FBFL: A Flexible Blockchain-based Federated Learning Framework in Mobile Edge Computing

Zhilin Wang, Shengyang Li
Department of Computer & Information Science, Indiana University-Purdue University Indianapolis, USA
Email: {wangzhil, sl137}@iu.edu

We will open-source a flexible and blockchain-based federated learning framework in mobile edge computing (FBFL). We first develop a federated learning (FL) framework that can handle both independent identically distributed (IID) and non-independent identically distributed (non-IID) data. Then, we use blockchain to connect the clients of federated learning and record the model updates generated in each round. After that, we deploy the blockchain-based federated learning framework on multiple MEC servers. Finally, we will test our framework using the MNIST dataset. The experiments we conduct include the impact of IID vs. non-IID datasets on the global model performance and the impact of poisoning attacks and backdoor attacks on the global model performance methods. To the best of our knowledge, FBFL will be the first blockchain-based federal learning framework that can be deployed in the mobile edge computing area. Our research will provide a convenient simulation environment for researchers and may also provide a basic framework for applications in the industry. We will publicize all the code of this project on GitHub and maintain and update it over time.

***Index Terms*—Blockchain, federated learning, mobile edge computing**

I. Introduction

With more data generated in our daily life, it has become a hot research topic to utilize these data to facilitate the development of artificial intelligence. The traditional approach transfers this data from local devices to a central server and performs machine learning uniformly. Such a framework has two serious drawbacks: 1) the data transmission imposes significant communication consumption and latency, and 2) the data privacy of local devices may be leaked. Fortunately, federated learning (FL) provides a possible solution.

FL was proposed by Google in 2016 and has been dramatically developed [1]. The basic idea of federated learning is that the local devices only need to upload the model weights obtained after training locally to the central server instead of transmitting their raw data to the central server. Moreover, the central server aggregates the received model weights to get the global model according to the aggregation algorithm. When this process has gone through several rounds, the central server can get a converged final model. FL has the following advantages: 1) reduces communication consumption and latency; 2) protects data privacy; 3) enables a broader range of data to be obtained so that the trained model can have better generalization ability.

Nevertheless, FL also has disadvantages. The first one is that FL may encounter the risk of a single point of failure caused by all local devices relying on a central server for model aggregation. The second is that FL lacks incentives to attract enough customers to participate in model training and to constrain their behavior through incentives. Third, FL faces many attacks, such as model poisoning attacks [2]–[4], data poisoning attacks [5]–[7], and backdoor attacks [8]–[10], which can seriously affect the performance of FL. Therefore, a robust FL framework needs to be designed.

There are many studies proposing the use of blockchain to assist FL [11]–[13]. Blockchain was proposed in 2008, and it is essentially a distributed ledger with properties such as record immutability, anonymity, and decentralization. Deploying FL on the blockchain can solve many of the challenges faced by FL. For example, blockchain can solve the problem of a single point of failure because blockchain is a network of multiple nodes, and the failure of one node will not affect the operation of the framework. The blockchain is decentralized, which also means that all nodes may be able to take on the role of model aggregation. In addition, the blockchain can provide rewards to the nodes involved in the work, which are determined by the consensus mechanism on the blockchain [14]–[16].

Although blockchain can bring robustness to FL, there is still a problem with where to deploy blockchain-based FL. Traditionally, multiple cloud servers are connected via blockchain, and local devices communicate with the cloud servers to form the FL framework. However, this approach is not very realistic since local devices may be mobile and distributed over a wide area. Thus cloud servers may not be able to accommodate such a scenario. In addition, local devices often do not have enough computing power for FL training. Therefore, some studies have proposed to deploy blockchain-based FL on mobile edge computing (MEC) servers [17]–[20]. It is a flexible framework where all devices with strong computing power and communication resources can become MEC servers. These servers are close to local devices to achieve broad coverage. Local devices only need to upload their data to the neighboring MEC servers to perform FL tasks, which can solve the problems of insufficient resources and wide geographical distribution.

Most current research on deploying blockchain-based FL on MEC servers focuses on theoretical studies, and there is no open-source framework. Therefore, in this paper, we will fill this gap. We develop a flexible blockchain-based FL framework on MEC based on the existing research. We have developed the FL and blockchain frameworks separately and integrated them effectively. We provide many directly callable interfaces for implementing the functionality in FL and blockchain, and researchers can design their approaches

as needed. Our work will facilitate researchers to perform simulation experiments. In summary, our main contributions are as follows.

