Computer Modeling in Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2023.026920

ARTICLE

Vertical Federated Learning Based on Consortium Blockchain for Data Sharing in Mobile Edge Computing

Yonghao Zhang[1,3], Yongtang Wu[2], Tao Li[1], Hui Zhou[1,3] and Yuling Chen[1,2,*]

[1]State Key Laboratory of Public Big Data, College of Computer Science and Technology, Guizhou University, Guiyang, 550025, China

[2]Blockchain Laboratory of Agricultural Vegetables, Weifang University of Science and Technology, Weifang, 262700, China

[3]Guangxi Key Laboratory of Cryptography and Information Security, Guilin University of Electronic Technology, Guilin, 541004, China

*Corresponding Author: Yuling Chen. Email: ylchen3@gzu.edu.cn

Received: 03 October 2022 Accepted: 22 December 2022

ABSTRACT

The data in Mobile Edge Computing (MEC) contains tremendous market value, and data sharing can maximize the usefulness of the data. However, certain data is quite sensitive, and sharing it directly may violate privacy. Vertical Federated Learning (VFL) is a secure distributed machine learning framework that completes joint model training by passing encrypted model parameters rather than raw data, so there is no data privacy leakage during the training process. Therefore, the VFL can build a bridge between data demander and owner to realize data sharing while protecting data privacy. Typically, the VFL requires a third party for key distribution and decryption of training results. In this article, we employ the consortium blockchain instead of the traditional third party and design a VFL architecture based on the consortium blockchain for data sharing in MEC. More specifically, we propose a V-Raft consensus algorithm based on Verifiable Random Functions (VRFs), which is a variant of the Raft. The V-Raft is able to elect leader quickly and stably to assist data demander and owner to complete data sharing by VFL. Moreover, we apply secret sharing to distribute the private key to avoid the situation where the training result cannot be decrypted if the leader crashes. Finally, we analyzed the performance of the V-Raft and carried out simulation experiments, and the results show that compared with Raft, the V-Raft has higher efficiency and better scalability.

KEYWORDS

Mobile edge computing; vertical federated learning; consortium blockchain; consensus algorithm

1 Introduction

In the Internet of Things (IoT), the massive amount of data collected by devices has to be analyzed, processed, and stored. Mobile Edge Computing (MEC) [1] pushes the manipulation of data in the IoT from the centralized cloud server to the network edges, so as to reduce network latency and increase the corresponding speed of IoT devices. At the same time, the devices in the edge network transmit the filtered data to the cloud server for analysis. The data can improve and optimize its own MEC services, and it also contains tremendous market value. Data owners share data through trading or

collaboration which can maximize the usefulness of the data. However, data sharing is not as easy as it seems. The data in MEC is vast and diverse [2], and some of it is quite sensitive, such as home addresses, driving routes, and people's behaviors or even physical characteristics. As a result, direct data sharing is not allowed in some cases because it violates privacy. Prohibiting data circulation and sharing will diminish the value and impact of the data, which in turn limits the development and application of the MEC. Therefore, data sharing should be done in a way that ensures that people's privacy is not leaked, and that is an element that system developers must consider [3]. Federated learning is a distributed machine learning framework with privacy protection [4–6], which can transfer encrypted intermediate results to complete model training without leaking the raw data during the training process. That is, federated learning indirectly shares data to build a better machine learning model and the data is kept in the local context. According to the distribution characteristics of the data, federated learning is classified into Horizontal Federated Learning (HFL) and Vertical Federated Learning (VFL) [7]. The VFL is applicable to the case that two training participants have their own data sets, and there are many overlapped samples, but few overlapped features in their data sets. With the assistance of a trusted third party, the two participants use encryption model training schemes [8,9] to combine different features from the common samples, and a more accurate model can be obtained by expanding the features of samples. Thus it can be seen the VFL can build a bridge between data demander and owner to complete data sharing while protecting data privacy.

In the training process of the VFL, the third party plays an important role. It generates a public-private key pair, in which the public key is distributed to the participants for homomorphic encryption [10] of the data, and then the private key will be used to decrypt the training results. Therefore, we request that the third party is credible and reliable. Blockchain [11] is a kind of distributed ledger, which has the features of decentralization, tamper-resistance, and security. Based on these advantages, blockchain is often used in privacy protection [12,13] and identity authentication [14,15]. Depending on the degree of openness of the network, blockchain can be divided into public blockchain, consortium blockchain, and private blockchain. The consortium blockchain is usually composed of various institutions, which is suitable for building a distributed network with fewer nodes. Because of the access mechanism, the consortium blockchain is highly reliable. In this article, we employed consortium blockchain as the third party and proposed a VFL architecture based on consortium blockchain for data sharing in MEC. The consortium blockchain uses consensus algorithm to selected leader and committee, which assist data demander and owner to complete data sharing through the VFL approach. Moreover, the leader and committee will record the relevant information about the data sharing. When the sharing is completed, the leader will pack the data records into a block and upload it to the consortium blockchain.

Consensus algorithm is the core of blockchain, which can ensure the consistency of distributed nodes. Raft [16] is a common consensus algorithm in the consortium blockchain. The working mechanism of Raft is to elect a leader from the distributed cluster, the leader accepts the request from the client and forwards it to other nodes in the form of logs. When receiving the response from most nodes, the request is submitted to the local state machine, and in this way, the distributed nodes can reach consensus. However, Raft has some issues that can be improved upon. The Raft uses voting for leader election, which is susceptible to network partitioning, and as the number of nodes increases. The communication overhead increases. Moreover, if the leader crashes, Raft will re-elect a new leader from the followers to replace it. But in the scenario of VFL, the leader holds the private key and we need the new leader can recover the private key to continue the model training. In this article, we have optimized and tackled the above issues, and our main contributions are as follows:

- We modify the Raft based on verifiable random functions and propose a new consensus algorithm, V-Raft. The V-Raft utilizes a method of sortition for leader election, which enhances the election speed and scalability. In addition, the V-Raft adds the committee state, which reduces node consensus time and replacement time.
- We introduce secret sharing to distribute the private key to avoid data sharing failure due to the leader crashes. The leader assists the data demander and owner in VFL and decomposes the private key into subkeys to send to the committee. If the leader crashes, the committee can collect the subkeys to recover the private key and ensure that the data sharing is completed smoothly.
- We design simulation experiments to compare Raft and V-Raft. The experimental results show that V-Raft has better efficiency and performance than Raft.

