This article has been accepted for publication in IEEE Transactions on Network Science and Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2024.3360311

Secure Decentralized Aggregation to Prevent Membership Privacy Leakage in Edge-based Federated Learning

Meng Shen, *Member, IEEE,* Jing Wang, Jie Zhang, Qinglin Zhao, *Senior Member, IEEE,* Bohan Peng, Tong Wu, Liehuang Zhu, *Senior Member, IEEE*, Ke Xu, *Fellow, IEEE*

***Abstract*—Federated Learning (FL) is a machine learning approach that enables multiple users to share their local models for the aggregation of a global model, protecting data privacy by avoiding the sharing of raw data. However, frequent parameter sharing between users and the aggregator can incur high risk of membership privacy leakage. In this paper, we propose LiPFed, a computationally lightweight privacy preserving FL scheme using secure decentralized aggregation for edge networks. Under this scheme, we ensure privacy preservation on the aggregation side, and promote lightweight computation on the user side. By incorporating blockchain and additive secret sharing algorithm, we effectively protect the membership privacy of both local models and global models. Furthermore, the secure decentralized aggregation mechanism safeguards against potential compromises of the aggregator. Meanwhile, smart contract is introduced to identify malicious models uploaded by edge nodes and return trustworthy global models to users. Rigorous security analysis shows the effectiveness of this scheme in privacy preservation. Extensive experiments verify that LiPFed outperforms the state-of-the-art schemes in terms of training efficiency, model accuracy, and privacy preservation.**

***Index Terms*—Federated learning, privacy preservation, decentralized aggregation, consortium blockchain.**

I. Introduction

FEDERATED learning (FL) has emerged as a highly promising paradigm for machine learning [1], experiencing remarkable growth in recent years. It facilitates collaborative model training among multiple users, each possessing local datasets. During each training iteration, users independently train local models on their respective datasets, and subsequently, these local model updates are aggregated to construct a global model, which is then broadcast to users for subsequent iteration. Unlike traditional centralized training methods, FL fosters the sharing of model updates instead of raw training data, thereby ensuring data privacy for individual users. As a result, FL has been widely applied by widespread applications in various fields, such as mobile computing, the Internet of Vehicles, and the Internet of Things [2, 3].

FL enhances user experience in various scenarios while ensuring privacy protection. For instance, Google Keyboard (Gboard) [4] utilizes FL to predict the following input phrase by training a model with user preference data. In the Internet of Things (IoT), information sensing devices gather extensive data for efficient completion of downstream tasks [3]. Similarly, in the Internet of Vehicles (IoV), intelligent cars leverage FL to train on collected data, enhancing traffic pattern predictions and optimizing information interaction [2]. Local model updates from mobile phones, information sensors, cars, and other devices are transmitted to an aggregator for the optimization of a global model, promoting collaborative intelligence while maintaining data privacy.

However, there are certain limitations when it comes to model sharing in such scenarios. First, despite the training data being kept local, adversaries can potentially infer membership privacy by analyzing local and global models. Second, the noise introduced to protect the model adversely affects model convergence, making it less suitable for training tasks that require high model accuracy. Last, limited computational resources in these scenarios make it challenging to employ complex encryption methods. Additionally, due to restricted cross-domain network communication capabilities, users prefer to communicate with nearby servers.

Membership refers to whether specific data has been used to train a machine learning model. This paper focuses on two attacks on membership: *membership inference attacks* [5–9] and *isolating attacks* [5] to enhance membership privacy leakage in FL. In membership inference attacks, an adversary aims to train an attack model to infer whether a data record exists in the user's training dataset by leveraging the discrepancy in the model's behavior on training and non-training data. In FL scenario, aggregation diminishes the contribution of the local model of an individual user, leading to negative influence on the accuracy of membership inference attacks. Isolation attacks can be employed to isolate the target user, preventing the local model of the target user from getting aggregated with the local models of other users. Thus, more information about the training dataset within the model is stored.

Centralized FL can incur membership privacy leakage,

This work is partially supported by National Key R&D Program of China with No. 2022YFB2902900, NSFC Projects with Nos. 62222201 and U23A20304, Beijing Nova Program with No. 20220484174, Beijing Natural Science Foundation with Nos. M23020, L222098 and 7232041, Science and Technology Development Fund, Macau SAR (File No. 0076/2022/A2).

M. Shen, J. Wang, B. Peng, T. Wu and L. Zhu are with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing 100081, China. (e-mail: shenmeng@bit.edu.cn, wang_jing@bit.edu.cn, bohanpeng_bit@163.com, tongw@bit.edu.cn, liehuangz@bit.edu.cn).

J. Zhang is with the School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China (e-mail: zhangjie_bit@hotmail.com).

Q. Zhao is with the School of Computer Science and Engineering, Macau University of Science and Technology, Macau 999078, China (e-mail: qlzhao@must.edu.mo).

K. Xu is with the Department of Computer Science, Tsinghua University, Beijing 100084, China (e-mail: xuke@tsinghua.edu.cn).

Authorized licensed use limited to: Iowa State University Library. Downloaded on March 06,2024 at 23:25:47 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Network Science and Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2024.3360311

TABLE I: Comparison of Representative Privacy Preserving FL Methods

Privacy Preserving Techniques	Representative Methods	Training Efficiency on User Side[1]	Model Accuracy	Verification of Aggregation	Defending against Attacks	
					Membership Inference Attack	Isolating Attack
Shamir Secret Sharing	VerifyNet [10]	$5\times \sim 10\times$	Unaffected	✓	×	✓
	PSA [11]	$5\times \sim 10\times$	Unaffected	×	×	×
Differential Privacy (DP)	CDPFed [12]	$<2\times$	Decreased	×	×	×
	LDPFed [13, 14]	$<2\times$	Decreased	×	✓	×
	HyDPFed [15]	$<2\times$	Decreased	×	✓	×
Homormophic Encryption (HE)	Partially-HE [16, 17]	$>10\times$	Unaffected	×	✓	×
	Fully-HE [18, 19]	$>10\times$	Decreased	×	✓	×
Additive Secret Sharing	**LiPFed (This paper)**	$\mathbf{<2\times}$	**Unaffected**	✔	✔	✔

[1] We take FL without any defense method as a baseline. Training Efficiency on User Side represents the training time required on the user side compared with the baseline. Model Accuracy represents the accuracy loss compared with the baseline.

which has prompted significant research efforts in developing defensive measures. Recent studies have proposed several privacy preserving FL schemes, as summarized in Table I, which mainly resort to three types of privacy preserving techniques, including Shamir secret sharing, differential privacy (DP), and homomorphic encryption (HE).

