HEBFL: Privacy-Preserving Blockchain-Based Federated Learning via Homomorphic Encryption

Abstract. Federated learning is a decentralized machine learning method, which can effectively protect the privacy of participants. There are many security problems in traditional federated learning. For example, the intermediate gradients uploaded by participants may cause serious privacy leakage. Attackers can damage the accuracy of the global model by uploading malicious models. Besides, stragglers may reduce the accuracy of the global model because of their stale model. To solve these problems, this paper proposes a blockchain-based privacy-preserving framework HEBFL. We use the model weights encryption mechanism based on partitioning aggregation strategy to protect each participant's privacy, propose the malicious-lazy node detecting and disposing mechanism to identify and dispose of malicious nodes and lazy nodes, and design the optimized asynchronous mechanism to mitigate the negative impact on the global model from stragglers. Finally, we implement a prototype of HEBFL and conduct some experiments on several real datasets. The privacy analysis and the results of the experiments show that our proposed framework can achieve better performance in terms of accuracy, robustness, and privacy.

Keywords: Federated learning (FL), Blockchain, Privacy-preserving, Homomorphic encryption.

1 INTRODUCTION

With the advent of the era of big data, machine learning has made exciting achievements in many fields. In order to achieve even higher accuracy, a huge amount of data must be fed to deep learning models, which leads to the constraints of relevant laws and regulations. Nowadays, data integration is extremely difficult, and 'data island' is forming [1]. Therefore, how to design a framework to enable people to share data safely is an important issue. In 2016, federated learning [2] emerged, which solved 'data island'. The core idea of federated learning is to conduct collaborative training by the way of parameter aggregation instead of data aggregation. The training data is stored at each of the training nodes in FL. Each node trains a model (the same one as maintained at the central server) on its local data individually, and uploads intermediate gradients to the central server. After that, the central server aggregates those gradients and sends them back accordingly. After that, each training node updates its local model and continues the training. This process repeats until the accuracy is more than the pre-specified threshold [3].

Federated learning has the following advantages: (1) Multiple parties can train the model without sharing any data, which is conforming to the laws, regulations, and industry norms; (2) Using the way of parameter aggregation to train the global model can achieve the same accuracy as using the way of data aggregation [4]. However,

traditional FL also has many disadvantages: (1) Single-point failure [5-7]. Once the central server fails, the whole system will be broken down. In this paper, we using blockchain to replace central server to solve single-point failure; (2) Privacy disclosure of intermediate parameters [8-13]. Some researchers have pointed out that according to local gradients, important information related to the original data can be inferred [10]; (3) Malicious participants [11],[14-16]. Dishonest participants can poison the global model by updating an exquisitely designed failure model; (4) Resource and data heterogeneity [17-20]. Due to highly heterogeneous CPU, memory, and network resources, there will be some stragglers in FL, which will slow down the speed of aggregation.

In this paper, we solve the aforementioned problems by considering a privacy-enhanced blockchain federated learning framework, named HEBFL. It proposes the model weights encryption mechanism based on partitioning aggregation strategy; proposes malicious-lazy node detecting and disposing mechanism as well as the optimized asynchronous mechanism. Compared to existing blockchain FL works, The major contributions of this paper are summarized as follows:

1. We use Paillier homomorphic encryption technology to encrypt weights and the partitioning aggregation strategy to avoid the exposure of privacy during the transmission process and aggregation process.
2. We propose the malicious-lazy node detecting and disposing mechanism to protect the damage from anomalous nodes.
3. We propose the optimized asynchronous mechanism to detect and update stale models.

The rest of this paper is organized as follows. Section II gives an introduction to the related work; Section III is the background knowledge of homomorphic encryption, smart contract, and IPFS; Section IV will introduce the HEBFL framework proposed in this paper and its specific details; Section V gives the experimental results; section VI gives the conclusion and future work.

2 PRELIMINARIES

Currently, more and more researchers are focusing their research on various problems of traditional FL. We will describe the existing related work from three aspects: the exposure of gradient information [8-13], anomalous nodes [11],[14-16], and resource heterogeneity [17-20].

For a long time, people believe that the gradient includes less information, so it can be shared safely. However, some researches have shown that the gradients can be used to speculate on the original data or even reconstruct private training data. In order to protect privacy in FL, Bonawitz et al. [8] used secret sharing and symmetrical encryption mechanism to make the central server only can get the aggregation result of gradients, instead of the local gradients of each training node. However, when the number of participants is large and the network is disconnected, security aggregation will fail frequently. Lu et al. [9] implemented the Laplace mechanism of differential privacy to add noise to the local dataset, Yang Zhao et al. [11] extracted features from the local

data and add noise to them in the middle layer of neural network. Moreover, Geyer R C et al. [12] used the Laplace mechanism of differential privacy to add noise to gradients, Truex S et al. [13] added noise to model weights to protect privacy to protect privacy. But the research of Le, T. P et al. [10] and Hitaj et al.[21] have shown that the original data can be inferred from the model also added noise by using Generative Adversarial Network(GAN). Although differential privacy can protect privacy to a certain extent, it is still possible to infer some information from the original data. Besides, differential privacy is helpless for Model Inversion Attack [22] and would affect the convergence and accuracy of the model due to the noises. In the field of deep learning, accuracy is very important. However, differential privacy cannot achieve high privacy protection without compromising accuracy.

Many previous papers in federated learning assume that all the participants are honest and trustworthy, but this assumption is difficult to hold in practice. Malicious nods can upload malicious model parameters to the server to damage the entire federated learning. Therefore, it is crucial for federated learning to detect failure models uploaded by anomalous nodes and prevent them from affecting the global model. To solve this problem, Toyoda and Zhang [14] introduced a competitive model update method. Before the next iteration of local training, training nodes select the top K model to update their global models. The rewards of the training node are determined by the voting, so as to encourage effective training updates. J. Kang et al. [15] proposed a consensus mechanism called Proof of Verification (PoV). They added the model quality assessment process into the consensus mechanism to identify and remove malicious models. Yang Zhao et al. [11] combined Multi-KRUM and a reputation-based incentive protocol to delete malicious updates and calculate reputation value. Kang et al. [16] used a multi-weighted subjective logic model to calculate the reputation value of each candidate who is preparing to participate in FL. Only candidates with reputation values greater than the threshold are allowed to participate in training. Although all these methods have improved the quality of the updates to some extent, if attackers adopt a method of maintaining accuracy thresholds and continuously submitting slight malicious models, the accuracy detection method will no longer be reliable. Moreover, lazy models cannot be identified through accuracy detection methods.