2 Related Work

In recent years, with the increased awareness of privacy protection, a variety of privacy protection schemes have been proposed [17–19]. As an important technology of privacy computing, the focus on federal learning continues to grow [20,21]. In 2019, more than 50 scholars from Google, Stanford, CMU and other institutions have come together to summarize the advances and problems in the field of federated learning and published a review paper [22]. In this paper, federated learning is classified into cross-device federated learning and cross-silo federated learning. Among them, the cross-silo federated learning corresponds to VFL proposed by Yang et al. [7]. The VFL has a wide range of applications in intelligent transportation and personalized recommendation [23]. For example, The work in [24] used VFL to train the traffic flow prediction model, since the participants' traffic flow datasets have the same sample space and different spatial characteristics, and VFL can share parameters while ensuring privacy. In [25], a cloudlet-based recommender model was proposed for electric vehicles to find the most relevant charging station, and the model utilizes the VFL technique so that the data does not leave the local. The cloudlets are data aggregators, and blockchain creates a secure network composed of only trusted cloudlets, as the third party in VFL.

Blockchain was born in 2008 with the emergence of Bitcoin [26]. Even to this day, blockchain technology is still very popular and has produced some rich research results. For example, blockchain technology has been introduced to improve security in mobile crowdsourcing [13], unmanned aerial vehicle [27], and healthcare [28]. Game theory [29] is a scientific analysis tool, which is often applied to the computer field [30]. In [31], using common concepts in game theory to address the rational behaviours in blockchain, and to achieve some desired conclusions. Selfish mining is an attack method in the blockchain, and the study in [32] proposed an improved selfish mining based on hidden Markov [33] decision processes to maintain the benefit from selfish mining. For the attack method mentioned above, [34] indicates that the semi-selfish mining attacks will be detected, and semi-selfish mining is impossible in practice. In the field of consensus algorithm, there is also a lot of research. Traditional consensus algorithms include PoW [35] and PoS [36], which are usually used in public blockchain with a large number of nodes. In consortium blockchain without Byzantine nodes, the Raft consensus algorithm is widely used. Verifiable Random Functions (VRFs) [37] is a low-energy, high-efficiency random number algorithm and provides an asymmetric key verification mechanism. Some current blockchain projects use VRFs to select nodes, such as Algorand [38], Dfinity [39].

3 Preliminaries

3.1 Vertical Federated Learning

VFL [7] is also known as feature-based federated learning. Let D_i denote the dataset held by each data owner i, X denotes the feature space, Y denotes the label space, and I denotes the sample ID space. VFL is expressed as:

$$X_i \neq X_j,\ Y_i \neq Y_j,\ I_i = I_j, \forall D_i,\ D_j,\ i \neq j \tag{1}$$

That is, the two datasets in VFL have a large number of overlapping samples, and these same samples each have different features. In addition, usually only one side of the dataset has sample labels. In VFL, a typical assumption is that the training participants and a third-party collaborator are honest-but-curious, that is, they will abide by the agreement but will try to get additional information from the training process. Moreover, the third-party collaborator does not collude with participants. The VFL consists of two parts. Part 1 is encrypted sample alignment, which is to find common samples of training participants by Private Set Intersection (PSI). The PSI can calculate the intersection of samples held by both parties without revealing additional information. Part 2 is encrypted model training. Training participants combine features of the common samples to train the machine learning model.

3.2 Raft Consensus Algorithm

Raft [16] decomposes the consensus problem into three relatively independent subproblems:

- **Leader election:** Raft has three states, which are follower, candidate and leader. Initially, all nodes are followers. When the cluster is started up or the leader crashes, a new leader will be elected. Each follower is set with a random timer, and when the timer times out, the follower converts to the candidate and sends a message requesting voting to other nodes. When more than half of the votes in the cluster are received, the candidate converts to leader, and the leader regularly sends heartbeat messages to other nodes to maintain its state.
- **Log replication:** The leader receives a request from the client and copies it as a log entry to followers in the cluster. When the leader receives the entry replication response sent back from a majority of the followers, the leader submits the log entry to the state machine and returns the execution result to the client. Once a follower receives a new log replication message or heartbeat message from the leader, it will commit the log entry to its local state machine, thus ensuring the log consistency between the leader and followers.
- **Safety:** Raft ensures that all nodes in the cluster execute the same instructions in the same order through relevant restrictions and rules.

3.3 Verifiable Random Functions

Verifiable Random Functions (VRFs) [37] means that the owner of the private key can calculate a hash value, while others can verify the correctness of the hash value through the public key corresponding to the private key, and the verification process will not expose the private key.

The VRFs constructed based on RSA is as follows:

$$< hash, proof >= VRF\_hash(sk, m) \tag{2}$$

$$True/False = VRF\_verify(pk, m, hash, proof) \tag{3}$$

Using the private key *sk* and an arbitrary value *m* as input to the *VRF_hash* function, a *hash* value and a *proof* value can be generated. The *VRF_verify* function is used to verify the correctness of the *hash* value. If the output is true, it means that the *hash* value is generated by *sk* and *m*, and the *sk* corresponds to the *pk*. In addition to the above verifiability, VRFs also has the following properties:

- **Uniqueness:** For any *hash* value generated by *VRF_hash* function, there is a unique proof value that can be proved to be valid.
- **Collision resistance:** Like any secure hash function, the *hash* value output by the VRFs is collision-resistant. More precisely, it should be computationally infeasible for two different inputs to get the same output.

4 Model

4.1 Model Structure

The VFL architecture based on consortium blockchain is shown in Fig. 1. Firstly, data demander and owner as training participants A and B, respectively. They send an assistance request of VFL to the consortium blockchain. After receiving the request, the consortium blockchain uses the V-Raft consensus algorithm to elect leader and committee. Then, as a third-party collaborator, the leader and committee assist the participants to start VFL. When the training is completed, the leader records the relevant parameters of the training process to form a block and uploads it to the consortium blockchain. The goal and assumption of our proposed structure are as follows.