- We open-source a blockchain-based FL framework deployed on MEC servers, which provides many interfaces and flexibility.
- Our framework can be used for both FL and blockchain research or individually.
- We illustrate the effectiveness and robustness of our framework through a large number of experiments.

The rest of this paper is organized as follows. We describe the system model design in Section II. Then, in Section III, we introduce the functions provided in our work. We conduct numerous experiments in Section IV. Finally, we conclude this paper in Section V.

II. System Model

In this section, we will introduce FBFL from the perspective of system design. We first describe the FL and blockchain systems and then discuss how these two systems work together on MEC.

A. Federated Learning System

The Federated learning (FL) framework consists of various clients and a central server. These clients are individually connected to the central server with bidirectional communications. Compared with traditional machine learning approaches, the distinctive feature of FL is that it performs the learning task without the scarification of data providers' privacy. In other words, the data providers (clients) will train the learning models locally but upload the trained learning model updates instead of their sensitive raw data to the central server. The central server will be responsible for aggregating these model updates to an updated global model. Then the updated global model will be sent back to each client to begin another new FL iteration. Model update contributions from various clients (data source providers) also ensure the accuracy and generality of the aggregated global model.

B. Blockchain System

Blockchain is essentially a distributed database system made up of the nodes of a computer network. These nodes maintain the blockchain system and connect with each other using the P2P connection. Compared with the typical database system, the unique feature of blockchain is that the data is structured in a decentralized manner. A blockchain records information together, known as blocks that hold groups of information. Each block is linked to the previously filled block, forming a chain of data known as the blockchain.

One of the most frequent critiques of FL in academia is that FL suffers a single-point failure. Whatsmore, in such a framework, the central server, also known as the model owner, holds full authority over the local model updates and updated global models. Blockchain technologies serve as a promising solution for the decentralization of the FL system. The central server could be seamlessly replaced with a blockchain, and clients are connected to the nodes in the blockchain instead. The model owner could publish the learning task into the blockchain to begin the training process. Such a blockchain-based federated learning (BCFL) framework is formulated.

C. BCFL in MEC

Our proposed system framework could be summarized in the figure 1. We deploy BCFL on the MEC server. Specifically, we use blockchain to connect the MEC servers, while each server connects multiple local devices. We also refer to MEC servers in the blockchain as blockchain nodes. It is worth noting that the local devices could be mobile and connected with the MEC server through wireless connections. Each local device occupies a local database where it stores the collected data. Each local device uploads the collected raw data to the connected MEC server. Each MEC server will train the learning model locally and derives the local model updates using the raw data received from the connected local devices. Then all the MEC servers will exchange their local model updates and executes the consensus algorithm afterward. The winner determined by the consensus will have the right to generate the new block. Then winner blockchain node will aggregate all the local model updates to the updated global model and pack necessary information into the new block. Eventually, the new block will be broadcast to the other MEC servers. Each MEC server could retrieve the updated global model from the new block to begin a new iteration.

III. Functions of FBFL

We will briefly introduce the functions supported in our system in this section.

A. Aggregation Methods of FL

In this subsection, we will introduce some popular aggregation algorithms of FL.

1) FedAvg

Federated Averaging (FedAvg) is a commonly applied aggregation method of FL. The basic idea of FedAvg is that it considers the proportion of each client's data volume to all data as the weight of the updates and then aggregates the weighted updates as the updated global model. This approach can counteract the effects of some of the outlier values to a certain extent and treat each client's contribution relatively fairly.

2) Krum

Krum selects some model updates which are similar to the others as the new global model [21]. The benefit is that even if the selected models are from malicious attackers, the impact may be limited since it is similar to other local models that may come from benign customers.

3) Trim-mean

There are two steps for trim-mean. First, the server drops out the maximum and minimum updates, and then it computes the mean of the rest of the updates as the global model [22].