Specifically, PSA [11] and VerifyNet [10], which rely on Shamir secret sharing techniques, may not always provide comprehensive defense against membership inference attacks as they expose the plaintext of the global model in certain scenarios. VerifyNet, while offering the ability to verify global models through homomorphic encryption-generated proofs, imposes computational burden on users. DP-based methods are divided into Local Differential Privacy (LDPFed) and Central Differential Privacy (CDPFed). In LDPFed [13, 14], participants add noise to their local models. They aim to ensure privacy by adding as much noise as possible to their local gradients, which can result in accumulated errors in the global model. CDPFed [12], on the other hand, involves a trusted aggregator adding noise. This method achieves acceptable accuracy only in scenarios with a large number of participants. HyDPFed [15] is a hybrid differential privacy method that combines local and centralized differential privacy. The HE-based method, Partially-HE [16, 17], supports additive homomorphism and effectively protects the privacy of local models. However, it comes with an increased computational overhead on the user side. Fully-HE [18, 19] ensures homomorphism for both addition and multiplication operations but introduces noise, which can diminish the usability of the model.

In centralized FL, membership privacy can be inferred through shared local models or aggregated global models, and existing privacy preserving schemes for centralized FL have some drawbacks. There is currently no specific attack targeting membership privacy in decentralized FL. Therefore, we propose LiPFed, a computationally lightweight and privacy preserving FL scheme using secure decentralized aggregation. It integrates FL with edge network and blockchain to achieve accurate and trustworthy global models while protecting membership privacy of users.

In contrast to centralized FL, our approach utilizes multiple edge nodes located at the user's edge to establish a secure decentralized aggregation platform, enabling collaborative aggregation of local models. In edge networks, the dominant central aggregator in FL is weakened as multiple distributed edge nodes participate. We employ blockchain as the underlying infrastructure due to its decentralized, immutable, and consensus-driven characteristics, which make it suitable for constructing a decentralized FL scheme. The shared ledger of the blockchain records all intermediate aggregation results from the edge nodes, providing the ability to detect any malicious aggregation attempts by compromised edge nodes.

In LiPFed, each user possessing private data can connect with multiple edge nodes. To protect the membership privacy of users while guaranteeing the accuracy of the global model, we design Additive Secret Sharing (Add-SS) algorithm and Secure Aggregation algorithm. Within this scheme, all local models and aggregated global models appear in the form of ciphertext throughout the FL process. The models cannot be recovered by either a single edge node or a group of collusive edge nodes under the byzantine assumption [20].

To summarize, the contributions of this paper are as follows:

- We propose LiPFed, a novel decentralized scheme for edge networks, which protects the membership privacy of users' training data, and enables verification towards aggregation of edge nodes by smart contracts on blockchain.
- We design an Additive Secret Sharing algorithm based on one-time padding to protect local and global models. This algorithm reduces the computational overhead of privacy preservation on the user side, and ensures high usability of the global model without any compromise on accuracy.
- We conduct a rigorous security analysis to prove the security of the proposed scheme. Simultaneously, extensive experiments are carried out to validate that LiPFed outperforms state-of-the-art approaches in terms of computational efficiency, model accuracy, and privacy protection against membership inference attacks.

The rest of this paper is organized as follows. We describe the preliminaries in Section II. We present the system model, threat model and design goals in Section III. Then, we describe LiPFed in detail in Section IV. After that, we present rigorous security analysis in Section V and evaluate the performance of the proposed scheme in Section VI. We sort out the related work in Section VII. We conclude this paper in Section VIII.

Authorized licensed use limited to: Iowa State University Library. Downloaded on March 06,2024 at 23:25:47 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Network Science and Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2024.3360311

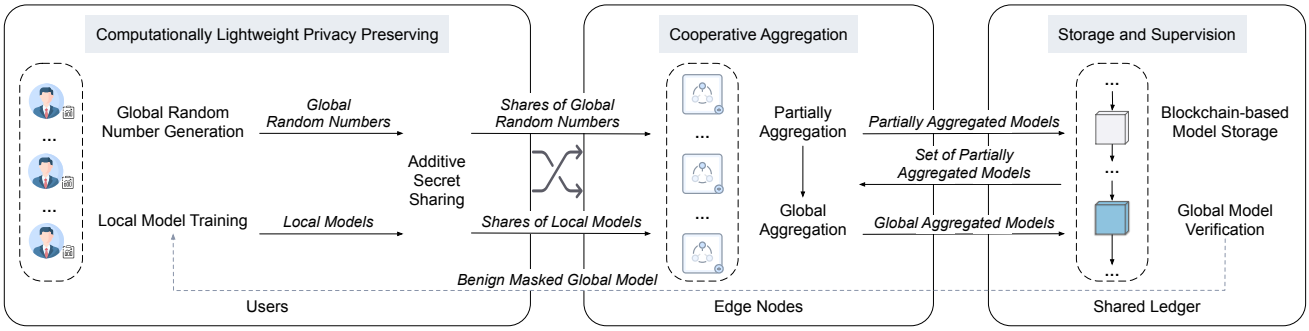


Fig. 1: System Overview of LiPFed.

II. Preliminaries

A. Federated Learning

The concept of FL was originally proposed by Google. The main idea is to establish a machine learning model based on the data set distributed on multiple devices, while users' data is stored locally. In an iteration of FL, the users first download the global model from the server. Then they use the local data to train the model and upload the parameters to the server. The server aggregates the parameters of each user to update the global model. Finally, the users get the updated model from the server and update their local models.

Suppose there are N users in the FL training process, and user i has a local training dataset D_i. The objective of each user i is to minimize the loss function:

$$L_i(\theta_i) = \frac{1}{|D_i|} \sum_{j \in D_i} l_j(\theta_i) \tag{1}$$

$$L(\theta) = \frac{1}{|N|} \sum_{i \in N} \sum_{j \in D_i} \tag{2}$$

where N is the number of users and $|.|$ denotes the size of datasets. $L_i(\theta_i)$ is a specific loss function. In this paper, we take cross entropy as the loss function.

The advantage of FL is that the users only need to share the local model with the server while the training data is saved locally from beginning to end. Compared with centralized training, FL can protect the user's privacy information against leakage. However, attackers can still carry out some privacy attacks through shared parameters (such as membership inference attacks), and the server can return well-constructed but incorrect models to users.