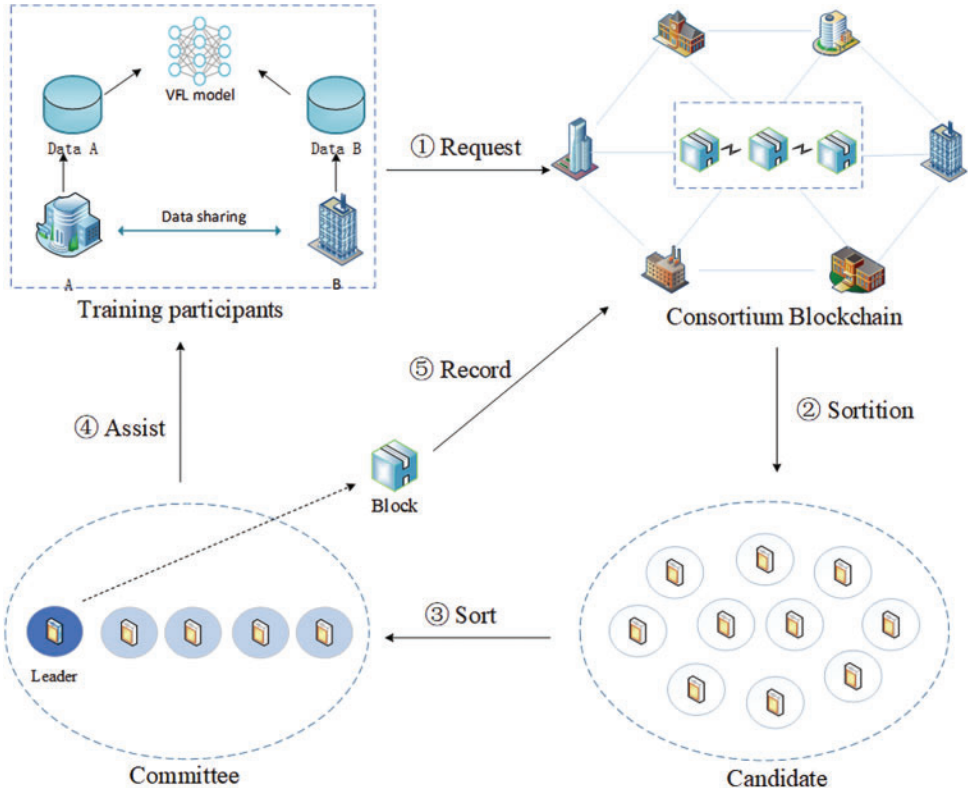

Figure 1: VFL architecture based on consortium blockchain

- **Goal:** Our goal is that the consortium blockchain can respond quickly to training assistance requests sent by training participants. The elected leader first generates a public-private key pair for homomorphic encryption. The public key is used to encrypt the data of the training participants. After a series of encryption operations, the training participants send the encrypted gradients G_A, G_B to the leader, which decrypts the gradients using the private key and returns them to the participants. Through multiple rounds of gradient calculation, until the model training is completed. And as a third party, the nodes elected by the consortium blockchain will not leak the privacy of the participants in the training process. In addition, when the leader crashes, it can be replaced in time.
- **Assumption:** We assume that the consortium blockchain is composed of data owners such as government, banks, hospitals, enterprises and other institutions. The access mechanism of the consortium blockchain will audit and verify the identity of nodes, so this model sets the nodes in the consortium blockchain to be honest-but-curious and not colluding with the training parties. According to the VFL setting, the third party does not collude with training participants.

4.2 V-Raft Consensus Algorithm

There are several problems with the current Raft algorithm: (1) Raft uses a voting mechanism for the leader election, which may fail due to network partitioning so that no node can get more than half of the votes. (2) In Raft, as the number of nodes in the consortium blockchain increases, the time of leader election, leader replacement and consensus increases. In response to the above problems, we propose an improved consensus algorithm based on the VRFs, called the V-Raft. Firstly, V-Raft uses a sortition algorithm based on VRFs, which can stably elect the leader and improve the election speed at the same time. And according to the characteristics of VRFs, the selected nodes are random and verifiable. In addition, we add the consensus committee node role, which can improve the consensus speed. If the leader crashes, select the qualified nodes in the committee to replace, which can improve the leader replacement speed.

4.2.1 Node States and Transitions

V-Raft sets four node states: follower, candidate, committee, and leader. Some nodes of 5 are a typical setting in Raft, which allows the system to tolerate two crash nodes. In a practical scenario, the probability of three nodes crashing simultaneously in a cluster containing five nodes is very small. Therefore, we set the number of committee nodes to 5 in V-Raft. Initially, all nodes in the consortium blockchain are followers. When receiving the assistance request of VFL sent by the training participants, followers execute sortition algorithm and generate a hash. If the hash matches the condition, follower is converted to candidate. Then, sorting the hash of candidates, the five candidates with the smallest hash value are elected as committee and the smallest hash value is selected as leader. The node states and their transitions are shown in Fig. 2.

4.2.2 Leader Election

Suppose that training participants A and B want to start VFL, and A and B send a request to the consortium blockchain. The request contains the public keys of A and B and a *seed*. The public key is issued by the consortium blockchain, which can be used for node identity authentication. The *seed* is a random number jointly generated by A and B using the Diffie-Hellman algorithm.

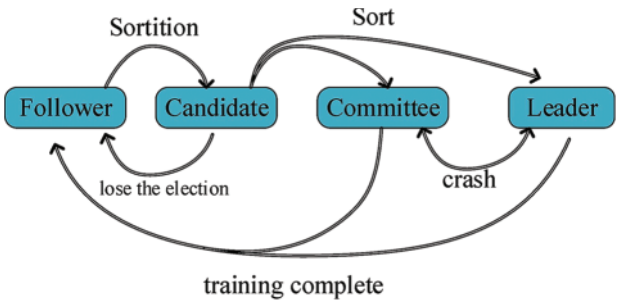

Figure 2: Node states and transitions

When followers in the consortium blockchain receive the request, they use the sortition algorithm to select candidate. As shown in Algorithm 1, a follower uses its private key and the *seed* in the request as the input of the VRFs function and outputs a *hash* value and a *proof* value. It then compares the size of $hash/2^{hashlen}$ and λ, where *hashlen* is the bit-length of *hash*, thus is essentially uniformly distributed between 0 and 1, and λ is the sortition threshold, which is set by the consortium blockchain. If *hashlen* is less than λ, then the follower changes to candidate. Under the control of λ, usually, at least 5 nodes will be selected as candidates each time, but there is a very small probability that the number of candidates is less than 5, which cannot meet the minimum number of the committee. In this case, the consortium blockchain can appropriately raise the threshold to increase the number of candidates selected. The candidate generates *m_candidate*[*u_pk*, *hash*, *proof*, *seed*] message and broadcast.

Algorithm 1: Sortition

Input: *u_sk*, *seed*
Output: *hash*, *proof*, *seed*, *true*/*false*
1: $< hash, proof > \leftarrow VRF\_hash(u\_sk, seed)$
2: **if if** $hash/2^{hashlen} < \lambda$ **then**
3: **return** *hash*, *proof*, *seed*, *true*
4: **else**
5: **return** *hash*, *proof*, *seed*, *false*
6: **end if**

After all candidates receive the message *m_candidate*, Algorithm 2 is used to select the committee and leader. First, a candidate uses the *VRF_verify* function to verify the hash and proof sent by the other candidates. If the verification result is true, put the hash value and its corresponding public key *u_pk* into the list. Next, the candidate uses the sorted function to sort the elements in the list. The sorting rule is to compare the *hash* values and sort them from smallest to largest. The sorted function will return a *committee_list*, which contains the 5 nodes with the lowest hash value. These 5 nodes constitute the consensus committee, and the node with the smallest hash value is elected as the leader. The leader generates *m_committee*[*committee_list*, *seed*] message and broadcast.

