group is as relation 11:

$$U_i = U_i + U_{-i}$$

$$U_i = \frac{R}{2}$$

$$U_{-i} = \sum_{i=1}^{T-1} \left(\frac{R}{2}\right)\left(\frac{r_i}{\sum_{j=1}^{k} r_j}\right) \quad (11)$$

If the leader prevents some of the k nodes that wants to participate in the consensus algorithm, in this case, the total reputation of the nodes decreases. If we show this amount of reduction with r' then the utility function of the colluding group will be as relation 12:

$$U_t = U_l + U_{-l}$$

$$U_l = \frac{R}{2}$$

$$U'_{-l} = \sum_{i=1}^{T-1} \left(\frac{R}{2}\right)\left(\frac{r_i}{\sum_{j=1}^{k} r_j}\right) - r' \quad (12)$$

It is clear that relation 12 is bigger than relation 11, so by violating the protocol, the leader and his group's benefit will increase. So this incentive mechanism causes that nodes tend to form collusion, and also the leader will tend to violate the protocol and not involve some nodes in the consensus algorithm.
In fact, this system's main drawback was that by motivating nodes to participate in the consensus algorithm, this motivation caused a violation in the algorithm by nodes

5. Analysis of other sharding methods

In the previous section, we have examined the Repchain method and found that its proposed solution for incentive does not work correctly.
In this section, we generally elaborate on sharding solutions that use traditional consensus algorithms, and these algorithms need a leader for achieving consensus. We show that these methods have similar challenges to Repchain.
In these methods, the leader sends the initial block to the other nodes of the network. These nodes must confirm or reject the initial block. Depending on which consensus algorithm is used, the steps can be more complex, but the general steps of these algorithms are as described. It is obvious that the leader has an important role in these consensus algorithms. The leader initiates the consensus process and can prevent some nodes from participating in the consensus algorithm. The leader can do it by not sending initial block to some nodes or ignore their opinions on the final block.
For example, in the following PBFT [19] consensus algorithm, if the leader does not send the initial block to a node in the first step, this node will not be able to continue participating in the consensus algorithm. In this algorithm, if the group has n nodes, the leader knows that only with the participation of 2/3 of these nodes can build the block and receive a reward. So the leader can easily prevent 1/3 of the group nodes from continuing to participate in the consensus algorithm without losing the reward.

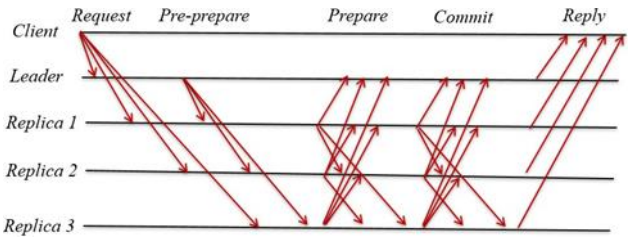

Fig. *3*: PBFT consensus [21]

Similarly, in other consensus algorithms, a minimum number of signatures is required to register the block, which is enough for the leader to collect this minimum number of signatures and ignore other nodes.
When the leader does not involve all of the nodes in participating in the consensus algorithm, it causes damage to the system. In fact, this can decrease the incentive of other nodes to remain in the network and tend to participate in the consensus algorithm. It can also increase the tendency of some nodes to collude with each other because when some nodes create collusion and try to ignore other nodes and don't involve them in the consensus algorithm, they can obtain more reward. It is obvious that collusion will bring many security problems.
Therefore, in sharding that uses the traditional consensus algorithm, it is very important to consider the role of the leader in involving other nodes in the consensus algorithm. So it is important to have an incentive mechanism and reward sharing system that encourages the leader and other nodes to act properly in the system.
In some proposed solutions such as solutions in paper [20] and [21], rewards are distributed among the participant nodes, this will increase the tendency of the leader to deviating from the correct action and does not involve all of the node in the consensus algorithm and also causes to creating collusion in a similar way to Repchain. So, we need an incentive mechanism to solve these problems.

