Poster: A Reliable and Accountable Privacy-Preserving Federated Learning Framework using the Blockchain

Sana Awan[1], Fengjun Li[1], Bo Luo[1], Mei Liu[2]
1. Department of Electrical Engineering and Computer Science, The University of Kansas
2. University of Kansas Medical Center
{sanaawan,bluo,fli}@ku.edu,meiliu@kumc.edu

ABSTRACT

Federated learning (FL) is promising in supporting collaborative learning applications that involve large datasets, massively distributed data owners and unreliable network connectivity. To protect data privacy, existing FL approaches adopt (k, n)-threshold secret sharing schemes, based on the semi-honest assumption for clients, to enable secure multiparty computation in local model update exchange which deals with random client dropouts at the cost of increasing data size. These approaches adopt the semi-honest assumption for clients, therefore they are vulnerable to malicious clients. In this work, we propose a blockchain-based privacy-preserving federated learning (BC-based PPFL) framework, which leverages the immutability and decentralized trust properties of blockchain to provide provenance of model updates. Our proof-of-concept implementation of BC-based PPFL demonstrates it is practical for secure aggregation of local model updates in the federated setting.

CCS CONCEPTS

• **Security and privacy → Public key (asymmetric) techniques**; **Privacy-preserving protocols**; • **Computing methodologies → Online learning settings**.

KEYWORDS

Federated Learning; Privacy; Blockchain

1 INTRODUCTION

Federated Learning (FL) was first proposed by Google [2, 3, 6] to facilitate large-scale collaborative learning. It no longer requires to transmit data samples distributed across multiple devices to a data center to train or update the global model. In this approach, each device trains its local model and exchanges local model updates to compute the global model update. Compared with conventional centralized learning, FL reduces not only communication costs by moving model updates instead of raw data items from clients to data centers, but also computational costs by leveraging the computing resource of each device. Moreover, since local data items never leave the devices (i.e., data owners), FL improves user privacy. With these desirable properties, FL is promising to support collaborative learning applications that involve large datasets and massively distributed data owners.

Early FL designs assumed that intermediate results, such as parameter updates of stochastic gradient descent, contain less information than the raw training data [6]. Therefore, ephemeral updates are often exposed. However, these gradients may leak important information of the local data items, especially when metadata such as the structure of data is available. To address this problem, several secure aggregation algorithms for FL have been proposed recently [2, 3], leveraging secret sharing and differential privacy techniques. However, these approaches assume a loose federation of participating clients that may join or leave the learning task unpremeditated, and therefore suffer from random client dropouts. [3] proposed to adopt (k, n)-threshold secret sharing to improve the robustness against dropouts. In fact, the security guarantee of these schemes is rooted in the semi-honest assumption for clients, which assumes the clients shall not submit “fake” shares nor collude with each other to manipulate learning process or outcome. This is a strong assumption, especially when the sources of updates are believed to be “not needed by the aggregation algorithm” and the updates “can be transmitted without identifying meta-data” [6]. Moreover, almost all existing work on FL explicitly or implicitly adopts a simple incentive model, which assumes that clients voluntarily participate in collaborative learning to trade their local model updates and computing resources for an improved global model. This flat incentive model neglects the fact that clients with different data sizes and computing capabilities make different contributions in the global optimization task and should be rewarded differently.

We propose a blockchain-based privacy-preserving federated learning (BC-based PPFL) framework in this work. As shown in Figure 1, the blockchain interconnects FL components, such as the server, clients and aggregators. It uses a distributed ledger of transactions to record information flows regarding FL tasks, participating clients, local and global model updates, etc., among the components. The immutable ledger supports data provenance by tracking data flows in each FL task and provides a good trust base to build a verification mechanism that existing FL approaches lack. With such a verification mechanism, we can further extend the *semi-honest client* assumption to a more realistic *malicious client* assumption, under which a client can drop out, submit fake local updates, or collude with other malicious clients. Moreover, with the distributed ledger, the server as well as other interested entities (e.g., clients and aggregators) in an FL task can track the contribution of every client towards a globally optimized learning model, which makes contribution-based incentive mechanisms possible. Based on this, we can further introduce a premium on the ownership of the improved model and reward the miners accordingly.

CCS '19, November 11–15, 2019, London, United Kingdom

ACM ISBN 978-1-4503-6747-9/19/11.
https://doi.org/10.1145/3319535.3363256

2 THE BC-BASED PPFL FRAMEWORK

2.1 The Federated Learning Model

The primary goal of federated learning is to train a global shared model on all the data across a large number of clients, wherein each client maintains her data securely. While different learning tasks have been proposed for FL, in this work, we adopt the federated averaging model introduced in [6], where a fixed set of K clients, each holding n_k data points, compute the *average gradients* on their local data with the current model w_t at a fixed learning rate η: $g_k = \nabla F_k(w_t)$, where $F_k(w_t) = \frac{1}{n_k} \sum_{i \in P_k} f_i(w_t)$ is the loss of the prediction on local data points. Therefore, each client can update its local model as $w^k_{t+1} = w_t - \eta g_k$, while the server can aggregate the received local model updates to update the global model as $w_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} w^k_{t+1}$.