Algorithm 2 : Sort

Input: *hash*, *proof*, *seed*
Output: *committee_list*, *true*/*flase*
1: $list \leftarrow \{\}$
2: **while** $VRF\_verify(u\_pk, hash, proof, seed) = true$ **do**

(Continued)

Algorithm 2 (Continued)

3: $list \cup = \{[u\_pk, hash]\} committee\_list \leftarrow sorted(list)$
4: **end while**
5: **if** $hash{=}committee\_list[i].hash, i = \{0, \ldots, 4\}$ **then**
6: **return** *committee_list*, *true*
7: **else**
8: **return** *committee_list*, *false*, *false*
9: **end if**

When the *m_committeemessage* is received, the committee will compare the *committee_list* with its own sorted list. If it is the same, it will respond to the message of the leader. When the leader receives a response from more than half of the committee, the leader will regularly send a periodic heartbeat to the committee to maintain its authority. Furthermore, the leader establishes a connection with A and B to start VFL.

If the election fails due to network or other reasons, the candidates do not receive the heartbeat message from the leader within the specified time, then the leader election will be repeated and the qualified candidate will broadcast the message containing its own public key and hash again.

4.2.3 Leader Replacement

In Raft, time is divided into terms of arbitrary length, and each term elects a leader. Index is the log entry number of each node, and the Raft uses term number and log index to check the consistency. In V-Raft, the term of each elected leader is one VFL from start to finish, so the term of each federation learning is the same, and we only consider the index to maintain consistency. If the leader crashes, the committee cannot receive the heartbeat of leader, then the leader will be replaced. The replacement node that meets the conditions is elected from committee, and this condition is that the log index number of the node is the maximum. If the index is the same, the node with the smallest hash value is elected as the leader. The node replacement process is shown in Algorithm 3.

Algorithm 3: Replace

Input: *hash*, *index*
Output: *replace_list*, *true*/*flase*
1: $list \leftarrow \{\}$
2: $list \cup = \{[index, hash]\} replace\_list \leftarrow compare(list)$
3: **if** $hash{=}replace\_list[0].hash$ **then then**
4: **return** *replace_list*, *true*, *true*
5: **else**
6: **return** *replace_list*, *false*
7: **end if**

4.3 VFL Based on Consortium Blockchain

The consortium blockchain uses the V-Raft consensus algorithm to select a leader and committee to assist the training participants A and B to start joint model training. First, leader generates a public-private key pair, and the public key is sent to A and B for encryption and decryption of the intermediate parameters. In order to avoid the leader crashing during the training process and the gradient cannot be decrypted, the leader sends the private key to the committee node by secret sharing method. If the

leader crashes, the replacement node can collect the subkeys to recover the private key to continue the model training. When the training is completed, the leader will chain up the training gradient and loss. The VFL based on consortium blockchain is shown in Fig. 3. The committee node stores the hash and index of the message in the list and uses the compare function to sort the index in the list from largest to smallest. If the index is the same, the committee node sorts the hash in the list from smallest to largest.

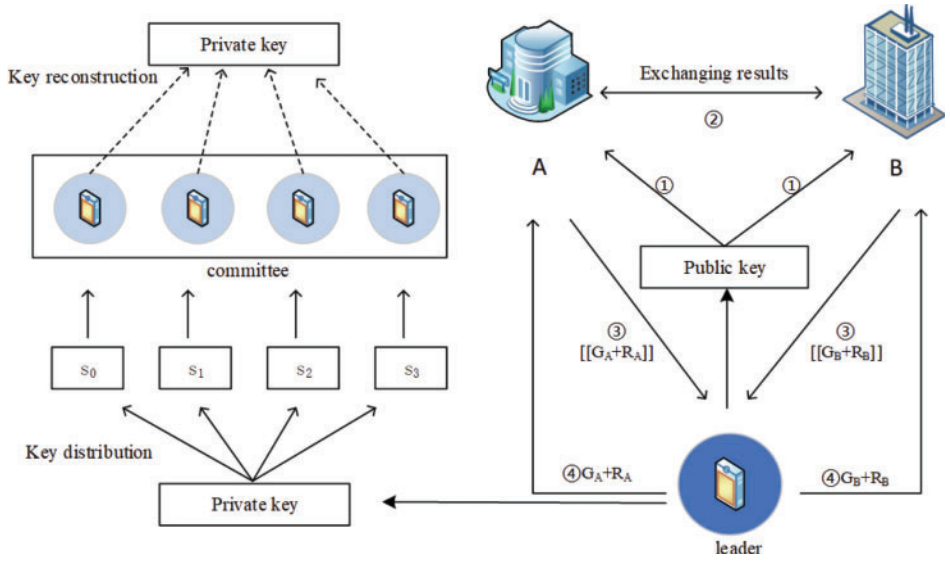

Figure 3: VFL based on consortium blockchain

4.3.1 Key Distribution

The leader generates a public-private key pair and sends the public key to the training participants A and B as a homomorphic encryption key. Homomorphic encryption allows the calculation of the encrypted data, and the calculation process will not disclose the plaintext information. The calculation result is still encrypted, and the result obtained after decryption is the plaintext after processing. This paper uses the homomorphic encryption scheme [9], which has the following properties:

$$[[u+v]] = [[u]] \cdot [[v]] \tag{4}$$

The leader uses secret sharing [40] to distribute the private key to the committee node. Secret sharing can divide the secret S into n-member secrets, and any t-share can reconstruct the secret S. More specifically, the leader uses the private key to construct a 2-order polynomial $f(x)$, selects 4 pairs $(x, f(x))$ and distributes them to the committee nodes, and any 3 of the 4 committee nodes can solve the polynomial coefficients by combining $(x, f(x))$ to reconstruct the private key. The formal definition of secret sharing is as follows:

$$S(s,t,n) - > \{< s_0 >, < s_1 >, ..., < s_n >\} \tag{5}$$

If the leader crashes, the committee uses Algorithm 3 to select the replacement leader. The new leader collects subkeys from the committee and reconstructs the private key. We stipulate that three subkeys can recover the private key. And then, the new leader establishes a connection with the training participants and continues VFL.