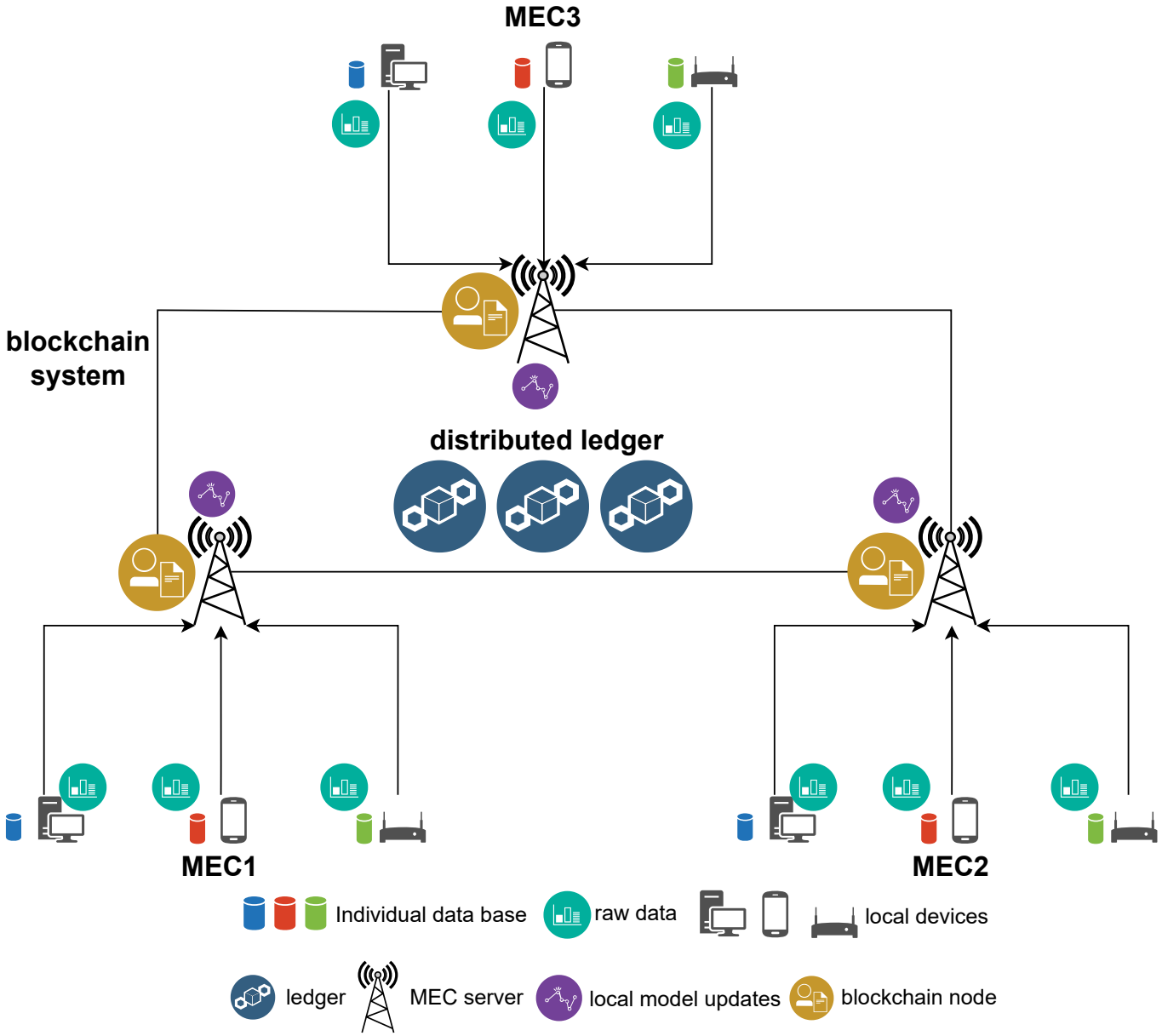


Fig. 1: The topology of FBFL.

B. Consensus Protocols

We will briefly introduce the consensus protocols implemented in FBFL.

1) Proof of Work

Proof of Work (PoW) [23], [24] is a process that requires nodes to try different random numbers (nonce) with their computing power to find the hash value that meets the requirement of computing power difficulty. It achieves the results by trying different random numbers until they find the one that meets the requirement. This process is called mining. The first node to find the correct nonce gets the right to bookkeeping. The node generates a new block and broadcasts it to other nodes. The other nodes will verify the block and accept it if it passes the verification. The consensus round is considered completed if the new block passes all the nodes' verification. Otherwise, it will reject the block and continue searching for a suitable nonce. So far, finding a nonce that meets the requirements is very difficult and requires nodes to consume a lot of computing power.