B. Pseudorandom Generator

Pseudorandom generator (PRG) is used to create random sequences of numbers in deterministic devices. All computer algorithms are strictly deterministic. PRG allows the encryption of many data blocks using data generated from secret keys that have only a few bits.

Definition 1. *Pseudorandom generator (PRG) is an efficient and deterministic function, which returns a longer pseudorandom output sequence based on the received shorter input:*

$$G : \{0,1\}_s \rightarrow \{0,1\}_n, n \gg s$$

PRG has to be unpredictable. There must not be any efficient algorithm that after receiving the previous output bits from PRG would be able to predict the next output bit with probability non-negligibly higher than 0.5.

PRG is used for creating pseudorandom functions and permutations, which are widely used in cryptography (for example, for the implementation of block ciphers).

C. Blockchain

Blockchain is a new distributed infrastructure and computing paradigm that uses a series of blocks to verify and store data [21, 22]. It uses distributed node consensus algorithm to generate and update data, uses cryptography technology to ensure the security of data transmission and access control, and uses a smart contract composed of automated script code to program and operate data. A smart contract is a piece of code deployed on blockchain network. Once an event triggers the terms in the contract, the code will be executed automatically. Smart contract provides LiPFed with functions for uploading, querying, and detecting malicious edge nodes.

To achieve a balance between the publication of blockchain and the protection of data privacy, we apply consortium blockchain in LiPFed, which ensures both security and efficient data processing. The consortium blockchain used in LiPFed is maintained collectively by multiple edge nodes that share resources and responsibilities, ensuring the stability and reliability of the entire system.

III. System Overview

In this section, we present a novel and computationally lightweight privacy preserving FL scheme (LiPFed), which achieves effective privacy preservation, high training efficiency, and high accuracy.

A. System Model

As shown in Fig.1, three types of entities are participating in the system model: users for local training, edge nodes for secure aggregation, and blockchain for data sharing. Three parts divide and conquer to complete the FL process.

1) Users: Users serve as data collectors and holders, utilizing their local data to train local models. As an example, within LiPFed, multiple users collaborate to train an image classifier. They utilize local models built on convolutional neural networks (CNNs) or ResNet to train this classifier,

Authorized licensed use limited to: Iowa State University Library. Downloaded on March 06,2024 at 23:25:47 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Network Science and Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2024.3360311

TABLE II: Notations Used in This Paper

Notations	Description
U	All users in FL
U_s^t	Selected users in a iteration t
λ	Learning rate
T	Total training iteration
m	The number of selected users in current iteration
n	The number of edge nodes
k	The number of semi-honest edge nodes
l	The number of edge nodes each user connected
u^i	The i-th selected user in current iteration
e^j	The j-th edge node in blockchain
$(w)^i$	Local model of u^i
$(w)_j^i$	The j-th share of $(w)^i$
$(g)_j^i$	Partial aggregated model of e^j for u^i
$(R)^i$	Global random number generated by of u^i
(g)	Plaintext of global model
$(g)^i$	Global model masked with $(R)^i$

continually updating the model through stochastic gradient descent (SGD) [23]. Then, they employ PRG to generate random numbers for masking and verification, respectively. Each user connects to multiple edge nodes (e.g. access point, base station), where honest ones among these nodes should exceed the total number of Byzantine nodes. Users split their parameters into shares using additive secret sharing and distribute them to the edge nodes.

2) Edge Nodes: Edge nodes serve as both local model aggregators and blockchain consensus nodes, working together to provide users with secure decentralized aggregation of local models. Edge nodes can be aggregation service platforms built at the edge of the user network, provided and supported by service providers offering storage, computing, and networking resources. Edge nodes reach consensus and collectively construct a blockchain shared ledger to facilitate decentralized aggregation.

3) Shared Ledger: Shared ledger is a data sharing and supervision platform aiming at helping accomplish decentralized model aggregation. A pre-written smart contract is deployed on the consortium blockchain to assist in the aggregation and detect the malicious global model. With a certain degree of access control security strategy, entities outside the edge nodes are unable to access the transactions on the ledger.

B. Threat Model

In this paper, we use a fixed number of edge nodes from the blockchain to act as aggregators, so we assume edge nodes are adversaries in the threat model. We consider *Honest but Curious Security* [24] in LiPFed, which means edge nodes are semi-honest. Each edge node honestly executes the smart contract specified in advance, but tries to infer useful private information of users through the shared parameters as much as possible. Here, we describe two types of attacks in our proposed LiPFed as follows:

- *Type-I Attack.* Type-I Attack is an isolating attack. The adversary refers to multiple semi-honest edge nodes. After completing decentralized aggregation, multiple edge nodes collude to jointly send the same well-constructed fake global model to the victim user. By exploiting the victim's local model updated based on the fake global model, the adversary expects to infer more membership privacy about training data.
- *Type-II Attack.* Type-II Attack is a membership inference attack. The adversary refers to multiple semi-honest edge nodes. Edge nodes are responsible for aggregating local models from multiple users to obtain a global model. Multiple semi-honest edge nodes collude to acquire shares of both local models and global random numbers as much as possible, and expect to infer the membership privacy about training data.

C. Design Goals

To meet the privacy training needs of computationally resource-constrained devices, LiPFed has three design goals:

1) Privacy Preservation: To defend against membership inference attacks [7, 25] that infer the users' privacy and isolating attacks [5] that aggravate the leakage of user's membership privacy, LiPFed employs privacy preserving methods to ensure the confidentiality of both local models and global models. To prevent edge nodes from computing fake global models to isolate users, the system needs to verify the aggregation results of global models and return the benign one to the user. For security, we require indistinguishability under ciphertexts of models. Assuming an adversary selects two plaintexts and receives one of their corresponding ciphertexts, if the adversary cannot distinguish which plaintext corresponds to the received ciphertext, then the goal of security satisfies indistinguishability.

2) Training Effciency: Considering the different computing power of each user, the lightweight privacy training time cost on the user side should be guaranteed. A large number of time-consuming operations such as lots of complex encryption and decryption operations should be avoided. We evaluate training efficiency by analyzing the time complexity of encryption on user side and comparing time overhead on both user side and aggregation side for different methods.