Due to heterogeneous hardware resources (CPU, memory, and network resources), the training speed and transmission speed of each device in FL are different. To solve this problem, Li et al. [17] held that only the top K nodes in the active state should be selected for local training. Li T et al. [18] formulated different local epochs for different clients according to their resource conditions. Chai Z et al. [20] assigned each node to different layers according to their different resource conditions. Thus, they applied synchronous intra-layer and asynchronous cross-layer training method. Mingrui Cao et al. [19] used DAG blockchain technology, in which the nodes have no need to wait for other nodes when processing new transactions. Although the above mechanisms have solved resource heterogeneity problems to varying degrees, they have not taken into consideration that extreme stale models of stragglers will still have adverse effects on the accuracy and convergence of the global model [23].

3 PRELIMINARIES

3.1 Smart Contract

Smart contract is a digital protocol deployed on the blockchain, which can flexibly embed various data and exchange information safely and efficiently. Besides, smart contract are visible and callable to all nodes in the blockchain [24].

In this paper, the following functions should be undertaken by the smart contract.

1. Data storage and information exchange. The training nodes should get the global model from the smart contract, and after training, they should upload the model to the smart contract.
2. Node authority control. In our malicious-lazy node detecting and disposing mechanism, the malicious and lazy nodes need to be invalidated. It will be carried out by the invalidation function.
3. Updating the model of the slow layer. The version-detecting function is responsible for judging the model version of each layer. Moreover, the replacing function is responsible for updating the global model of slow layers.

3.2 Paillier Homomorphic Cryptosystem

In this paper, we utilize additively homomorphic encryption-Paillier homomorphic cryptosystem[25] to guarantee safe aggregation in FL. The encryption and decryption process in this paper is shown below:

KeyGen: Choose two large primes (p, q) randomly to satisfy $gcd\big(pq, (p-1)(q-1)\big) = 1$. Then calculate the RSA modulus as $N = pq$ and Carmichael function as $\lambda = lcm(p-1, q-1)$, while lcm is the lowest common multiple and gcd is the greatest common divisor. Randomly choosing $g \in Z_{N^2}^*$ to satisfy $gcd\big(L\big(g^{\lambda} mod N^2\big), N\big) = 1$, Z indicates an integer, and the subscript indicates the number of elements in the integer set. Then, calculate the functions $L(x) = \frac{(x-1)}{N}$ and $\mu = \frac{1}{L\left(g^{\lambda} mod N^2\right)} mod\ N$. Thus, the public and private key pairs are shown as $(pk,\ sk) = \{(n, g)、(\lambda, \mu)\}$.

ENC: For plaintext m, randomly generate a number r such that $\gcd(r, n) = 1$, the ciphertext can be calculated as

$$c = (g^m \cdot r^n) mod\ N^2 \tag{1}$$

AGG: Given a set of ciphertext $c(i)$, all the ciphertexts satisfied with $c(1) * c(2) * \cdots * c(k) = Enc[m(1) + m(2) + \cdots + m(k)]$, where k is the number of ciphertexts.

DEC: The plaintext m can be calculated as

$$m = D(c) = \frac{L\left(g^{\lambda} mod N^2\right)}{L\left(c^{\lambda} mod N^2\right)} mod\ N \tag{2}$$

3.3 IPFS (InterPlanetary File System)

The blockchain has storage limits in smart contract (named gas), so the model parameters cannot be totally stored on the blockchain. Currently, the common solution is to introduce the InterPlanetary File System (IPFS) [26]. The InterPlanetary File System (IPFS) is a peer-to-peer distributed file system and has no single-point failure, or size limit, and not easy to be attacked. It can provide reliable and safe storage space for our framework. In our framework, model parameters encrypted will be stored in IPFS.

4 Design of HEBFL Framework

In this section, we will introduce the workflows of HEBFL framework and give the specific implementation details of the model weights encryption mechanism based on the partitioning aggregation strategy, the malicious-lazy node detecting and disposing mechanism, and the optimized asynchronous mechanism.

4.1 Model Weights Encryption Mechanism based on Partitioning Aggregation Strategy

When a complete target model is known, attackers can launch member inference attacks [27] and model inversion attacks to steal privacy. In our framework, we use the trusted node of the Proof of Authority consensus mechanism [28] — verifying nodes to aggregate model weights. However, if the verifying nodes defect or be hijacked in some extreme cases, there are still risks of privacy leakage during model parameter aggregation. Therefore, we propose the partitioning aggregation strategy. First, training nodes split the model into several parts and encrypt each part with different Paillier homomorphic keys so that different verifying nodes can aggregate different parts. The partitioning aggregation strategy is shown in Fig. 1.

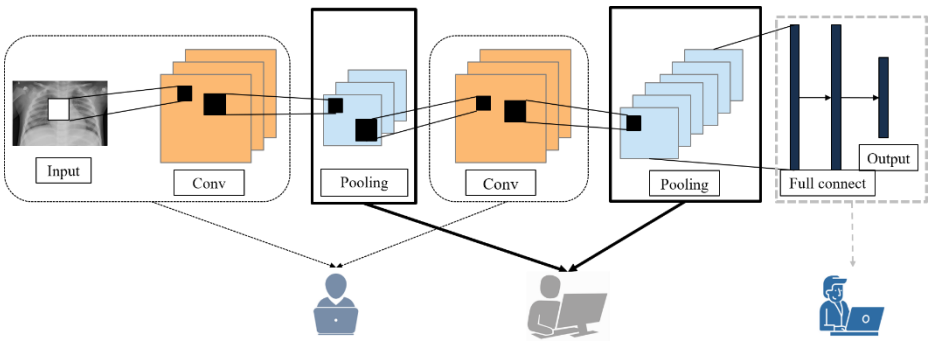


Fig. 1. The partitioning aggregation strategy

The model weights encryption mechanism based on partitioning aggregation strategy is described as follows:

The public key is used to encrypt model weights, and the private key is used to decrypt the model weight ciphertext. After an iteration, the training nodes split the model into several parts and use different public keys to encrypt model $Enc\left(\frac{1}{n}W_n\right)$, and then upload them to IPFS. Then different verifying nodes fetch the model parameters to execute the aggregation algorithm(Equation 3).

$$Enc(W_{global}) = Enc\left(\frac{1}{n}W_1 + \frac{1}{n}W_2 + \frac{1}{n}W_3 + \cdots + \frac{1}{n}W_n\right) = Enc\left(\frac{1}{n}W_1\right) * Enc\left(\frac{1}{n}W_2\right) * Enc\left(\frac{1}{n}W_3\right) * \cdots * Enc\left(\frac{1}{n}W_n\right) \quad (3)$$

After aggregation, training nodes retrieve the global model and then decrypt them using the private keys. Then training nodes splice each part of the model together and overwrite the local model.