The primary **privacy objectives** of BC-based PPFL are to (a) protect local data points from any external entity by never shipping them out of the devices; (b) protect local data from the server by never leaking individual local model updates to the server; and (c) protect local and global model updates from irrelevant internal (i.e., aggregators, miners) and external entities. Meanwhile, the **security objectives** are to ensure (a) the confidentiality and integrity of the local model updates from committed clients and (b) the provenance of model updates.

2.2 The Architecture

To achieve the security and privacy protection goals, we propose a BC-based PPFL framework, which consists of five components as shown in Figure 1. For an FL task, the *server* first advertises task specifications, such as the types of applications (e.g., keystroke or activity predication), types of devices, types and formats of training data (e.g., gyroscope or motion sensor data, floating point format), types of learning models (e.g., CNN) , computing requirements (e.g., learning rate), and task settings (e.g., aggregators, number of needed clients, batch size, etc.) in a task transaction. The *clients* who are willing to join a task register to the *aggregator*, which then creates a commitment transaction for all the committed clients. Finally, the *miner* records all the transactions into the ledger.

During the FL task, the server first shares the initial model with all committed clients. For model privacy, this initial model is shared in protected mode. Then, the clients compute the local model updates over their local training data and upload the updated model parameters to the aggregator. All the updates received by the aggregator in the same batch will be wrapped in a local update transaction and recorded into the ledger with the assistance of the miner.

2.3 Building Blocks

BC-based PPFL relies on three building blocks to provide data privacy and provenance in FL.

Homomorphic Encryption and Proxy Re-encryption. Most of the existing privacy-preserving federated learning approaches adopt secret sharing schemes for secure multiparty computation (MPC), which suffer from random client dropouts. To tackle this problem, we adopt a variant of the Paillier cryptosystem with two trapdoor functions proposed by Cramer and Shoup [4] to protect local models during model exchange.

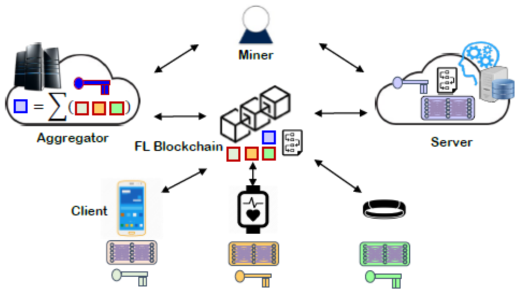

Figure 1: The BC-based PPFL framework.

In our design, the server generates a pair of public and private task keys for each task and includes the public task key in the task specification, while each aggregator involved in a task generates a pair of public and private batch keys and distributes the public batch key to each committed client. The task keys and batch keys are constructed in a form such that the batch key is the *transformation* key of the corresponding task key to support proxy re-encryption. Therefore, the server cannot directly recover individual local updates recorded in the ledger, instead, it only recovers the aggregated value after the aggregator aggregates the updates and re-encrypts it with the transformation key.

Blockchain. There have been several proposals of using blockchain to support deep learning [7] and federated learning [5] for incentive purposes. While we adopt blockchain for provenance and verification, the logic design of the blockchain network is similar. The miners can physically be any node with sufficient computing resources to run the Proof-of-Work (PoW) or Proof-of-Stake (PoS), and record the task, commitment and update transactions into the task block. The use of blockchain also makes contribution-based incentive mechanisms possible, whereby we introduce a premium on the ownership of the improved model held by the server and reward the miners accordingly. The size of each block is related to the header and the learning rate η. It is also worth noting that for large model updates atop large datasets, we can store the encrypted model updates in a distributed database such as IPFS [1] and record only the addresses in the blockchain.

Verification. The blockchain records the flows of encrypted model updates, which makes the tracking and verification of clients' contribution to the globally optimized model possible. In particular, the verifier can retrieve all the updates of the clients in a batch, generate the aggregate excluding the input from the suspicious client, and compare the performance of the global models before and after verification. In the preliminary experiment, we implemented the basic verification function in which the server works as the verifier to evaluate the gradients after recovering the aggregate in each round. It compares the performance of the updated global model with the initial model of that round using the loss function.

2.4 A Proof-of-Concept Implementation and Evaluation

We developed a variant of the Paillier cryptosystem in Python to implement homomorphic encryption and proxy re-encryption for gradient aggregation. We built the blockchain over Ethereum and chain operations are controlled using the Truffle suite, which is a development and testing framework for Ethereum. Finally, to assess the performance of our proposed framework, we adopted the *nn* package of the deep-learning library PyTorch for FL tasks.