3) Model Accuracy: While the privacy preserving scheme is introduced, the accuracy of the trained model should not be significantly reduced. At the same time, the accuracy is less affected by the number of iterations of model training, ensuring that the accuracy is still kept in a large scale scenarios that require a large number of training iterations. Model accuracy is calculated by equation $R = t_p/(f_n + t_p)$, where t_p is the amount of positive instances that are classified correctly, and f_n is the amount of positive instances that are classified incorrectly. We expect the model accuracy under LiPFed to be as close as possible to FedAvg [1], which is a FL aggregation strategy without privacy preserving methods.

IV. The Proposed LiPFed

LiPFed is designed to protect users' membership privacy. Meanwhile, LiPFed protects users from isolating attacks by verifying aggregated global models. In this part, the detailed protocol shows how to combine additive secret sharing and decentralized aggregation to achieve the design goals.

Authorized licensed use limited to: Iowa State University Library. Downloaded on March 06,2024 at 23:25:47 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Network Science and Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2024.3360311

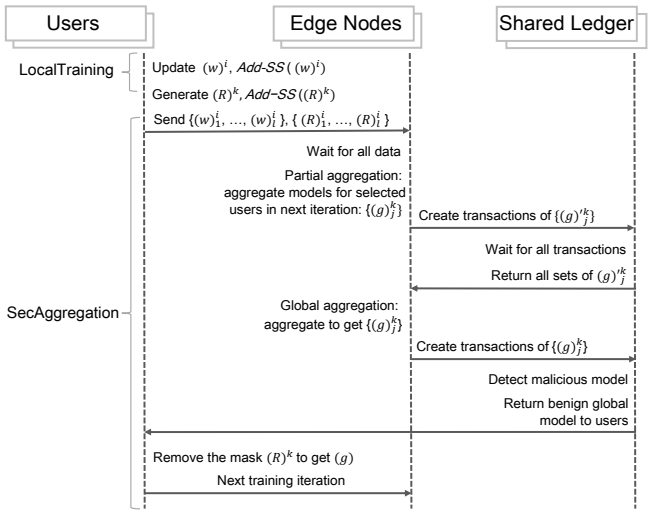


Fig. 2: High-Level Protocol of LiPFed. Before the protocol, the setup operations have been finished. This figure just shows the process per training iteration. iterations.

A. Overview of LiPFed

The high-level protocol of the LiPFed is summarized in Fig. 2. It contains three steps in each iteration of training, which are referred to as Setup Phase, LocalTraining and SecAggregation.

1) Setup Phase: The setup phase is shown in detail in Fig. 3. For users, some parameters about the training process and models need to be negotiated in advance. For blockchain shared ledger, LiPFed adopts a consortium blockchain to facilitate the decentralized aggregation process. This begins with the selection of edge nodes designated to participate in the consortium blockchain, collectively responsible for maintaining the integrity of the blockchain network. Edge nodes have to communicate with each other to build a consortium blockchain and begin to consensus, with LiPFed's preference being the Byzantine Fault Tolerance algorithm. Following this, the blockchain is meticulously configured to restrict access solely to authorized entities, bolstering security. To streamline operations, smart contracts are deployed onto the blockchain, enabling automated execution of model updates and the recording of intermediate results.

A single user is connected to several edge nodes which can promise a byzantine-tolerant training system. Meanwhile, each user chooses a fixed number of edge nodes and generates global random numbers for each iteration of the following training process. These global random numbers are split into shares by Algorithm 1 and sent to the connected edge nodes separately, which can be used to mask the global model.

2) LocalTraining: As shown in Fig. 3, selected users train local models using their local datasets and get $\{(w)^i \mid i \in [1, m]\}$ in this round. LiPFed is not concerned with the method of model training. Each user in LiPFed can train an image classifier based on CNNs or ResNet and adopt SGD [23] to update their local models.

3) SecAggregation: As shown in Fig. 4, in this round, LiPFed performs secure decentralized aggregation on local models from the selected users. In all, user u^i splits local model $(w)^i$ through an additive secret sharing algorithm `Add-SS` described in Algorithm 1 to obtain $\{(w)_j^i \mid j \in [1, l]\}$. Then they are sent to l connected edge nodes separately. Decentralized edge nodes perform two aggregation operations on the shares of local models and global random numbers and finally get the global model masked by global random numbers. All intermediate calculation results (i.e., $(g)_j^{'k}$ and $(g)_j^k$) are recorded in blockchain shared ledger. The smart contract detects the malicious models recorded in the shared ledger and eventually returns a benign global model to the user. Once a suspected malicious edge node is detected, the smart contract can mark it and replace the connection between it and the user, otherwise the connection state remains unchanged.

B. Additive Secret Sharing

We design an additive secret sharing algorithm and employ it in LiPFed to achieve three design goals, through which the shares will not disclose any information about plaintexts, and aggregation under ciphertexts is convergent.

Arithmetic sharing [26] is a low-cost secret sharing and it orients two-party secure computing. The main idea is to randomly split a number into two or more numbers. The split numbers belong to different calculation parties. Each calculation party can carry out arithmetic calculations under privacy protection according to the shared data. As an extension of this idea, we designed an additive secret sharing algorithm, through which we can share parameters to multiple edge nodes and perform secure aggregation.

Additive secret sharing allows multiple participants to perform additional operations without exposing their private inputs. This typically involves splitting a number or value into multiple parts and distributing them to different entities, ensuring that the final addition result can only be obtained when all parts are collected. This approach ensures that at no point during the process can any single participant gain insight into the private inputs of others. Complete information about the original value is revealed only when the final result is reconstructed. This enhances privacy protection, particularly in scenarios involving sensitive data in multi-party computations.

Additive secret sharing algorithm is based on additively sharing of private values between multiple parties as follows.

Shared Parameters. For a sharing value v, we have $v_1 + v_2 + ... + v_n \equiv v$.

Splitting. `Add-SS(`v`)` computes $v^* = v/n$, generates a set of pseudorandom numbers $\{r_1, r_2, ..., r_n\}$, sets v_i using the formula below, then sends v_i to the i^{th} connected edge node.

$$\begin{cases} v_i = v^* + r_i - r_{i+1}, & i \neq n \\ v_n = v^* + r_n - r_1, & i = n \end{cases} \tag{3}$$

Recovery. For all edge nodes aggregated, $v = \sum_{i=1}^{n} v_i$, all random numbers can be cancelled out.