Privacy Analysis. First, the Paillier homomorphic cryptosystem is proven to be indistinguishability against Chosen Plaintext Attack (CPA) [29] based on Decisional Composite Residuosity Problem, which means the ciphertexts will not leak any information about the plaintexts [30]. Second, due to the fact that member inference attacks and model reverse attacks can be launched under the condition that the target model is known, the partitioning aggregation strategy can prevent the verifying nodes from obtaining the complete target model. Third, due to the fact that federated learning task publishers do not publish the identity and the task of each node in the blockchain, verifying nodes do not know the identity of other nodes in the blockchain. Therefore, it is difficult for verifying nodes to collude. Finally, we use different keys to encrypt model parameters, privacy attackers are difficult to decrypt the complete model under the condition that they do not have all the private keys.

4.2 Malicious-lazy Node Detecting and Disposing Mechanism

Malicious nodes can perform poisoning attacks by incorporating wrong samples into local data, in order to deviate from the training direction of FL. Lazy nodes[8] refer to those harmful nodes that always copy old models from existing transactions and upload them as their own training results to plagiarize the training results. In order to ensure the security and fairness of FL, we propose the malicious-lazy node detecting and disposing mechanism. In normal training, the update frequency of sensitive neurons and insensitive neurons is different. Due to the fact that lazy nodes do not undergo real local training, they are difficult to distinguish the difference. Therefore, we record the weights update frequency of the penultimate layer of each training node and save it as the weight change frequency matrix. The weight change frequency matrix for lazy nodes is different from the normal training nodes.

We use Manhattan Distance to measure the difference of the weight change frequency matrix. For matrices A and B, the Manhattan distance calculation formula is as (Equation 4):

$$ManhattanDist(A, B) = \sum_{i=1}^{n}\sum_{j=1}^{n}|a_{ij} - b_{ij}| \quad (4)$$

As for lazy nodes, because lazy nodes do not intentionally disrupt federated learning and only plagiarize the training results, we can tolerate lazy behavior λ times. If a node uploads the lazy model over λ times, it will be invalidated.

As for malicious nodes, current studies use the accuracy detection method to identify. However, if attackers adopt a method of maintaining accuracy thresholds and continuously submitting slightly poisoned models, the accuracy detection method will no longer be reliable. Because malicious nodes have completely different learning objectives from normal training nodes, the Euclidean Distance between malicious models and normal models is larger than it is between normal models [31]. Therefore, we use the Euclidean Distance to detect the malicious model. We define the Euclidean Distance between models as (Equation 5):

$$Dis\ (M_1, M_2) = \sqrt{\sum (M_1 - M_2)^2} \tag{5}$$

As for normal distance and abnormal distance, we use the binary K-Means clustering algorithm to judge. We set the normal Euclidean Distance (got from pre-training) as the initial center of the normal class and set the maximum distance as the center of the abnormal class. Through several iterations, the algorithm will update the values of each clustering center until the final clustering result is obtained.

Algorithm 1 $h_i \in (h_1, \cdots, h_n)$ is the IPFS hash address of node i's model parameters in layer S, $m_i \in (m_1, \cdots, m_n)$ is the IPFS hash address of node i's weight change frequency matrix in layer S, n is the number of Layer S's participants, S' is a set of model parameters, S is one of the layers in HEBFL.

Verifier executes:
Get the layer S from smart contract
While *i*<=n **do**
    Fetch model parameters W_i by h_i and the weight change frequency matrix M_i by m_i
    Calculate the Manhattan distance MD_i between M_i and the other node
    if MD_i is abnormal **then**
        Drop W_i
        Add the lazy value of the node(which submitted the hash h_i)
    else
    Calculate the Euclidean distance D_i between W_i and last global model W_{global} of layer S
    if D_i is abnormal **then**
            Drop W_i
            Invalidate the node(which submitted hash h_i)
    else
        Put W_i in S'
Aggregate S' with $W_{global} = \sum_{i=1}^{n} \frac{1}{n} W_i$
Upload W_{global} to IPFS
Submit the hash address of W_{global} to smart contract

The specific description of the malicious-lazy node detecting and disposing mechanism is as follows:

First, verifying nodes calculate the Manhattan Distance between the weight change frequency matrix of each node and the other. If the Manhattan Distance is abnormal, we determine it as a lazy model and drop it. Second, if the Manhattan Distance is normal, verifying nodes will calculate the Euclidean distance between the model and the global model. If the Euclidean distance is abnormal, we determine that it is a malicious model and drop it.

The malicious-lazy node detecting and disposing algorithm is shown in algorithm 1. S is an asynchronous layer in our framework, *n* is the number of nodes in S, $(h_1, \cdots, h_n)$ is the collection of IPFS hash address of model parameters in layer S. $h_i \in (h_1, \cdots, h_n)$ is the IPFS hash address of node *i*. $(m_1, \cdots, m_n)$ is the collection of the weight change frequency matrix of all the nodes in layer S. $m_i \in (m_1, \cdots, m_n)$ is the weight change frequency matrix of node *i*. When the local training of nodes in layer S is completed, verifying node will retrieve the collection $(h_1, \cdots, h_n)$ and $(m_1, \cdots, m_n)$, and traverse the elements in these two collections to get the model parameters and the weight change frequency matrix (step 4). Verifying nodes will calculate the Manhattan distance MD_i between node *i* and the other node (step 5). If MD_i is abnormal, model parameters of node *i* will be dropped and the lazy value of the node will be increased (step 7-8). If MD_i is normal, verifying node will calculate the Euclidean Distance D_i between W_i and the global model W_{global} (step 10). If D_i is abnormal, model parameters of node *i* will be dropped and node *i* will be invalidated (step 12-13). The normal models will be stored in the collection S' and waiting for aggregation (step 15-18).