2) PBFT

Byzantine Fault Tolerance (BFT) is a core problem to be solved in the blockchain consensus algorithm. Practical Byzantine Fault Tolerance (PBFT) is generally used in federated chain scenarios, and it is a solution to BFT in the case of fewer consensus nodes. Miguel Castro and Barbara Liskov [25], [26] proposed the algorithm in 1999 to solve the problem of inefficiency of the previous Byzantine Fault Tolerance algorithm.

C. Data Distribution

Different data distributions can affect the performance of the FL model. We provide different types of data distributions: identical independent distributed (IID) data and non-identical independent distributed (non-IID) data.

1) IID Data

Existing machine learning tasks follow IID for training data by default. Common algorithms such as neural networks and deep learning generally assume that the data follows IID as part of their derivation. Such distribution can ensure that the trained model has good generalization ability.

2) Non-IID Data

However, sample data correlation is almost ubiquitous in the real world. The distribution of non-homogenous data/labels may also have different probability distributions, all of which follow Non-IID. In some scenarios, the direct application of existing machine learning algorithms to train the model based on Non-IID data still gives better training results due to its superiority. However, for some application scenarios, based on existing machine learning algorithms and frameworks, using Non-IID data for training can have unexpected adverse effects, such as low model accuracy and model failure to converge.

D. Malicious Attacks

Since FL requires multiple devices to be involved in training, there is no way to ensure that the devices are all honest, i.e., there is no guarantee that all of them will work as required. We will present some typical attacks against FL.

1) Poisoning Attacks

Poisoning attacks refer to malicious attackers controlling some clients and modifying the raw data or trained models on the clients, generally by adding noise or changing the directions of the updates to launch the attack.

- Model poisoning attacks is an attack that affects the performance of the global model by modifying the client's trained model. This attack generally occurs between the time the model is trained and the time it is uploaded.
- Data poisoning attacks are malicious attackers who degrade the performance of the global model by modifying the original data on the clients.

2) Backdoor Attacks

Backdoor attacks are more difficult to defend against than the attacks described above. A backdoor attack is caused by setting a backdoor in the training data, which makes it unrecognizable during the training process. Once it is used on a new dataset, the backdoor attack is triggered, resulting in the low generalization ability of the model.

IV. Experimental Results

A. Experiment Setup

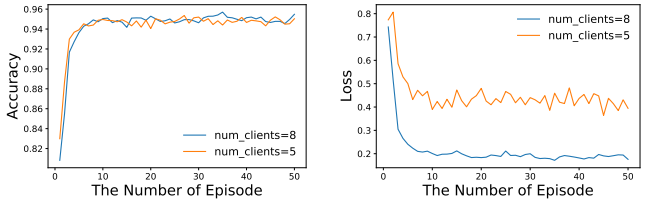
We have designed an FBFL framework from scratch. This framework consists of an FL system and a blockchain system; and FL system is based on TensorFlow, and the blockchain system is based on Flask. We have open-sourced parts of the code from this project on GitHub [27]. Since this project is based on our previous theoretical research and some parts of it are currently completed but not officially published, we cannot post the complete code.

We conduct the experiments using Python 3.9 in macOS 11.6 running on an Intel i7 processor with 32 GB RAM and 1 TB SSD. We use TensorFlow 2.8.0 to build a neural network and Flask 1.1.4 to support our blockchain system. To avoid statistical errors, all experimental results were obtained as the mean after five experiments.

B. Performance Evaluation

1) Experiments of FL

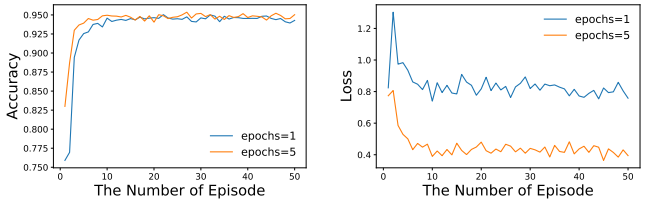
First, we analyze the performance of FL by changing the number of clients. The experimental results are shown in Fig. 2. In Fig. 2.a, we can see that the accuracy rate does not change much as the number of clients increases. This is because we are utilizing the IID data, and the classifier can still learn all the features well when the number of clients is small. However, in Fig. 2.b, the difference between the losses in the two cases is significant. This is because when the number of clients increases, the global model can better fit all the data and, therefore, can converge faster.