We apply the above algorithm to LiPFed, each weight w^i in local model $(w)^i$ is split into several shares and masked with well-constructed pseudorandom numbers by Algorithm 1 `Add-SS(`w^i`)`. When a set of shares w_j^i from several users needs to be aggregated, local model can be reconstructed

Authorized licensed use limited to: Iowa State University Library. Downloaded on March 06,2024 at 23:25:47 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Network Science and Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2024.3360311

Setup Phase of LiPFed

Users

- All users negotiate the training parameters $\{w, T, \lambda\}$, which can be transmitted through the reliable channel offline, or through the assistance of the edge nodes.
- All users are connected to several edge nodes. In order to ensure security, the number of connections should not be less than $1 + n/3$, where n is the number of edge nodes.

Edge Nodes

- A smart contract for secure aggregation is deployed. All edge nodes participate in the consensus and form a consortium blockchain to provide services for users.
- Edge nodes select users $U_s^1 \subseteq U$ and $U_s^2 \subseteq U$ for training in iteration $t = 1$ and iteration $t = 2$ and reach a consensus.

LocalTraining

Users

- Selected users $U_s^t \subseteq U$ randomly take a batch of data instances from their local datasets.
- For selected user $u^i \in U_s^t$, it updates the local model w^i using Eq.(1)

Fig. 3: Setup Phase and LocalTraining of LiPFed.

Algorithm 1 Additive Secret Sharing: `Add-SS()`

Input: Parameter: v, number of shares: n, pseudorandom number seed: $seed$, pseudorandom number generator: **PRG**
Output: A set of shares: $\{v_i \mid i \in [1, n]\}$

1: Generate a set of pseudorandom number using **PRG**($seed$) and get $\{r_1, r_2, ..., r_n\}$
2: $v^* = v/n$
3: **for** $i = 1, 2, ..., n-1$ **do**
4: $\quad v_i = v^* + r_i - r_{i+1}$
5: **end for**
6: $v_n = v^* + r_n - r_1$
7: **return** $\{v_i \mid i \in [1, n]\}$

SecAggregation

Users

- $u^i \in U_s^t$ compute local model update $(w)^i$, splits secret $(w)^i$ into l pieces $\{(w)_1^i, ..., (w)_l^i\}$ by *Add-SS($(w)^i$)*. l is number of connected edge nodes.
- $u^k \in U_s^{t+1}$ generates global random numbers $(R)^k$ by *PRG(seed)*. u^k splits secret $(R)^k$ into l pieces $\{(R)_1^k, ..., (R)_l^k\}$ by *Add-SS($(R)^k$)*.
- $u^i \in U_s^t$ sends $\{(w)_1^i, ..., (w)_l^i\}$ to l edge nodes separately. $u^k \in U_s^{t+1}$ sends $\{(R)_1^k, ..., (R)_l^k\}$ to l edge nodes separately.

Edge Nodes

- Edge node e^j receive shares of model updates: $\{(w)_j^1, ..., (w)_j^p\}$ from p connected users in iteration t and shares of global random numbers: $\{(R)_1^j, ..., (R)_q^j\}$ from q users in next iteration $t + 1$.
- Edge node e^j computes the partial aggregated models of users selected in iteration $t + 1$. If edge node e^j has the share of global random number $(R)_j^k$ of user u^k, $(g)'^k_j = \sum_{i=1}^p (w)_j^i + (R)_j^k$. Otherwize, $(g)'^i_j = \sum_{i=1}^p (w)_j^i$.
- Edge nodes create transactions to record partial aggregated models $\{(g)'^k_j | k \in [0, q], j \in [0, n]\}$. When all transactions are recorded, they query transactions created by other edge nodes.
- Edge node e^j computes $(g)_j^k = \sum_{j=1}^n (g)'^k_j = (g) + (R)^k$ for user u^k.
- Edge nodes create transactions to record masked aggregated models $\{(g)_j^k | i \in [0, q], j \in [0, n]\}$.

Shared Ledger

- It receives transactions of $\{(g)'^k_j\}$, broadcasts and records $\{(g)'^k_j\}$.
- It receives transactions of $\{(g)_j^k\}$, broadcasts and records $\{(g)_j^k\}$.
- It verifies the calculation results of each edge node, marks the suspected malicious nodes, and obtains benign global models $\{(g)^k\}$ for $u^k \in U_s^{t+1}$.
- Edge nodes send the masked global model to connected users in the next iteration. User $u^k \in U_s^{t+1}$ can recover (g) by remove $(R)^k$ from $(g)^k$.
- It selects users $U_s^{t+2} \subseteq U$ for training in iteration $t + 2$ allocates corresponding edge nodes.

Fig. 4: SecAggregation of LiPFed.

without any accuracy loss. The security of Algorithm 1 and the correctness of secret recovery are proved in section V.

Additive secret sharing involves linear mapping and has lower computational complexity. Compared to FedAvg [1], LiPFed performs additive secret sharing on local models with a time complexity of $O(1)$. The time complexity of other privacy preserving methods is greater than or equal to LiPFed. Time complexity of Shamir secret sharing is $O(nlogn)$, where n is the number of shares. Time complexity of differential privacy relying on Gaussian noise to protect local models is $O(1)$. Paillier algorithm is a partially homomorphic encryption algorithm that supports additive homomorphism, time complexity of Paillier is $O(k^3)$, where k is the length of secret key.

C. Secure Decentralized Aggregation

Secure Local Model Aggregation. In SecAggregation, user u^i splits local model$(w)^i$ through the additive secret sharing algorithm (Algorithm 1) as $\{(w)_1^i, (w)_2^i, ..., (w)_l^i\}$ and sends them to multiple connected edge nodes. l is the number of connected edge nodes. To guarantee security, the value of l needs to be greater than $n/3$.

To protect the membership privacy of global model and enable global model verification, they are masked with a specific number. For all selected users for next iteration, user u^k generates a pseudorandom number $(R)^k$, and splits it into l parts as $\{(R)_1^k, (R)_2^k, ..., (R)_l^k\}$ using `Add-SS`$((R)^k)$ in Algorithm 1. Then they are sent to l edge nodes, respectively. Thus edge node e^j holds a shares of local models from p users: $\{(w)_j^1, (w)_j^2, ..., (w)_j^p\}$ and a shares of q global random numbers: $\{(R)_j^1, (R)_j^2, ..., (R)_j^q\}$. Then LiPFed aims to compute the partial aggregated models for the users selected in the next training iteration.