Settings. In particular, we built a binary classifier using a simple two layer neural network, and used the Breast Cancer Dataset from UCI Machine Learning Repository as the training data. The dataset has 569 samples and 30 features. For all three cryptosystems used in our framework, we generated public and private key pair of default length 2048 bits.

We ran the blockchain on an Amazon EC2 t2.micro instance with 1GiB RAM and 3.3GHz Intel Scalable Processor. For Server, Aggregator and Client nodes, we adopted two settings: in *Setting I*, we ran all three types of nodes on a laptop with 2 GHz Intel core i5 processor and 3 GiB of memory, from which they can interact with the Blockchain implemented on Amazon EC2; in *Setting II*, we ran them on Amazon EC2 instances too, which minimizes communication delays since all instances are on the same cloud platform. In the future, we will further study the scalability of our framework and the communication delays by deploying different nodes on multiple cloud platforms.

Evaluation. To assess the performance of our scheme, we ran a comparably small-scale experiment for FL model training. We first randomly divided the training data among 10 clients, each having 50 examples from the IID-partitioned data. Then, we measured the execution time for different FL operations (i.e., model encryption, local model update, aggregation, blockchain write for local updates and gradient average, and global model update).

We repeated model training four times and calculated the average execution time in a single iteration, as shown in Table 1. Obviously, aggregating the gradients and writing the result to the blockchain are the most computationally costly operation among all FL operations under both settings. The average execution time per iteration is the summation of all FL operations, which is 1.62 and 1.27 seconds in Setting I and II, respectively. The difference between the two settings indicates the communication cost introduced by reading and writing to the blockchain.

The performance of the proposed scheme, especially the gradient aggregation and blockchain write operations, is related to the number of the clients participating in the PPFL tasks [6]. So, we increased the client batch size to 20, 30, and 40, and measured the execution time. The average execution time increased from 1.62 seconds for 10 clients to 1.73, 2.36, and 2.43 seconds for 20, 30 and 40 clients, respectively. This indicates that increasing the number of clients results in a sub-linear increase in the protocol execution time.

Table 1: Average Processing Time of each FL Operation (in seconds; computed over 4 runs for 10 clients).

Operations	Setting I	Setting II
Key Generation (offline)	6.568e-05	6.317e-05
Model Encryption	0.00337	0.00403
Blockchain Write for Model	0.347	0.201
Local Model Update	0.046	0.0434
Blockchain Write for Gradient	0.36	0.236
Aggregation	0.605	0.572
Blockchain Write for Gradient Average	0.261	0.174
Global Model Update	0.079	0.1201

3 CONCLUSIONS AND FUTURE WORK

We present the design of a privacy-preserving federated learning framework that performs gradient aggregation over private data following a cryptographic protocol. We design a variant of the Paillier cryptosystem to support additive homomorphic encryption and proxy re-encryption so that the encrypted local model updates can be aggregated and transformed into a form recoverable by the server. Throughout the process, individual local model updates are protected from the server, aggregators and other clients, while the global model update is shared among participating clients. By integrating the blockchain in PPFL process, we can overcome random client dropouts since their local updates are written to the blockchain asynchronously. Moreover, due to the immutability and data provenance properties of the global ledger, it provides a means to identify and exclude malicious client updates. Our preliminary experimental results show that BC-based PPFL supports model training in a fully decentralized manner and provides better transparency and verifiability, while protecting data privacy.

In the future, we will explore federated optimization by incorporating mini-batching during each training epoch at the client, and increase multi-client parallelism to achieve a target test-set accuracy. We will also consider non-IID partitioned data and investigate incentive schemes to reward clients based on their contributions.

ACKNOWLEDGEMENT

This work is sponsored in part by NSF CNS-1422206 and DGE-1565570, NSA SoS Initiative, and the Ripple University Blockchain Research Initiative.

REFERENCES

[1] Juan Benet. Protocol Labs, 2016. InterPlanetary File System (IPFS), 2016. (Protocol Labs, 2016). Retrieved July 07, 2019 from https://www.ipfs.io

[2] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konecny, Stefano Mazzocchi, H. Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. 2019. Towards Federated Learning at Scale: System Design. In *Proceedings of The 2nd SysML Conference, Stanford, California, March, 2019.*

[3] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In *Proceedings of ACM Conference on Computer and Communications Security, 2017.*

[4] Ronald Cramer and Victor Shoup. 2002. Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption. In *Proceedings of the 21st Annual International Conference on the Theory and Applications of Cryptographic Techniques, EUROCRYPT 2002.*

[5] Hyesung Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. 2019. On-Device Federated Learning via Blockchain and its Latency Analysis. In *IEEE Communications Letters, 2019.*

[6] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proccedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017.*

[7] Jia-Si Weng, Jian Weng, Ming Li, Yue Zhang, and Weiqi Luo. 2018. DeepChain: Auditable and Privacy-Preserving Deep Learning with Blockchain-based Incentive. *IACR Cryptology ePrint Archive* 2018 (2018), 679.