4.3 Optimized Asynchronous Mechanism

In large-scale FL, resource heterogeneity may cause stragglers. Stragglers refer to those training nodes that are extremely slow in training and maintain a stale model [20]. In FL, as we know, the quality of the global model depends on the quality of each local model, and the quality of the local model depends on whether the local training is sufficient. When the stale models are aggregated into the global model, the accuracy and convergence of the global model will be seriously affected. Therefore, we propose the optimized asynchronous mechanism to solve this problem. First, we assigned training nodes to different asynchronous layers according to their computing and network conditions. The nodes with superior CPU and GPU, high memory capacity, and the best communication status are assigned to the first layer (fastest layer), followed by the second layer, and the nodes with the worst conditions are assigned to the last layer. The training nodes aggregate synchronously within layers. The training nodes of different layers train asynchronously. Second, we introduce a delay tolerance value δ to control the backwardness of the slow layer and set the first layer as the fastest version layer. Third, the smart contract checks the difference in the model version between the fastest layer and the others regularly. If the backwardness is exceeded δ, the global model of this layer will be replaced with the newest global model of the fastest version layer.

The optimized asynchronous algorithm is shown in algorithm 2.

Algorithm 2 Layer A is the fastest layer in the system.

1：**Smart contract executes：**

2：Get the model version i of layer A

3：Get version j of the layer B

4：**if** i-j>δ **then**

5： Change the latest global model parameters $W_{global(B)}$ of B with the latest global model parameters $W_{global(A)}$ of A

As shown in algorithm 2, layer A is the fastest layer in our framework, and layer B is an asynchronous layer waiting for aggregation. After all the training results of layer B in this iteration are uploaded, the smart contract retrieves the version of layer B (step 2); After that, the smart contract compares the model version numbers of layer B and layer A to judge whether the backwardness is more than δ (step 4); If it is, the smart contract will replace the global model $W_{global(B)}$ of layer B with the latest global model $W_{global(A)}$ of layer A (step 5).

4.4 System Workflows

Fig. 2 shows an overview of our framework architecture. The framework consists of two kinds of participants: training node and verifying node.

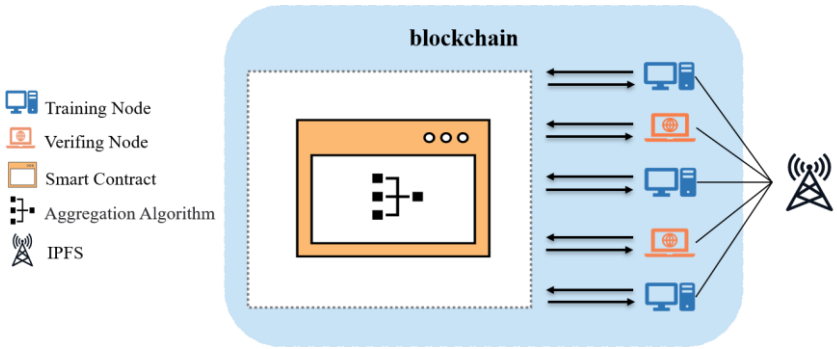


Fig. 2. Blockchain Federated Learning Framework

- Training node: train model with their local data and is also a node in Blockchain.
- Verifying nodes: verify the model parameters uploaded by training nodes, and judge whether they have poisoning attack or lazy attack. Besides, they can call the functions of the smart contract to dispose of anomalous nodes. It is also a node in Blockchain.

A round of training consists of several steps. This process will continue until the accuracy is more than the pre-specified threshold. The specific steps are as follows:

- Local training: training nodes get the initial model and start local training after registering in the smart contract. The local training uses Stochastic Gradient Descent (SGD) algorithm to minimize the loss function. After the training is over, the model (in the form of weight) is split into several parts and encrypted with homomorphic encryption. The encrypted model parameters and the weight change frequency matrix are uploaded to IPFS. The returned hash index from IPFS is encrypted with MD5, and finally submitted to the smart contract.
- Model aggregation: verifying nodes retrieve the IPFS hash index of model updates and the weight change frequency matrix, and then conduct the malicious-lazy node detecting and disposing mechanism to discriminate and dispose of anomalous nodes. Finally, the normal model parameters are aggregated with $W_{global} = \sum_{i=1}^{n} \frac{1}{n} W_i$. The aggregation results are uploaded to IPFS. The returned hash index from IPFS is encrypted with MD5, and finally submitted to the smart contract.
- Parameter update: training nodes retrieve the aggregation results and decrypt them with the private keys. Finally training nodes splice each part of the model together and overwrite the local model to continue the next iteration.

When federated learning finishes, verifying nodes will get the global models of all asynchronous layers to generate the final model. Because the update times of each asynchronous layer are different, so FedAvg aggregation strategy[2] doesn't fit our framework. To achieve unbiased and more balanced aggregation, we adjust the proportion assigned to each layer based on the update frequency:

Assuming there are *N* asynchronous layers, the frequency of updates from each layer is $T_1, T_2, \cdots, T_N$ respectively. The sum of $T_1, T_2, \cdots, T_N$ is $T_1 + T_2 + \cdots + T_N$. We define the final model as:

$$W_{final} = \sum_{i=1}^{N} \frac{T_N + 1 - i}{T} W_i \tag{6}$$

5 EXPERIMENT

In this section, we compare our malicious-lazy node detecting and disposing mechanism with the anomaly detection mechanism proposed by Yang Zhao et al. [11] to illustrate the robustness (against anomaly nodes) of our framework. And compare our optimized asynchronous mechanism with the asynchronous mechanism proposed by Chai Z et al. [20] to illustrate the better accuracy performance (against stale model) of our framework. The two experiments were conducted on the Chest X-ray dataset, Flowers dataset, CIFAR-10 dataset, and FashionMNIST dataset.