In the first aggregation, each edge node aggregates all shares of local models kept by itself and gets $\sum_{i=1}^p (w)_j^i$. Then a subset of them is covered with the shares of global random numbers $\{(R)_j^1, (R)_j^1, ..., (R)_j^q\}$. If edge node e^j has the share of global random number of the selected user u^i in the next iteration, the aggregated result is $(g)_j'^k = \sum_{i=1}^p (w)_j^i + (R)_j^k$, otherwise $(g)_j'^k = \sum_{i=1}^p (w)_j^i$. For all selected users in the next iteration, the same sum operations are performed. Then edge nodes create transactions to record the partial aggregated model set on the shared ledger. With the aid of blockchain, each edge node queries all other edge nodes' data. For edge node e^j, it aggregates the relevant masked partial aggregated models for the selected users connected to it.

In the second aggregation, edge node e^j aggregates partial aggregated models $(g)_j'^k$ from all edge nodes and get $(g)_j^k = \sum_{j=1}^n (g)_j'^k$ for user u^k. Algorithm 1 promises that $(g)_j^k = \sum_{j=1}^n (g)_j'^k = \sum_{y=1}^m \sum_{x=1}^l (w)_x^y + \sum_{x=1}^l (R)_x^k = (g) + (R)^k$

Authorized licensed use limited to: Iowa State University Library. Downloaded on March 06,2024 at 23:25:47 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Network Science and Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2024.3360311

is correct. Edge nodes create transactions to record the final computation results. For user u^i, l masked global models from different edge nodes are recorded on the shared ledger. After all edge nodes complete uploading aggregated results, a smart contract deployed on the blockchain is triggered to determine the existence of an isolating attack in the system. Finally, the smart contract selects benign masked global model $(g)^k$ for user u^k and returns it to each user. User u^k removes $(R)^k$ and gets the plain global model (g). Selected users start the next training iteration.

Offline Optimization. In each training iteration, all selected users in next training iteration have to share their global random numbers with the edge nodes. To reduce the calculation and communication overhead during the training process on the user side, the global random numbers of each user in each iteration are generated and shared in setup phase. It is in the offline phase and we call it offline optimization.

Specifically, for user u^i, it generates a random number set $\{(R)^i_{t=1}, (R)^i_{t=2}, ..., (R)^i_{t=T}\}$ as the global random numbers in the next t training iteration. Then the values in the set are split by Algorithm 1 and then sent to several edge nodes. In this way, edge nodes can directly use these shares of global random numbers in each iteration without extra online communication. In the experimental part, we pre-test the expected number of iterations for the model to reach convergence and use it as an initialization parameter.

Robustness to Users Dropping Out. LiPFed takes dropped out users into consideration. We claim that LiPFed can perform smoothly even some users drop out during the training process. In LiPFed, each user independently completes local model training, local model protection, and global random number elimination. Dropping out of users does not affect the operations of other users. Additionally, since additive secret sharing is only applied to the local model of each individual user, user dropping out does not affect the partial aggregation or the global aggregation at the edge nodes.

D. Data Format on Blockchain

The edge nodes calculate the partial aggregation model in the first aggregation, and then create transactions to record the above calculations. In the consortium blockchain, we deploy a pre-written smart contract to assist decentralized aggregation, which implements the following functions: partial aggregated models sharing and training user selection. As shown in Fig. 5, the following content is the format of the uploaded data.

Iteration Round	Edge Node ID	Masked Partial Global Model Set	User ID

Fig. 5: Formats of the Uploaded Data.

Iteration Round: During the training process, there are several iterations at which the data needs to be shared between blockchain nodes.

Edge Node ID: An unique identifier distinguishing an edge node from others. When an edge node calls the smart contracts, its address is automatically recorded in this field.

Masked Partial Aggregated Models Set: It is a variable set of the sum of the pseudorandom numbers of one user and the partial local models of all connected users. So the partial aggregation model is masked with random numbers which can guarantee data security when shared on the blockchain.

Users ID: It randomly selects k users in U to determine users participated in training in the next iteration. The chaincode sorts out all the uploaded user's ID number to achieve that. The mobile users selected in the next iteration are determined by the last contract. In this way, the edge nodes have the knowledge of for which users to do the partial aggregation without extra communication.

V. Security Analysis

In this section, we make a comprehensive security analysis of LiPFed. We first analyze the convergence of decentralized aggregation. Then, we demonstrate the security of LiPFed under two types of attack defined in threat model as Type-I Attack and Type-II Attack.

A. Convergence under Decentralized Aggregation

The convergence of FedAvg [1] has already been proved. Decentralized aggregation ensures privacy protection for models through additive secret sharing. Consequently, we demonstrate its convergence by proving that this method does not introduce accumulated noise when compared to FedAvg [1].

Lemma 1. *Assuming that the Additive Secret Sharing process is executed correctly under the semi-honest setting, then for any number v, and its shares of variables $v_1, v_2, \cdots, v_n$ calculated by* `Add-SS(`v`)` *using any set of pseudorandom numbers, we have:*

$$v = \sum_{i=0}^{n} v_i \tag{4}$$

Proof. Assuming that there is a private variable v, which is divided into multiple shares by additive secret sharing algorithm, and no information can be disclosed by each part. The combination of all shares can remove the disturbance and restore the variable. For example, there are n different edge nodes that each user intends to connect which means v will be split into n parts. The user needs to calculate the shares of variable v_i by following equation.

$$\begin{cases} v_i = v^* + r_i - r_{i+1}, & i \neq n \\ v_n = v^* + r_n - r_1, & i = n \end{cases} \tag{5}$$

where $v^* = v/n$, and $\{r_1, r_1, \cdots, r_n\}$ are pseudo random numbers generated at the same time.