4.3.2 Encrypted Model Training

In this article, we use linear regression model training based on homomorphic encryption as an example. Suppose that the features of the common samples of A and B are x_i^A, x_i^B, respectively, θ_A, θ_B are model parameters corresponding to features, and B has label data y_i, the objective loss function is:

$$loss = \sum_i ||\theta_A x_i^A + \theta_B x_i^B - y_i||^2 + \frac{\alpha}{2}(||\theta_A||^2 + ||\theta_B||^2) \tag{6}$$

Let $u_i^A = \theta_A x_i^A$, $u_i^B = \theta_B x_i^B$, $[[\cdot]]$ denotes homomorphic encryption, then the encrypted loss function is:

$$[[loss]] = [[\sum_i (u_i^A)^2 + \frac{\alpha}{2}||\theta_A||^2]] + [[\sum_i (u_i^B)^2 + \frac{\alpha}{2}||\theta_B||^2]] + 2[[\sum_i u_i^A(u_i^B - y_i)]] \tag{7}$$

Let $[[l_A]] = [[\sum_i (u_i^A)^2 + \frac{\alpha}{2}||\theta_A||^2]]$, $[[l_B]] = [[\sum_i (u_i^B)^2 + \frac{\alpha}{2}||\theta_B||^2]]$, $[[l_{AB}]] = 2[[\sum_i u_i^A(u_i^B - y_i)]]$, then:

$$[[loss]] = [[l_A]] + [[l_B]] + [[l_{AB}]] \tag{8}$$

Let $[[d_i]] = [[\theta_A x_i^A + \theta_B x_i^B - y_i]]$, and calculate the partial derivatives of A and B, respectively, then:

$$[[G_A]] = [[\frac{\partial l}{\partial \theta_A}]] = [[d_i]]x_i^A + [[\alpha\theta_A]] \tag{9}$$

$$[[G_B]] = [[\frac{\partial l}{\partial \theta_B}]] = [[d_i]]x_i^B + [[\alpha\theta_B]] \tag{10}$$

The training process is as follows:

1) A uses the public key as homomorphic encryption key to calculate $[[u_i^A]]$, $[[l_A]]$ and sends them to B. B calculates $[[d_i]]$ to send to A, and calculates $[[loss]]$ to send to leader.
2) According to Eq. (5), A and B calculate $[[G_A + R_A]]$, $[[G_B + R_B]]$ and send to leader, respectively. The R_A and R_B are additional mask.
3) Leader uses private key to decrypt encrypted gradient and sends $G_A + R_A$, $G_B + R_B$ and loss to A and B. A and B remove mask to get the gradient G_A and G_B, and use it to update the model, respectively. Repeat the above process until the training is completed.
4) When the training is completed, the leader packages each round of training logs to form a block and uploads it to the consortium blockchain, and other nodes synchronize the block.

5 Analysis of Algorithm

5.1 Safety Analysis

Election safety means that at most one leader can be elected in a given term. In V-Raft, leader election is based on VRFs. The nodes in the consortium blockchain use their own private keys and the seed of the training participants as inputs of VRFs, and output hash value and proof value. The VRFs in this article is constructed based on the sha-256 algorithm. The sha-256 is a secure hash algorithm, which has strong collision resistance, that is, the probability that different inputs get the same output is extremely low. The private key of node is unique and confidential, and the seed of the training participants is variable. Therefore, the output hash value is unique. The V-Raft sorts the hash values to elect leader, so the elected leader is also unique. Similarly, when the leader crashes and a replacement node is selected from the committee, the committee will compare the index and the hash value, and since the hash value is unique, the new leader selected is also only one.

State machine safety: if a node has applied a log entry at a given index to its state machine, no other node will ever apply a different log entry for the same index. That is, the state machines of more than

half of the committee nodes store the same log entries under the same term. The leader forwards the message sent by the training participant to the committee in the form of a log entry. When receiving the reply from more than 2 committee nodes (The number of committee nodes is 5), the leader submits the log entry to its own state machine, and this operation is included in the next heartbeat. When the committee nodes receive the heartbeat, they update their local state machines, so that more than half of the committee nodes reach a consensus to store the same log entries.

5.2 Liveness Analysis

Liveness is defined as a transaction being completed in a limited time. In the leader election stage of the V-Raft, the probability and number of selected nodes are controlled by the sortition threshold λ, and the appropriate λ value ensures that the leader and the committee can be successfully elected within a limited time in each VFL task. In addition, the timeout retransmission mechanism ensures that a new round of leader election is performed when the election fails. In the node replacement stage, if the leader crashes, the qualified nodes will be selected from the committee for replacement to ensure the completion of the model learning.

6 Experimental Results

6.1 Sortition Threshold λ Setting

In V-Raft, the number of committee nodes is 5, so at least 5 candidates need to be selected for each sortition. The number of candidates is related to the sortition threshold λ, and the appropriate λ can ensure the success of the election. We conducted 1000 sortition tests with the total number of nodes $n = 50$ and $n = 100$, respectively, and at least 5 nodes selected by each sortition are denoted as success. The experimental results are shown in Figs. 4 and 5, with the sortition threshold λ increases, the probability of sortition success increases. When the number of nodes $n = 100$, $\lambda = 0.15$, the success rate of 1000 sortition is 100%. When the number of nodes $n = 50$, $\lambda = 0.3$, the success rate of 1000 sortition is 100%.