5.1 Dataset and Experimental Setup

Dataset. The X-ray dataset was selected from pediatric patients in the Women and Children's Medical Center, Guangzhou. There are 5,863 X-Ray images (JPEG) and 2 categories (Pneumonia/Normal) in X-ray dataset. The Flowers dataset contains 2040 images (JPEG) and 2 categories (dandelion/tulip) downloaded from Kaggle. The

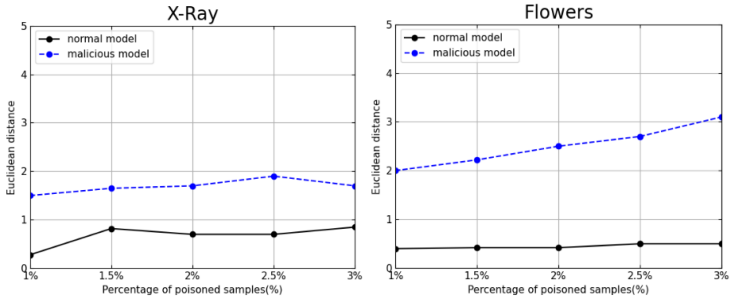
CIFAR-10 dataset [19] consists of 32x32x3 sized 60000 images (50000 for training, 10000 for testing), and 10 categories. FashionMNIST[20] consists 60,000 examples for training and 10,000 examples for test provided by Zalando(a German fashion technology company). Each example is a 28x28 grayscale clothing image, associated with a label from 10 classes. The MNIST[21] dataset(ten classes) as well as Cat and Dog dataset(two classes) are set to malicious datasets in our experiment.

Experimental Setup. Our training model is a Convolution Neural Network (CNN) with structure: Input → Conv1 → Maxpool → Conv2 → Maxpool → Conv3 → Fully Connected →Fully Connected →Output. The activation function is ReLU. The output of Conv1, Conv2, and Conv3 are 6, 16, and 16 respectively. After Conv1 and Conv2, there is a 2x2 max pooling.

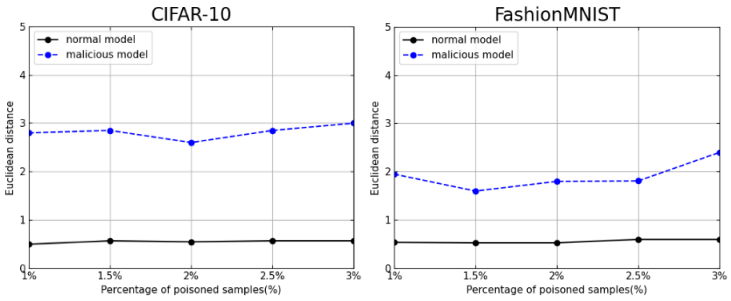
We use 12 computers to emulate the participants in our framework, which are equipped with 1.8-GHz AMD Ryzen 7 4800U with Radeon Graphics processors, 8 GB of RAM, and Windows 10 system. Besides, we build the deep learning environment with Python (version 3.9.6), and PyTorch (version1.11.0). Ethereum is built through Geth(version 1.8.26). Smart contracts are written and compiled by Remix. During the local training procedure, we set the local epoch E = 2, learning rate $\eta = 0.01$, momentum = 0.8, and batch size = 16 in two classes classification task. In ten classes classification task, we set local epoch E = 3, learning rate $\eta = 0.005$, momentum = 0.9, and batch size = 16.

5.2 Malicious-lazy Node Detecting and Disposing Mechanism

Firstly, we incorporate 1%, 1.5%, 2%, 2.5%, and 3% malicious samples into local data to train the model. Then we calculate the Euclidean Distance between them and the normal model. Moreover, we calculate the Euclidean Distance between two normal models for comparison. The results are shown in Fig. 3.



(a) (b)



(c) (d)

Fig. 3. Euclidean Distance between tow models

We set 12 nodes in our experiment and divide them into 9 training nodes as well as 3 verifying nodes. The 9 training nodes are assigned to three asynchronous layers (No.1,2,3 in the first layer, No.4,5,6 in the second layer, and No.7,8,9 in the third layer). To simulate the poison attack environment, we incorporate 2% malicious samples into local data of malicious node and let node No.3 continuously upload a malicious model in communication round 3 and beyond, node No.9 continuously upload a malicious model in communication round 1 and beyond; No.6 continuously upload a lazy model in communication round 2 and beyond. The node settings is shown in Table 1.

Table 1. NODE SETTINGS

Node Number	***1iteration***	***2 iteration***	***3 iteration***	***4 iteration and beyond***
1	normal	normal	normal	normal
2	normal	normal	normal(malicious)	normal(malicious)
3	normal	normal	malicious	malicious
4	normal	normal	normal	normal
5	normal	normal	normal	normal
6	normal	lazy	lazy	lazy
7	normal	normal	normal	normal
8	normal(malicious)	normal(malicious)	normal(malicious)	normal(malicious)
9	malicious	malicious	malicious	malicious
10				
11			verifying	
12				

The experiment results of IoT mechanism [11] are obtained under the same experimental settings as our paper.