For an entity that has all the shares of variables v_i, he can recover the user's private variable v by

$$v_{rec} = \sum_{i=0}^{n} v_i \tag{6}$$

According to Equation 5 and Equation 6, we achieve:

$$\begin{aligned} v_{rec} &= \sum_{i=0}^{n} v_i \\ &= v^* + r_1 - r_2 + ... + v^* + r_n - r_1 \\ &= v^* + ... + v^* + r_1 - r_1 + ... + r_n - r_n \\ &= v \end{aligned} \tag{7}$$

Authorized licensed use limited to: Iowa State University Library. Downloaded on March 06,2024 at 23:25:47 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Network Science and Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2024.3360311

From Equation 7, we prove n shares can recover v correctly. □

Theorem 1. *Assuming each user executes Additive Secret Sharing correctly and edge nodes adhere to decentralized aggregation protocol in a semi-honest setting.* (g) *is a global model aggregated of* p *local models by FedAvg, which means* $(g) = \sum_{i=1}^{p}(w)^i$ *and* $(g)_{rec}$ *is aggregated by LiPFed. For any local models* $(w)^i$ *of user* u^i*, and its shares of* $(w)^i_1, (w)^i_2, \cdots, (w)^i_l$*, then:*

$$(g)_{rec} = (g) \tag{8}$$

Proof. There are l different edge nodes that each user intends to connect which means $(w)^i$ is split into l parts. Edge node e_j holds $(w)^i_j$ of user u^i, which is a share of $(w)^i$ calculated by additive secret sharing algorithm. Edge node e_j holds $(R)^k_j$ of user u^k, which is a share of $(R)^i$ by an additive secret sharing algorithm. According to the decentralized aggregation protocol, user u^k can calculate global model $(g)_{rec}$ by following equation.

$$(g)_{rec} = (g)^k - (R)^k \tag{9}$$

According to Lemma 1, Equation 9 and decentralized aggregation protocol, we achieve:

$$\begin{aligned}(g)_{rec} &= (g)^k - (R)^k \\ &= \sum_{j=1}^{n}\sum_{i=1}^{p}(w)^i_j + \sum_{j=1}^{n}(R)^k_j - (R)^k \\ &= \sum_{i=1}^{p}(w)^i + (R)^k - (R)^k \\ &= (g)\end{aligned} \tag{10}$$

Through Equation 10, we prove that global model $(g)_{rec}$ in LiPFed is equal to global model (g) in FedAvg. Therefore, they share the same convergence properties. □

B. Defense against Isolating Attacks

Type-I Attack. We analyze the ability of LiPFed to defend against Type I Attack as defined in our threat model. The attack is most offensive when multiple edge nodes collude to infer membership privacy and return the same well-constructed global model to a specific user.

Assumption 1. *There exists* c *compromised edge nodes* $\{\mathcal{A}_i\}$ *under the isolating attacks who send the fake global model* $(w)^{fake}_i$ *to the victim* u^{victim}*.*

Assumption 2. *The isolating attack is the strongest when all compromised edge nodes collude, and send the same well-constructed global model* $(g)^{fake}$ *to the victim* u^{victim}*.*

Theorem 2. *Suppose Assumptions 1 and 2 hold. There are a total of* n *edge nodes, with* c *being the number of compromised edge nodes. When* $c < n/2$*, for any well-constructed global model* $(g)^{fake}$*, users can successfully defend against Type-I Attack isolating attacks.*

Proof. In LiPFed, there is a user connecting to n edge nodes, and the maximum compromised edge nodes c is $n/2 - 1$. After edge nodes aggregate the global model, assuming that these c compromised edge nodes $\{E_1, ..., E_c\}$ send same fake global model $(g)^{fake}$ to user u^{victim} connected with them. Meanwhile, some real global models $(g)^{real}$ are sent from honest edge nodes to u^{victim}. Thus, u^{victim} has received two kinds of different global models in this iteration which prove that he may be under *an isolating attack*. Then, the victim will accept the majority of the two global models. Therefore, if and only if $c \leq n/2 - 1$ is met, the user will choose the real model $(g)^{real}$ as the global model. □

C. Defense against Membership Inference Attacks

Type-II Attack. We analyze the security of LiPFed under Type-I Attack as defined in our threat model. The most rigorous definition currently widely accepted is described in Goldreich's paper [27], which establishes a trusted third-party T as the ideal model that can communicate securely with other users. Because this ideal model is in the most secure state, if the simulated output of the ideal model is indistinguishable from the real protocol, it proves that the real protocol has the same security as the ideal protocol. The adversary in the ideal model can conduct the same attack with the adversary in the real model.

Definition 2. *(Semi-honest Security): There is a* ***Protocol*** Π *between one user and multiple edge nodes to perform an aggregation operation.* ***Protocol*** Π *is secure when satisfying the following guarantees:*

1): For one semi-honest edge node, any share $(w)^i_j$ *of local model* $(w)^i$ *is computationally indistinguishable with a random nuber* r*.*

2): For $c < n$ *edge nodes colluding, the summation of shares of variables from them is computationally indistinguishable with a random value* r*, which means the compromised edge nodes cannot recover* $(w)^i$*. Where* n *represents the number of shares.*

Then, we prove that multiple semi-honest edge nodes collude, the adversary can not infer membership privacy about the training dataset.

Theorem 3. *Suppose there is a probabilistic polynomial time (PPT) adversary* $\mathcal{A}$*, who has a PPT simulator* SIM*, then for all* n, c, w_U, U, E, E_c*, and* C *such that* $C \subseteq E_C \subset E$*, the output of simulator* SIM *is indistinguishable from the output of* $\mathsf{REAL}^{U,n,c}_C$*, formulated as:*

$$\mathsf{REAL}^{U,n,c}_C((w)_U, U, E_c) \equiv_c \mathsf{SIM}^{U,n,c}_C((w)_C, U, E_c)$$

Proof. We use a standard hybrid argument to prove Theorem 3. We construct a simulator SIM that makes a series of modifications to LiPFed. Finally the view $\mathsf{SIM}^{U,n,c}_C$ is indistinguishable from the real view $\mathsf{REAL}^{U,n,c}_C$. The simulator runs as follows:

1) SIM chooses a uniform random tape for C.
2) SIM receives $\{(w)^i_j \mid i \in [1,p], j \in [1,c]\}, \{(R)^k_j \mid k \in [1,q], j \in [1,c]\}$ from C, which are the shares of values of local models and the shares of global random number respectively.

Authorized licensed use limited to: Iowa State University Library. Downloaded on March 06,2024 at 23:25:47 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Network Science and Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2024.3360311

3) SIM simulatively executes partial aggregation of $\{(w)_j^i \mid i \in [1,p], j \in [1,c]\}$ and add split global random number in $\{(R)_j^k \mid k \in [1,q], j \in [1,c]\} \cup \emptyset$
4) SIM calculates the sum of all partial aggregation results to obtain a global model (g) masked with $\{(R)^k \mid k \in [1,q]\}$.

In the following hybrid argument, through the modification of the original protocol, the simulator is computationally indistinguishable from the distribution of the real world execution.

Hyb_1: SIM has the input of all compromised edge nodes: $\{(w)_j^i \mid i \in [1,p], j \in [1,c]\}$ and $\{(R)_j^k \mid k \in [1,q], j \in [1,c]\}$. This corresponds to the real world distribution.