(a) Influence of Clients Number on Accuracy (b) Influence of Clients Number on Loss

Fig. 2: Influence of Clients Number on Accuracy and Loss.

Next, we explore the impact of the number of epochs each client trained locally on the FL performance. From Fig. 3, we can see that the model converges faster in terms of correctness and loss when the number of epochs is larger. This is because once clients use more epochs to train the model locally, the updates of each round will contain more information, thus improving the correct rate and speeding up the convergence.

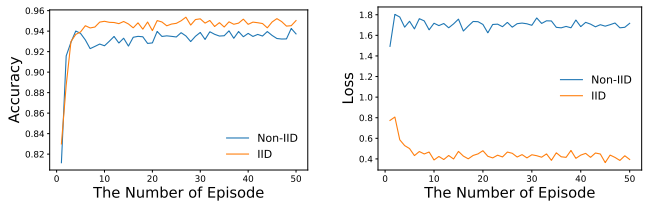


(a) Influence of Epochs on Accuracy (b) Influence of Epochs on Loss

Fig. 3: Influence of Epochs on Accuracy and loss.

We study the effect of different data distributions, i.e., non-IID and IID, on FL. We use two data types, and the result is shown in Fig. 4. We can see that the IDD data will obtain a higher correct rate, while the convergence of the loss will be more realistic. When the data is non-IID, the loss converges

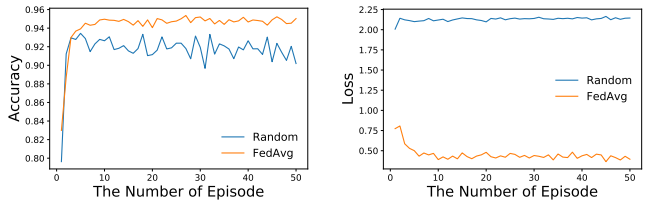
to a larger value. This is due to the characteristic of non-IID data, i.e., the labels are not uniformly distributed, which leads to a greater loss in the learned model.



(a) Influence of Data Distribution on Accuracy (b) Influence of Data Distribution on Loss

Fig. 4: Influence of Data Distribution on Accuracy and Loss.

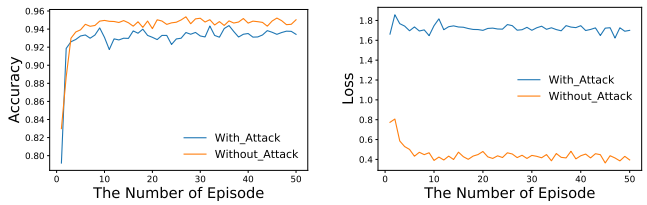
We also test the effectiveness of different aggregation methods. We design a method of aggregation that randomly selects the local model as the global model. From Fig. 5 we can see that FedAvg is effective in obtaining higher accuracy and convergence speed, while the method of randomly selecting the global model fails to converge.



(a) Influence of Aggregation Methods on Accuracy (b) Influence of Aggregation Methods on Loss

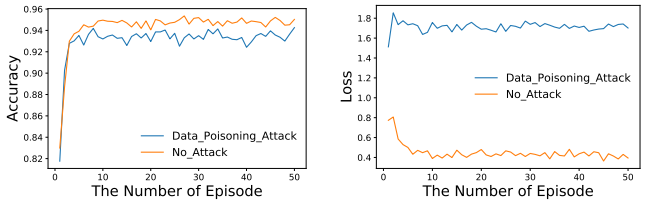
Fig. 5: Influence of Aggregation Methods on Accuracy and Loss.

Then, we will use three sets of experiments to illustrate the impact of poisoning attacks and label flipping attacks on FL. The results are shown in Figs. 6-8, we can see that all of the attacks can harm the performance of FL severely. This also reflects the inability of FedAvg aggregation methods to defend against these attacks. This is because FedAvg adopts aggregation based on the mean value of the data volume and cannot counteract the impact of the modified model, data, and labels on the global model.