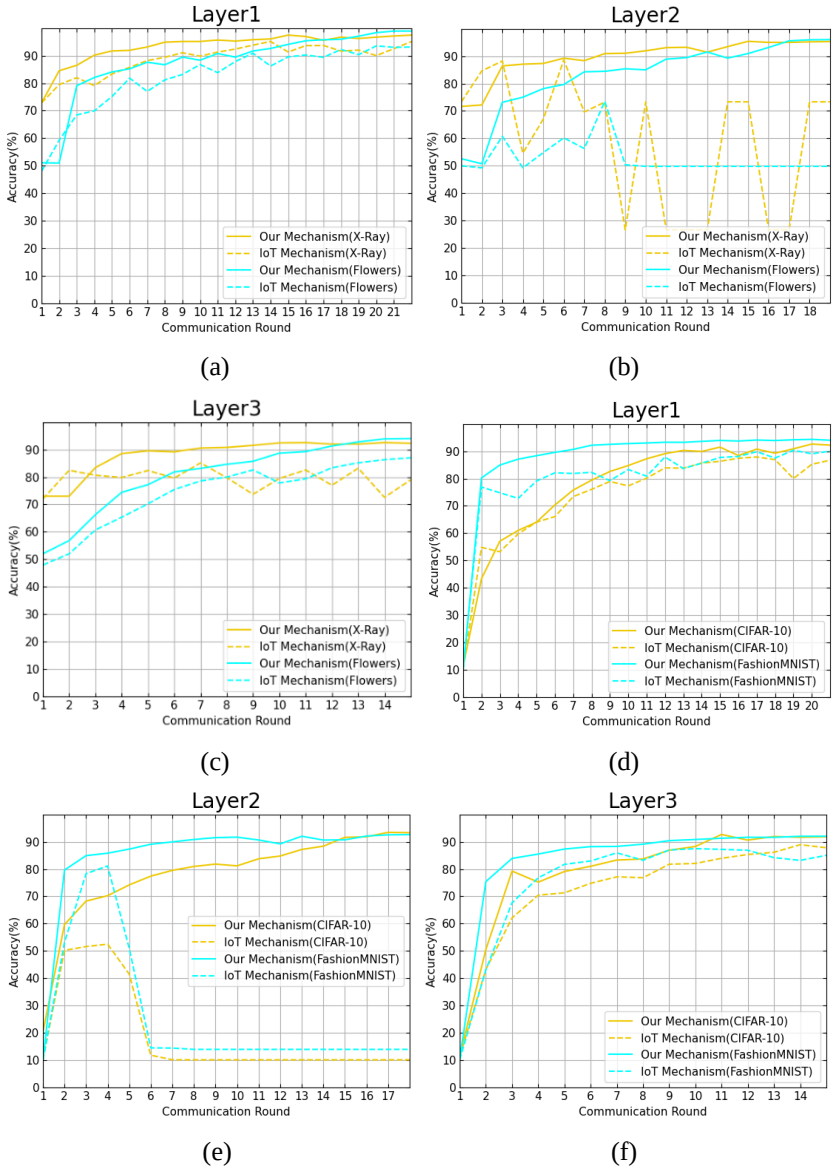


Fig. 4. The accuracy of the global model of different mechanisms under two and ten classes classification task(1)

Fig. 4 shows that the accuracy changes of the global model in this paper and IoT mechanism [11] on the X-ray, Flowers, CIFAR-10, and FashionMNIST dataset under the 2 malicious nodes and 1 lazy node. From Fig.4, it can be observed that although the global model in the first and the third layer are poisoned by the malicious node in communication round 3 and communication round 1 respectively, As the training goes on,

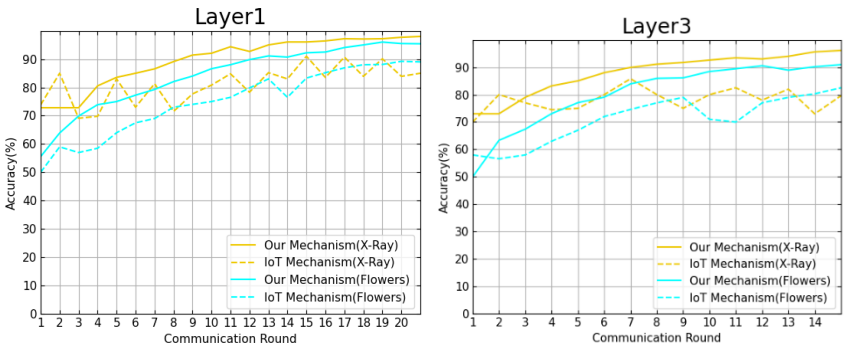
the accuracy of the global model of the two layers in this paper is unaffected. However, the accuracy of the global model in IoT mechanism is obviously affected. In the second layer, the convergence of the model in IoT mechanism was significantly damaged by the lazy node. In contrast, this paper is not affected by the lazy node.

Table 2. THE ACCURACY OF THE FINAL MODEL（1）

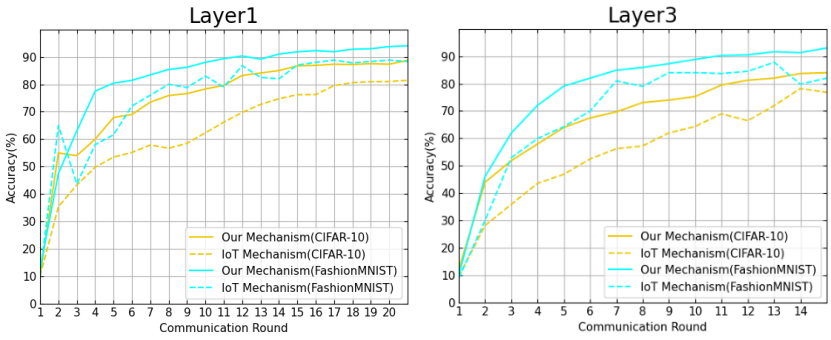
Dataset	Our mechanism	IoT mechanism[11]
Chest X-ray	94.886	91.691
Flowers	95.991	90.840
CIFAR-10	93.237	88.293
FashionMNIST	93.919	89.723

Table 2 shows the test accuracy of the final model of our paper and the IoT mechanism [11] on the X-ray, Flowers, CIFAR-10, and FashionMNIST dataset under 2 malicious nodes and 1 lazy node. As shown in Table 2, the accuracy of the final model of our mechanism is 3.195%, 5.151%, 4.944%, and 4.196% higher than that of IoT mechanism on the X-ray, Flowers, CIFAR-10, and FashionMNIST datasets respectively.

Furthermore, we increase the number of malicious nodes to 4: nodes No.2 and No.3 in the first layer continuously upload malicious models in communication round 3 and beyond; The No.8 and No.9 nodes in the third layer continuously upload malicious models in communication round 1 and beyond. Because the node settings of the second layer remain unchanged, the global model accuracy changes of the first and third layer is displayed.



(a) (b)



(c) (d)

Fig. 5. The accuracy of the global model of different mechanisms under two and ten classes classification task(2)

Fig. 5 shows the accuracy changes of the global model of this paper and IoT mechanism on the X-ray, Flowers, CIFAR-10, and FashionMNIST datasets under 4 malicious nodes. From Fig. 5, it can be observed that the accuracy of the global model in IoT mechanism is affected seriously more than the accuracy under 2 malicious nodes. However, the accuracy was not affected in this paper.