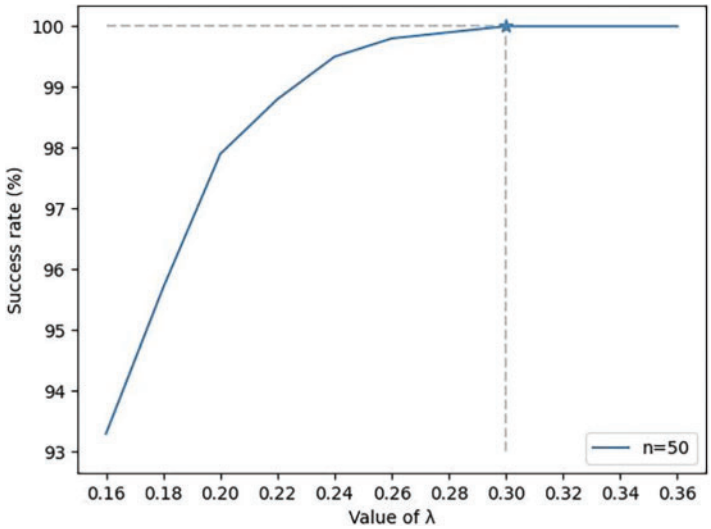

Figure 4: n=50, node sortition success rate

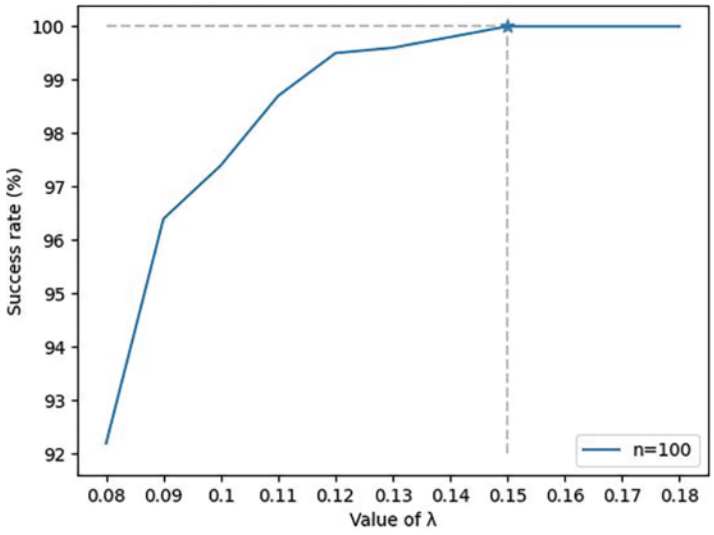

Figure 5: n=100, node sortition success rate

In addition, we tested the success rate when the $\lambda = 15/n$, and the number of nodes is different, as shown in Table 1. The experimental results show that the sortition threshold λ is dynamically adjusted as the total number of nodes in the consortium blockchain increases. When the λ is set to 15/n, the election success rate is high, that is, at least 5 nodes can be elected more stably in each sortition.

Table 1: λ=50, node sortition success rate

Number of nodes n	Sortition threshold λ	Number of sortition	Number of selected node < 5	Success rate (%)
50	0.30	1000	0	100
100	0.15	1000	0	100
150	0.10	1000	1	99.9
200	0.075	1000	1	99.9
250	0.06	1000	0	100

6.2 Latency

6.2.1 Leader Election

In the experiment of this section, we compare the leader election latency of Raft, KRaft [41] and V-Raft as the number of nodes increases. Raft uses voting for leader elections. To begin an election, a follower is converted to candidate when the local timer times out, and then the candidate broadcasts a request vote message to other nodes. The timeout time is usually set to [150–300] ms, and the broadcast time is usually set to 15 ms. When the candidate receives votes from a majority of the nodes, it is elected as the leader. KRaft implements leader election based on Kademlia protocol. It establishes a DHT topology for the nodes in the system and prioritizes the node with the least latency as a candidate node to send a request vote message. While V-Raft uses a VRFs-based sortition algorithm to elect leader, the experimental results are shown in Fig. 6. When the number of nodes in the consortium blockchain

is small, the leader election latency of KRaft and V-Raft are close and both are significantly lower than Raft. In different cases where the number of nodes in the consortium blockchain is incremented from 50 to 250, the leader election latency of V-Raft is significantly lower than Raft. In Raft, the leader election latency will increase as the number of nodes in the consortium blockchain increases. While the latency of V-Raft is only related to the sortition threshold λ and not to the number of nodes. Therefore, V-Raft has higher efficiency and better scalability.

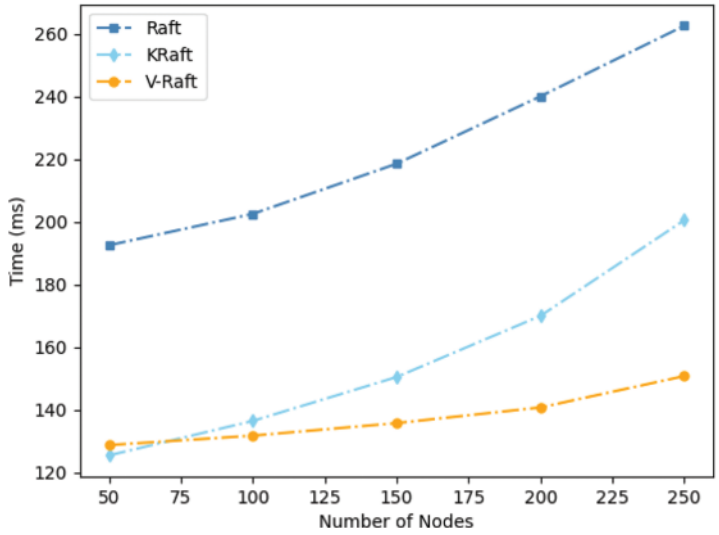

Figure 6: Leader election latency

6.2.2 Consensus

In Raft, When the leader receives the client request, it will forward the request to the follower. When the leader receives a reply from more than half of the followers, it indicates that the nodes in the consortium blockchain have reached a consensus on the request. Instead of a single leader forwarding the logs in Raft, in KRaft the leader and the candidate jointly forward the logs. While V-Raft adds committee state, the leader only needs to reach consensus with fewer and fixed committee members. The experimental results are shown in Fig. 7. For the speed of nodes reaching consensus, V-Raft is higher than KRaft and Raft.

6.2.3 Replacement

In Raft and KRaft, leader election is restarted if the leader crashes. Due to the committee being elected in advance in V-Raft, the V-Raft will select a qualified node from the committee to replace it. The experimental results of the three algorithms are shown in Fig. 8. Obviously, V-Raft spends less time than Raft and KRaft.