6. Future work

One of the problems with sharding methods and the use of traditional consensus algorithms is not considering the role of the leader in involving or not involving other network nodes in the consensus algorithm. The problem with the Repchain system can also be seen in other methods. We intend to introduce an incentive mechanism that solves the problems mentioned in the proposed method and can properly motivate network nodes to participate in the consensus algorithm and act correctly.

7. Conclusion

We elaborate on the problems that consensus algorithms have in the blockchains and state that one of the main problems of blockchain is scalability. In this regard, we mention previous work that has tried to solve the problem of scalability of

blockchain. One of these methods is sharding. We explained the proposed methods in sharding and finally the challenge that these methods face. In one of the previous methods called Repchain, we have given a brief explanation of why the recent method has not been entirely successful in motivating network nodes. For this purpose, we provide two scenarios to show why this algorithm cannot succeed in providing an incentive mechanism. At the end, we also elaborate on other sharding system that use traditional consensus algorithm and show that all of these methods have some challenges for motivating network nodes.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Switzerland: Nakamoto S.*, 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf
[2] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *Proc. IEEE Int. Congr. Big Data (BigData Congress)*, 2017, pp. 557–564.
[3] F. Sur and B. Tschorsch, "Bitcoin and beyond: A technical survey on decentralized digital currencies," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 2084–2123, 2016.
[4] W. Wang *et al.*, "A survey on consensus mechanisms and mining management in blockchain networks," *arXiv preprint arXiv:1805.02707*, 2018.
[5] S. King, "Primecoin: Cryptocurrency with prime number proof-of-work," Jul. 7, 2013. [Online]. Available: https://primecoin.io
[6] A. Miller *et al.*, "Permacoin: Repurposing bitcoin work for data preservation," in *Proc. IEEE Symp. Security and Privacy (SP)*, 2014, pp. 475–490.
[7] M. Ball *et al.*, "Proofs of useful work," *IACR Cryptology ePrint Archive*, vol. 2017, p. 203, 2017.
[8] Q. Zhou *et al.*, "Solutions to scalability of blockchain: A survey," *IEEE Access*, vol. 8, pp. 16440–16455, 2020.
[9] Bitcoin Core, "Segregated witness benefits," 2016. [Online]. Available: https://bitcoincore.org/en/2016/01/26/segwit-benefits/
[10] J. Poon and T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments," 2016. [Online]. Available: https://lightning.network/lightning-network-paper.pdf
[11] J. Poon and V. Buterin, "Plasma: Scalable autonomous smart contracts," 2017. [Online]. Available: https://plasma.io
[12] Raiden Network, "Raiden Network," Accessed: Sep. 1, 2019. [Online]. Available: https://raiden.network
[13] L. Luu *et al.*, "A secure sharding protocol for open blockchains," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2016, pp. 17–30.
[14] E. Kokoris-Kogias *et al.*, "Omniledger: A secure, scale-out, decentralized ledger via sharding," in *Proc. IEEE Symp. Security and Privacy (SP)*, 2018, pp. 583–598.
[15] M. Zamani, M. Movahedi, and M. Raykova, "RapidChain: Scaling blockchain via full sharding," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2018, pp. 931–948.
[16] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *Proc. Symp. Oper. Syst. Design Implement. (OSDI)*, 1999, pp. 173–186.
[17] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *Proc. USENIX Annu. Tech. Conf. (USENIX ATC)*, 2014, pp. 305–319.
[18] C. Huang *et al.*, "RepChain: A reputation based secure, fast and high incentive blockchain system via sharding," *arXiv preprint arXiv:1901.05741*, 2019.
[19] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *Proc. Symp. Oper. Syst. Design Implement. (OSDI)*, 1999, pp. 173–186.
[20] M. H. Manshaei, M. Jadliwala, A. Maiti, and M. Fooladgar, "A game-theoretic analysis of shard-based permissionless blockchains," *IEEE Access*, vol. 6, pp. 78161–78177, 2018.
[21] The Harmony Team, "Open consensus for 10 billion people," 2018. [Online]. Available: https://harmony.one

Mojdeh Hemati received her MSc degrees in software engineering from Amirkabir University of Technology, in 2019

Email: m.hemati0094@aut.ac.ir

Mehdi Shajari Received his Ph.D. degrees in computer engineering

Email: mshajari@aut.ac.ir