Table 3. THE ACCURACY OF THE FINAL MODEL (2)

Dataset	Our mechanism	IoT mechanism[11]
Chest X-ray	92.213	83.975
Flowers	92.885	85.326
CIFAR-10	86.960	78.293
FashionMNIST	90.016	79.453

Table 3 shows the test accuracy of the final model of our paper and the IoT mechanism on the X-ray, Flowers, CIFAR-10, and FashionMNIST datasets under 4 malicious nodes. As shown in Table 3, the accuracy of the final model of our paper is 8.238%, 7.559%, 8.037%, and 10.563% higher than that of IoT mechanism on the X-ray, Flowers, CIFAR-10, and FashionMNIST datasets respectively.

Due to the fact that the anomaly detection mechanism proposed by Yang Zhao et al. [11] use the accuracy detection method. It is not reliable to identify the slight malicious models and lazy models. Therefore, the malicious-lazy node detecting and disposing mechanism proposed by our paper performs better.

5.3 Optimized Asynchronous Mechanism

In this experiment, we compare our optimized asynchronous mechanism with the asynchronous mechanism proposed by Chai Z et al. [20] to illustrate better model accuracy performance(against the stale model) of our framework. In this experiment, delay

tolerance δ is set to 4. In addition, in order to simulate the resources heterogeneity environment, we add the time delay (10 minutes in the second layer and 25 minutes in the third layer)in each communication round to simulate the different training speed of different layers(layer 1 > Layer 2 > layer 3). The experiment result of FedAT algorithm [20] we obtained in the figure below is under the same experimental settings as this paper.

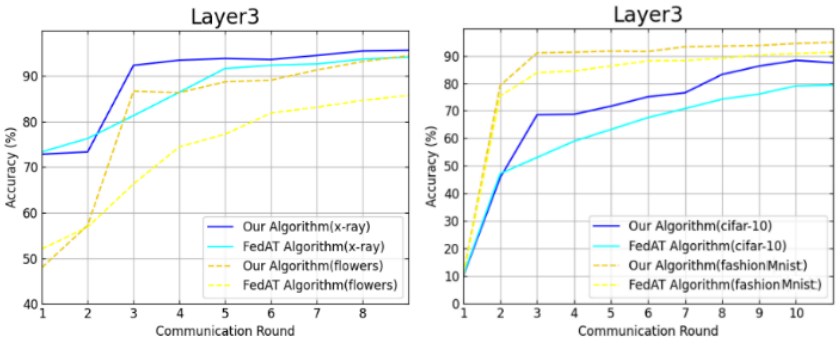


Fig. 6. The accuracy of different algorithms under two and ten classes classification task

During this experiment, the backwardness of the third layer exceeded the delay tolerance δ in communication round 3 and communication round 7. So the global model in the third layer in communication round 3 and communication round 7 was forced to update.

Fig. 6 shows the accuracy changes of the global model of this paper and FedAT [20] on the X-ray, Flowers, CIFAR-10, and FashionMNIST datasets. From Fig.6, we can observe that the accuracy of the global model in the third layer has been improved after being forced to update in communication round 3 and communication round 7. As the training goes on, it has maintained a good advantage. On the contrary, Because FedAT algorithm does not consider the negative impact of the stale model, its accuracy is lower than our paper.

Table 4. THE ACCURACY OF THE FINAL MODEL (3)

Dataset	Our Algorithm	FedAT Algorithm[20]
Chest X-ray	96.40	94.35
Flowers	95.991	92.746
CIFAR-10	87.98	84.66
FashionMNIST	94.919	93.242

Table 4 shows the accuracy of the final model of our paper and FedAT[20] in this experiment. As shown in Table 4, the accuracy of the final model of our paper is 2.05%, 3.245%, 3.32%, and 1.677% higher than that of FedAT on the X-ray, Flowers, CIFAR-

10, and FashionMNIST datasets respectively. We can be concluded that our optimized asynchronous mechanism performs better than FedAT (against the stale model).

6 Conclusion

This paper proposes a privacy-preserving blockchain Federated Learning framework, HEBFL. It uses Paillier homomorphic encryption technology to encrypt weights and the partitioning aggregation strategy to avoid the exposure of privacy during the transmission process and aggregation process. In addition, it proposes the malicious-lazy node detecting and disposing mechanism to detect and dispose of malicious nodes and lazy nodes. Finally, it proposes the optimized asynchronous mechanism to mitigate the negative impact of stragglers. The results show that our proposed framework achieves better performance in terms of privacy, accuracy, and robustness compared with existing frameworks.

However, this paper also has some shortcomings: Taking the key size of 1024 bits and the 67742 parameters of the model (CNN used in our experiment) as an example, it costs about 15 minutes to use Paillier homomorphic encryption to encrypt model weights in a communication round. We can conclude that the framework proposed by this paper needs high computing condition and time.

With the advent of the era of big data, paid deep learning arises at the historic moment. Nowadays, more and more data holders want to use their own data to make profits. Accordingly, ledger technology of blockchain can be used in blockchain federated learning to distribute the repay and reward of participants, which is also a great research interest of blockchain federated learning.

Acknowledgment.

References

1. Liang, Y., Guo, Y., Gong, Y., Luo, C., Zhan, J., Huang, Y.: Flbench: A benchmark suite for federated learning. In: Intelligent Computing and Block Chain: First BenchCouncil International Federated Conferences, FICC 2020, Qingdao, China, October 30–November 3, 2020, Revised Selected Papers 1. pp. 166–176. Springer(2021)
2. McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.:Communication-efficient learning of deep networks from decentralized data. In:Artificial intelligence and statistics. pp. 1273–1282. PMLR (2017)
3. Li, T., Sahu, A.K., Talwalkar, A., Smith, V.: Federated learning: Challenges, methods, and future directions. IEEE signal processing magazine 37(3), 50–60 (2020)
4. Jianming, Z., Qinnan, Z., Sheng, G., Qingyang, D., Liping, Y.: Privacy preserving and trustworthy federated learning model based on blockchain. Chinese Journal of Computers 44(12), 2464–2484 (2021)
5. Wu, X., Wang, Z., Zhao, J., Zhang, Y., Wu, Y.: Fedbc: blockchain-based decentralized federated learning. In: 2020 IEEE international conference on artificial intelligence and computer applications (ICAICA). pp. 217–221. IEEE (2020)