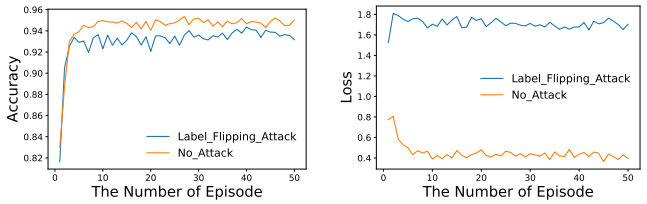
(a) Influence of Model Poisoning Attack on Accuracy (b) Influence of Model Poisoning Attack on Loss

Fig. 6: Influence of Model Poisoning Attack on Accuracy and Loss.



(a) Influence of Data Poisoning Attack on Accuracy (b) Influence of Data Poisoning Attack on Loss

Fig. 7: Influence of Date Poisoning Attack on Accuracy and Loss.



(a) Influence of Label Flipping Attack on Accuracy (b) Influence of Label Flipping Attack on Loss

Fig. 8: Influence of Label Flipping Attack on Accuracy and Loss.

2) Experiments of Blockchain

We create a blockchain system that is mainly used to record all FL-related transactions that occur on the MEC servers.

We have created a blockchain system that is mainly used to record all FL-related transactions that occur on the MEC server. When clients upload the updates to the MEC server, the MEC server as a blockchain node will broadcast the updates and pack them into the block to be processed, and then execute the consensus algorithm. After all nodes reach consensus, the pending block where all the updates are located can be adopted as the new block. We show the blocks with the updates in Listing 1. The new block has the updates submitted by each client and the new global model updates. From this experiment, we find that all the updates of FL are tensors, so we need to design a reasonable storage method to reduce the complexity of accessing the data.

V. Conclusion

We open-source a flexible blockchain-based FL framework for deployment on MEC servers called FBFL. We present the system of this framework, including its structure and workflow. We also briefly introduce the functions that FBFL currently has and verify its reliability through extensive experiments. We do not cover some developed functions in this article because they are not been formally published yet. In the future, we will continue to improve and maintain this framework for the convenience of researchers and industry.

Acknowledgment

References

[1] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep

Listing 1 Block Example


```
{
    "length": 2,
    "chain": [
            {
                "index": 0,
                "transactions": [],
                "timestamp": 1651177454.069026,
                "previous_hash": "0",
                "nonce": 0,
                "hash": "df016d2096955231961593 3bd708cfb2adefe8f329
                e4f29114134cffd4509b35"},
            {
                "index": 1,
                "transactions": [[0.005951858591288328, 0.002528209937736392,
                 -0.0016629062592983246, -0.0016865723300725222,
                 -0.008074020966887474, 0.006826113909482956,
                 -0.001547589898109436, -0.004972842056304216,
                 -0.0003091186226811 26, 0.0048900567926466465,
                 -0.0004124455153942108, 0.007141427136957645,
                 2.6170164346694946e-05, -0.0038283064495772123,
                 0.0022981807123869658, 0.005529563874006271,
                 -0.005910353269428015, -0.0054094563238322735,
                 -0.005291022825986147, 0.0006317742518149316,
                 ... ... , ... ...
                 -0.0005334370071068406, 0.005220452789217234,
                 0.008070948533713818, -0.006574320141226053,
                 0.005917795933783054, -0.0015515469713136554,
                 -0.0024922348093241453, 0.0003876827831845730 5]],
                "timestamp": 1651177457.4264069,
                "previous_hash": "df016d2096955231961593 3bd708c
                 fb2adefe8f329e4f29114134cffd4509b35",
               "nonce": 23879,
                "hash": "000009f3d41f22a2ec2356817928c03b0e0457
                 438d9c2670e4460916968b9de7"
             }
        ]
}
```


networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[2] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. Local model poisoning attacks to {Byzantine-Robust} federated learning. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 1605–1622, 2020.

[3] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Model poisoning attacks in federated learning. In *Proc. Workshop Secur. Mach. Learn.(SecML) 32nd Conf. Neural Inf. Process. Syst.(NeurIPS)*, pages 1–23, 2018.

[4] Virat Shejwalkar and Amir Houmansadr. Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning. In *NDSS*, 2021.

[5] Vale Tolpegin, Stacey Truex, Mehmet Emre Gursoy, and Ling Liu. Data poisoning attacks against federated learning systems. In *European Symposium on Research in Computer Security*, pages 480–501. Springer, 2020.