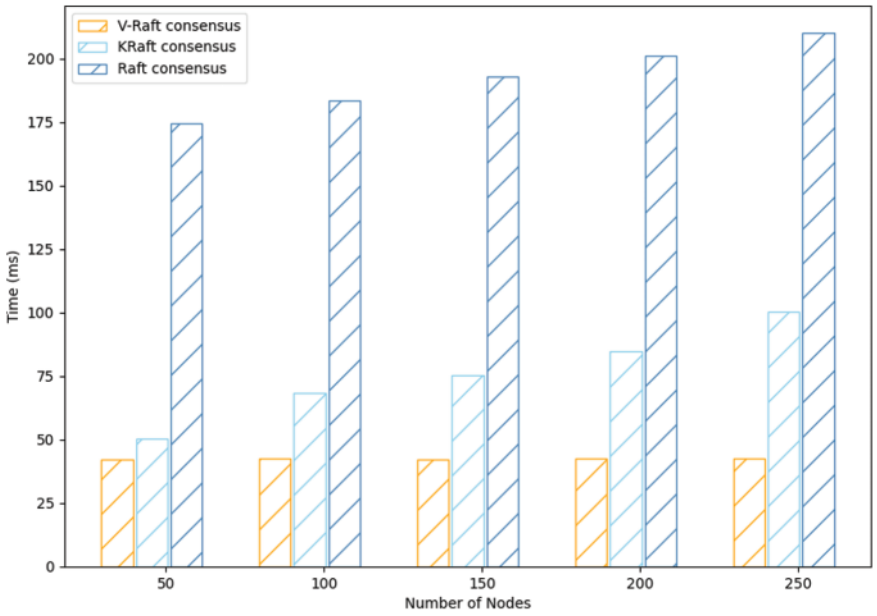

Figure 7: Nodes replacement and consensus latency

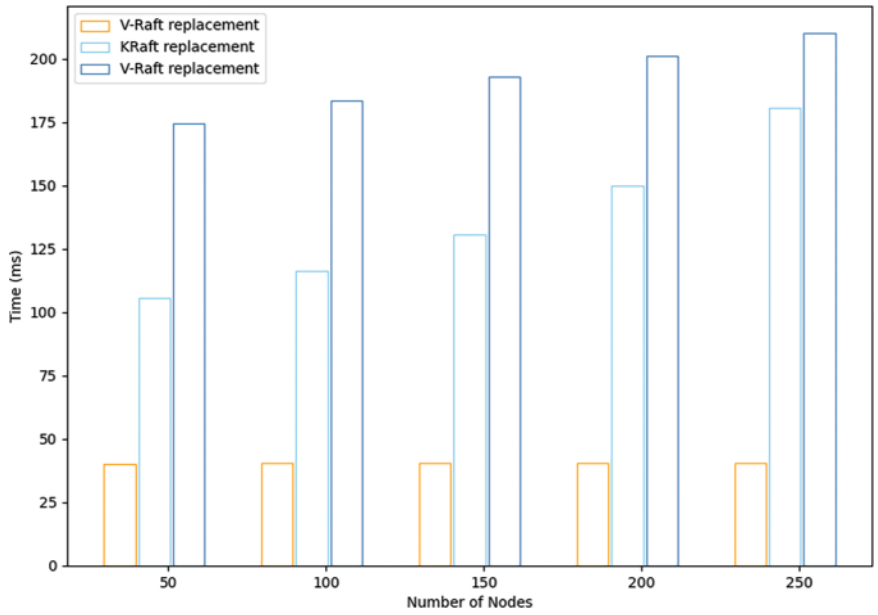

Figure 8: Nodes replacement and consensus latency

7 Conclusions

In this article, we propose a VFL architecture based on consortium blockchain for data sharing in MEC. Firstly, we improve the popular consensus algorithm in the consortium blockchain, Raft, and propose a new consensus algorithm based on VRFs, V-Raft. In addition, we compare the performance of the two algorithms through experiments. The experimental results show that compared with Raft,

the V-Raft has higher efficiency and better scalability in leader election, leader replacement and log replication. Also, we apply secret sharing to ensure that even if the leader crashes, the new leader can still recover the private key and continue to complete the joint model training. Therefore, the consortium blockchain can act as a trusted third party to assist the data demander and owner to realize the data sharing by VFL. The circulation and sharing of data can effectively promote the development of MEC. This article is carried out under the setting of honest-but-curious nodes and future work will consider the case where there are byzantine nodes in the consortium blockchain.

Funding Statement: This research is funded by the National Natural Science Foundation (61962009) and the National Natural Science Foundation (62202118). Top Technology Talent Project from Guizhou Education Department (Qianjiao ji [2022]073) and Foundation of Guangxi Key Laboratory of Cryptography and Information Security (GCIS202118).

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

References

1. Mao, Y., You, C., Zhang, J., Huang, K., Letaief, K. B. (2017). A survey on mobile edge computing: The communication perspective. *IEEE Communications Surveys & Tutorials, 19(4),* 2322–2358. https://doi.org/10.1109/COMST.2017.2745201
2. Sandhu, A. K. (2022). Big data with cloud computing: Discussions and challenges. *Big Data Mining and Analytics, 5(1),* 32–40. https://doi.org/10.26599/BDMA.2021.9020016
3. Noppen, A. V. (2021). Creating technology worthy of the human spirit. *Journal of Social Computing, 2(4),* 309–322. https://doi.org/10.23919/JSC.2021.0024
4. Konecný, J., McMahan, H. B., Ramage, D., Richtárik, P. (2016). Federated optimization: Distributed machine learning for on-device intelligence. arXiv:1610.02527.
5. Konecný, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T. et al. (2016). Federated learning: Strategies for improving communication efficiency. arXiv:1610.05492.
6. McMahan, H. B., Moore, E., Ramage, D., Arcas, B. A. (2016). Federated learning of deep networks using model averaging. arXiv:1602.05629.
7. Yang, Q., Liu, Y., Chen, T., Tong, Y. (2019). Federated machine learning: Concept and applications. arXiv:1902.04885.
8. Kim, M., Song, Y., Wang, S., Xia, Y., Jiang, X. (2018). Secure logistic regression based on homomorphic encryption: Design and evaluation. *JMIR Medical Informatics, 6(2),* e19. https://doi.org/10.2196/medinform.8805
9. Acar, A., Aksu, H., Uluagac, A. S., Conti, M. (2017). A survey on homomorphic encryption schemes: Theory and implementation. arXiv:1704.03578.
10. Chen, Y., Dong, S., Li, T., Wang, Y., Zhou, H. (2021). Dynamic multi-key fhe in asymmetric key setting from lwe. *IEEE Transactions on Information Forensics and Security, 16,* 5239–5249. https://doi.org/10.1109/TIFS.2021.3127023
11. Kosba, A. E., Miller, A. K., Shi, E., Wen, Z., Papamanthou, C. (2016). Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. *2016 IEEE Symposium on Security and Privacy (SP)*, pp. 839–858. San Jose, CA, USA.
12. Wang, W., Wang, Y., Duan, P., Liu, T., Tong, X. et al. (2022). A triple real-time trajectory privacy protection mechanism based on edge computing and blockchain in mobile crowdsourcing. *IEEE Transactions on Mobile Computing,* 1–18. https://doi.org/10.1109/TMC.2022.3187047