6. Hou, D., Zhang, J., Man, K.L., Ma, J., Peng, Z.: A systematic literature review of blockchain-based federated learning: Architectures, applications and issues. In: 2021 2nd Information Communication Technologies Conference (ICTC). pp.302–307. IEEE (2021)
7. Korkmaz, C., Kocas, H.E., Uysal, A., Masry, A., Ozkasap, O., Akgun, B.: Chainfl: Decentralized federated machine learning via blockchain. In: 2020 Second international conference on blockchain computing and applications (BCCA).pp. 140–146. IEEE (2020)
8. Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H.B., Patel, S., Ramage, D., Segal, A., Seth, K.: Practical secure aggregation for privacypreserving machine learning. In: proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 1175–1191 (2017)
9. Lu, Y., Huang, X., Dai, Y., Maharjan, S., Zhang, Y.: Blockchain and federated learning for privacy-preserved data sharing in industrial iot. IEEE Transactions on Industrial Informatics 16(6), 4177–4186 (2019).
10. Moriai, S.: Privacy-preserving deep learning via additively homomorphic encryption. In: 2019 IEEE 26th Symposium on Computer Arithmetic (ARITH).pp. 198–198. IEEE (2019).
11. Zhao, Y., Zhao, J., Jiang, L., Tan, R., Niyato, D., Li, Z., Lyu, L., Liu, Y.:Privacy-preserving blockchain-based federated learning for iot devices. IEEE Internet of Things Journal 8(3), 1817–1829 (2020)
12. Geyer, R.C., Klein, T., Nabi, M.: Differentially private federated learning: A client level perspective. arXiv preprint arXiv:1712.07557 (2017)
13. Truex, S., Liu, L., Chow, K.H., Gursoy, M.E., Wei, W.: Ldp-fed: Federated learning with local differential privacy. In: Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking. pp. 61–66 (2020)
14. Toyoda, K., Zhang, A.N.: Mechanism design for an incentive-aware blockchainenabled federated learning platform. In: 2019 IEEE international conference on big data (Big Data). pp. 395–403. IEEE (2019)
15. Kang, J., Xiong, Z., Jiang, C., Liu, Y., Guo, S., Zhang, Y., Niyato, D., Leung, C., Miao, C.: Scalable and communication-efficient decentralized federated edge learning with multi-blockchain framework. In: Blockchain and Trustworthy Systems: Second International Conference, BlockSys 2020, Dali, China, August 6–7, 2020, Revised Selected Papers 2. pp. 152–165. Springer (2020)
16. Kang, J., Xiong, Z., Niyato, D., Xie, S., Zhang, J.: Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory. IEEE Internet of Things Journal 6(6), 10700–10714 (2019)
17. Li, X., Huang, K., Yang, W., Wang, S., Zhang, Z.: On the convergence of fedavg on non-iid data. arXiv preprint arXiv:1907.02189 (2019)
18. Li, T. , et al.:Federated Optimization in Heterogeneous Networks. (2018).
19. Cao, M., Cao, B., Hong, W., Zhao, Z., Bai, X., Zhang, L.: Dag-fl: Direct a cyclicgraph-based blockchain empowers on-device federated learning. In: ICC 2021-IEEE International Conference on Communications. pp. 1–6. IEEE (2021)
20. Chai, Z., Chen, Y., Zhao, L., Cheng, Y., Rangwala, H.: Fedat: A communicationefficient federated learning method with asynchronous tiers under non-iid data.ArXivorg (2020)
21. Hitaj, B., Ateniese, G., Perez-Cruz, F.: Deep models under the gan: information leakage from collaborative deep learning. In: Proceedings of the 2017 ACM SIGSAC conference on computer and communications security. pp. 603–618 (2017)
22. Fredrikson, M., Jha, S., Ristenpart, T.: Model inversion attacks that exploit confidence information and basic countermeasures. In: Proceedings of the 22nd ACM SIGSAC conference on computer and communications security. pp. 1322–1333(2015)

23. Wu, W., He, L., Lin, W., Mao, R., Maple, C., Jarvis, S.: Safa: A semiasynchronous protocol for fast federated learning with low overhead. IEEE Transactions on Computers 70(5), 655–668 (2020)
24. HU Tian-Yuan, et al.:Contractual Security and Privacy Security of Smart Contract: A System Mapping Study.Chinese Journal of Computers 012(2021):044.
25. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Advances in Cryptology—EUROCRYPT'99: International Conference on the Theory and Application of Cryptographic Techniques Prague, Czech Republic, May 2–6, 1999 Proceedings 18. pp. 223–238. Springer (1999)
26. Benet, J.: Ipfs-content addressed, versioned, p2p file system. arXiv preprint arXiv:1407.3561 (2014)
27. Shokri, R., Stronati, M., Song, C., Shmatikov, V.: Membership inference attacks against machine learning models. In: 2017 IEEE symposium on security and privacy(SP). pp. 3–18. IEEE (2017)
28. Yang, J., Dai, J., Gooi, H.B., Nguyen, H.D., Paudel, A.: A proof-of-authority blockchain-based distributed control system for islanded microgrids. IEEE Transactions on Industrial Informatics 18(11), 8287–8297 (2022)
29. Goldreich, O.: Foundations of cryptography: Basic applications 2004 cambridge cambridge university press 10.1017. CBO9780511721656 Google Scholar Google Scholar Cross Ref Cross Ref
30. Wei, G., Jinyong, C., Lei, C.: Kdm-cca security based on dcr assumption. Computer Engineeringand Applications 53(12), 116–120 (2017)
31. Cao, D., Chang, S., Lin, Z., Liu, G., Sun, D.: Understanding distributed poisoning attack in federated learning. In: 2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS). pp. 233–239. IEEE (2019)
32. Krizhevsky, A., Hinton, G.: Convolutional deep belief networks on cifar-10. Unpublished manuscript 40(7), 1–9 (2010)
33. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747 (2017)
34. Deng, L.: The mnist database of handwritten digit images for machine learning research [best of the web]. IEEE signal processing magazine 29(6), 141–142 (2012)