[6] Florian Nuding and Rudolf Mayer. Data poisoning in sequential and parallel federated learning. In *Proceedings of the 2022 ACM on International Workshop on Security and Privacy Analytics*, pages 24–34, 2022.

[7] Ronald Doku and Danda B Rawat. Mitigating data poisoning attacks on a federated learning-edge computing network. In *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–6. IEEE, 2021.

[8] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*, pages 2938–2948. PMLR, 2020.

[9] Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H Brendan McMahan. Can you really backdoor federated learning? *arXiv preprint arXiv:1911.07963*, 2019.

[10] Hongyi Wang, Kartik Sreenivasan, Shashank Rajput, Harit Vishwakarma, Saurabh Agarwal, Jy-yong Sohn, Kangwook Lee, and Dimitris Papailiopoulos. Attack of the tails: Yes, you really can backdoor federated learning. *Advances in Neural Information Processing Systems*, 33:16070–16084, 2020.

[11] Zhilin Wang and Qin Hu. Blockchain-based federated learning: A comprehensive survey. *arXiv preprint arXiv:2110.02182*, 2021.

[12] Paritosh Ramanan and Kiyoshi Nakayama. Baffle: Blockchain based aggregator free federated learning. In *2020 IEEE International Conference on Blockchain (Blockchain)*, pages 72–81. IEEE, 2020.

[13] Yuanhang Qi, M Shamim Hossain, Jiangtian Nie, and Xuandi Li. Privacy-preserving blockchain-based federated learning for traffic flow prediction. *Future Generation Computer Systems*, 117:328–337, 2021.

[14] Dinh C Nguyen, Ming Ding, Quoc-Viet Pham, Pubudu N Pathirana, Long Bao Le, Aruna Seneviratne, Jun Li, Dusit Niyato, and H Vincent Poor. Federated learning meets blockchain in edge computing: Opportunities and challenges. *IEEE Internet of Things Journal*, 2021.

[15] Dun Li, Dezhi Han, Tien-Hsiung Weng, Zibin Zheng, Hongzhi Li, Han Liu, Arcangelo Castiglione, and Kuan-Ching Li. Blockchain for federated learning toward secure distributed machine learning systems: a systemic survey. *Soft Computing*, pages 1–18, 2021.

[16] Hyesung Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. Blockchained on-device federated learning. *IEEE Communications Letters*, 24(6):1279–1283, 2019.

[17] Umer Majeed and Choong Seon Hong. Flchain: Federated learning via mec-enabled blockchain network. In *2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pages 1–4. IEEE, 2019.

[18] Rui Wang, Heju Li, and Erwu Liu. Blockchain-based federated learning in mobile edge networks with application in internet of vehicles. *arXiv preprint arXiv:2103.01116*, 2021.

[19] Zhilin Wang, Qin Hu, Ruinian Li, Minghui Xu, and Zehui Xiong. Incentive mechanism design for joint resource allocation in blockchain-based federated learning. *arXiv preprint arXiv:2202.10938*, 2022.

[20] Jiawen Kang, Zehui Xiong, Dusit Niyato, Han Yu, Ying-Chang Liang, and Dong In Kim. Incentive design for efficient federated learning in mobile networks: A contract theory approach. In *2019 IEEE VTS Asia Pacific Wireless Communications Symposium (APWCS)*, pages 1–5. IEEE, 2019.

[21] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. *Advances in Neural Information Processing Systems*, 30, 2017.

[22] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learning*, pages 5650–5659. PMLR, 2018.

[23] Yang Xiao, Ning Zhang, Wenjing Lou, and Y Thomas Hou. A survey of distributed consensus protocols for blockchain networks. *IEEE Communications Surveys & Tutorials*, 22(2):1432–1465, 2020.

[24] Sivleen Kaur, Sheetal Chaturvedi, Aabha Sharma, and Jayaprakash Kar. A research survey on applications of consensus protocols in blockchain. *Security and Communication Networks*, 2021, 2021.

[25] Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In *OsDI*, volume 99, pages 173–186, 1999.

[26] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance and proactive recovery. *ACM Transactions on Computer Systems (TOCS)*, 20(4):398–461, 2002.

[27] Zhilin Wang Shengyang Li. Fbfl: A flexible blockchain based federated learning framework in mobile edge computing. https://github.com/wzljerry/FBFL-A-Flexible-Blockchain-based-Federated-Learning-Framework-in-Mobile-Edge-Computing.