13. Sun, Z., Wang, Y., Cai, Z., Liu, T., Tong, X. et al. (2021). A two-stage privacy protection mechanism based on blockchain in mobile crowdsourcing. *International Journal of Intelligent Systems, 36(5),* 2058–2080. https://doi.org/10.1002/int.22371
14. Li, F., Yu, X., Ge, R., Wang, Y., Cui, Y. et al. (2022). BCSE: Blockchain-based trusted service evaluation model over big data. *Big Data Mining and Analytics, 5(1),* 1–14. https://doi.org./10.26599/BDMA.2020.9020028
15. Liu, G., Wu, J., Wang, T. (2021). Blockchain-enabled fog resource access and granting. *Intelligent and Converged Networks, 2(2),* 108–114. https://doi.org/10.23919/ICN.2021.0009
16. Ongaro, D., Ousterhout, J. K. (2014). In search of an understandable consensus algorithm. *USENIX Annual Technical Conference*, Philadelphia, PA, USA.
17. Qi, L., Hu, C., Zhang, X., Khosravi, M. R., Sharma, S. et al. (2021). Privacy-aware data fusion and prediction with spatial-temporal context for smart city industrial environment. *IEEE Transactions on Industrial Informatics, 17(6),* 4159–4167. https://doi.org/10.1109/TII.2020.3012157
18. Chen, Y., Sun, J., Yang, Y., Li, T., Niu, X. et al. (2021). PSSPR: A source location privacy protection scheme based on sector phantom routing in wsns. *International Journal of Intelligent Systems, 37(2),* 1204–1221. https://doi.org/10.1002/int.22666
19. Wang, Y., Li, T., Liu, M., Li, C., Wang, H. (2022). STSIIML: Study on token shuffling under incomplete information based on machine learning. *International Journal of Intelligent Systems, 37(12),* 11078–11100. https://doi.org/10.1002/int.23033
20. Pang, J., Huang, Y., Xie, Z., Li, J., Cai, Z. (2021). Collaborative city digital twin for the COVID-19 pandemic: A federated learning solution. *Tsinghua Science and Technology, 26(5),* 759–771. https://doi.org/10.26599/TST.2021.9010026
21. Zhang, W., Li, Z., Chen, X. (2021). Quality-aware user recruitment based on federated learning in mobile crowd sensing. *Tsinghua Science & Technology, 26(6),* 869–877. https://doi.org/10.26599/TST.2020.9010046
22. Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M. et al. (2021). Advances and open problems in federated learning. arXiv:1912.04977.
23. Qi, L., Lin, W., Zhang, X., Dou, W., Xu, X. et al. (2022). A correlation graph based approach for personalized and compatible web APIs recommendation in mobile APP development. *IEEE Transactions on Knowledge and Data Engineering,* 1. https://doi.org/10.1109/TKDE.2022.3168611
24. Yuan, X., Chen, J., Yang, J., Zhang, N., Yang, T. et al. (2022). FedSTN: Graph representation driven federated learning for edge computing enabled urban traffic flow prediction. *IEEE Transactions on Intelligent Transportation Systems,* 1–11. https://doi.org/10.1109/TITS.2022.3157056
25. Teimoori, Z., Yassine, A., Hossain, M. S. (2022). A secure cloudlet-based charging station recommendation for electric vehicles empowered by federated learning. *IEEE Transactions on Industrial Informatics, 2(4),* 277–294. https://doi.org/10.1109/TII.2022.3148997
26. Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. https://doi.org/10.2139/ssrn.3440802
27. Wang, Z., Zhang, F., Yu, Q., Qin, T. (2021). Blockchain-envisioned unmanned aerial vehicle communications in space-air-ground integrated network: A review. *Intelligent and Converged Networks, 2(4),* 277–294. https://doi.org/10.23919/ICN.2021.0018
28. Ren, J., Li, J., Liu, H., Qin, T. (2022). Task offloading strategy with emergency handling and blockchain security in sdn-empowered and fog-assisted healthcare iot. *Tsinghua Science and Technology, 27(4),* 760–776. https://doi.org/10.26599/TST.2021.9010046
29. Chi, C., Wang, Y., Tong, X., Siddula, M., Cai, Z. (2022). Game theory in Internet of Things: A survey. *IEEE Internet of Things Journal, 9(14),* 12125–12146. https://doi.org/10.1109/JIOT.2021.3133669
30. Li, F., Wang, Y., Gao, Y., Tong, X., Jiang, N. et al. (2022). Three-party evolutionary game model of stakeholders in mobile crowdsourcing. *IEEE Transactions on Computational Social Systems, 9(4),* 974–985. https://doi.org/10.1109/TCSS.2021.3135427

31. Li, T., Chen, Y., Wang, Y., Wang, Y., Zhao, M. et al. (2020). Rational protocols and attacks in blockchain system. *Security and Communication Networks, 2020,* 8839047. https://doi.org/10.1155/2020/8839047
32. Li, T., Wang, Z., Yang, G., Cui, Y., Chen, Y. et al. (2021). Semi-selfish mining based on hidden markov decision process. *International Journal of Intelligent Systems, 36(7),* 3596–3612. https://doi.org/10.1002/int.22428
33. Wang, Y., Cai, Z., Hui Zhan, Z., Zhao, B., Tong, X. et al. (2020). Walrasian equilibrium-based multiobjective optimization for task allocation in mobile crowdsourcing. *IEEE Transactions on Computational Social Systems, 7(4),* 1033–1046. https://doi.org/10.1109/TCSS.2020.2995760
34. Li, T., Wang, Z., Chen, Y., Li, C., Jia, Y. et al. (2021). Is semi-selfish mining available without being detected? *International Journal of Intelligent Systems, 37(12),* 10576–10597. https://doi.org/10.1002/int.22656
35. Gervais, A., Karame, G. O., Wüst, K., Glykantzis, V., Ritzdorf, H. et al. (2016). On the security and performance of proof of work blockchains. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, Vienna, Austria.
36. Kerber, T., Kiayias, A., Kohlweiss, M., Zikas, V. (2019). Ouroboros crypsinous. *Privacy-Preserving Proof-of-Stake//2019 IEEE Symposium on Security and Privacy (SP)*, San Francisco, CA, USA.
37. Gilad, Y., Hemo, R., Micali, S., Vlachos, G., Zeldovich, N. (2017). Algorand: Scaling byzantine agreements for cryptocurrencies. *Proceedings of the 26th Symposium on Operating Systems Principles*, Monterey, California, USA.
38. Chen, J., Micali, S. (2019). Algorand: A secure and efficient distributed ledger. *Theoretical Computer Science, 777(1),* 155–183. https://doi.org/10.1016/j.tcs.2019.02.001
39. Hanke, T., Movahedi, M., Williams, D. (2018). DFINITY technology overview series, consensus system. arXiv:1805.04548.
40. Shamir, A. (1979). How to share a secret. *Communications of the ACM, 22(11),* 612–613. https://doi.org/10.1145/359168.359176
41. Wang, R., Zhang, L., Xu, Q., Zhou, H. (2019). K-bucket based raft-like consensus algorithm for permissioned blockchain. *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 996–999. Tianjin, China.