Hyb_2: In this hybrid, instead of sending data masked with r generating by **PRG** to edge nodes in SecAggregation, the simulator generates some random values whose distribution is identical with (r) by **PRG**.
Note that, $(w)^i$ is splitted as:

$$\begin{cases} (w)_j^i = (w)^i/l + (r)_i - (r)_{i+1}, & i \neq l \\ (w)_l^i = (w)^i/l + (r)_l - (r)_1, & i = l \end{cases} \tag{11}$$

Since $\{(r)_i \mid i \in [1,l]\}$ are all generating by **PRG**, $(w)^i/l + (r)_i - (r)_{i+1}$ are also random numbers. We replace them by uniformly random numbers. The security of the **PRG** makes this step computationally indistinguishable from the previous one.

Hyb_3: The compromised edge node E_c then gets all other edge nodes' (including honest and compromised edge nodes) aggregated data from blockchain ledger. They are all masked with uniformly global random values $\{(R)_j^k \mid k \in [1,q], j \in [1,c]\}$. The procedure for splitting $(R)^k$ is the same as that for $(w)^i$, which means SIM can easily generate an indistinguishable random value like previous. Then SIM simulates the aggregated masked global models. Same as the previous step, this step is computationally indistinguishable from previous.

Based on hybrid 1 to 3, the output of SIM is indistinguishable from REAL, completing the proof. □

VI. Performance Evaluation

In this section, image classification is used as a task to evaluate the performance of LiPFed from three aspects: the accuracy of the final model, the ability to defend against attacks, and training efficiency. We make a comparison with five methods to prove that the scheme is comparable in the above three aspects, especially the training efficiency.

A. Experimental Setup

FL has a broad range of applications, from training Gboard with a user base of tens of thousands to data collection and training utilizing several hundred IoT devices. In response to diverse scenario requirements, we categorize our user base into three groups: large-scale (10,000 users), medium-scale (1,000 users), and small-scale (100 users). Furthermore, to ensure FL doesn't disrupt a device's primary tasks, we select users whose devices are in an idle state for training. Typically, the percentage of users in an idle state fluctuates around 10%. As such, we've set the proportions of idle users at 5%, 10%, and 15% to accommodate these variations.

To demonstrate the superior performance of the LiPFed, we compared LiPFed with existing methods, mentioned in Section I, which all aim at protecting data privacy in FL process. **FedAvg** [1] is with no privacy preservation measure. **Partially-HE** [16] exploits Paillier cryptosystem to protect privacy. **CDPFed** [12] adds noise to models. **PSA** [11] and **VerifyNet** [10] cover local models with random numbers.

1) Experiment environments: Several users cooperate to train an image classifier with their local dataset securely. Machine learning models are implemented by Python 3.9 and Pytorch 1.9. The program runs on a PC (AMD Ryzen 5 2600X Six-Core processor at 3.60GHz and 32 GB RAM) to simulate the training process. Meanwhile, we implement the chain code by Go to run them on the consortium blockchain (Fabric 1.3) in a virtual machine set with 4GB memory.

2) Criterions: **Model accuracy**, **attack accuracy** and **training efficiency** are adopted to evaluate LiPFed. We compute model accuracy by equation $R = t_p/(f_n + t_p)$ as described in our design goals. Besides the security analysis to prove LiPFed theoretically, we train an attack model to verify the security of LiPFed. Isolating attack can be detected by comparison of aggregation results, which has been proved in Section V-A. We use the membership inference attack model proposed in [5] to verify the security of LiPFed experimentally, which is the mainstream attack model of FL privacy issues. We evaluate the efficiency of LiPFed from the time cost of total training and blockchain communication.

3) Dataset: To evaluate the performance of LiPFed, we used two datasets in the experience: MNIST[1] and CIFAR-10 [2]. The MNIST is a handwritten digit dataset of 60,000 28x28 grayscale images of the 10 digits, along with a test set of 10,000 images. The CIFAR-10 consists of 60,000 32x32 color images in 10 different classes, with 6,000 images per class, and 10,000 images reserved for testing. To ensure the consistency of the LiPFed and other comparison methods in the training process, experiments about accuracy and time-consuming are implemented on the MNIST dataset. To evaluate the ability of different methods to defend against membership inference attacks, we carried out attack experiments on the CIFAR-10 dataset. We assume that the classification task requires a total of 50,000 to 60,000 data for training, which are either independently and identically distributed (IID) or non-IID across different users. For IID data, we randomly and equally distribute all data to all users, while for non-IID data, the label and amount distribution is different among different users.

B. Performance of LiPFed with Varying Parameters

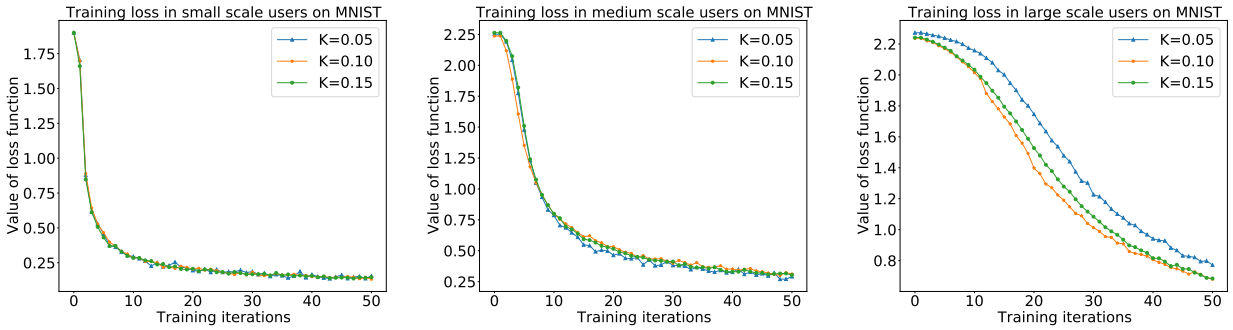
In this section, we evaluate the performance of LiPFed by changing various parameters to obtain the accuracy and training efficiency of LiPFed in various cases.

1) Accuracy: We set the percentage of users whose devices are idle at 5%, 10% and 15% respectively. We explore the relationship between accuracy and user scale, the percentage

[1]http://yann.lecun.com/exdb/mnist/
[2]https://www.cs.toronto.edu/ kriz/cifar.html

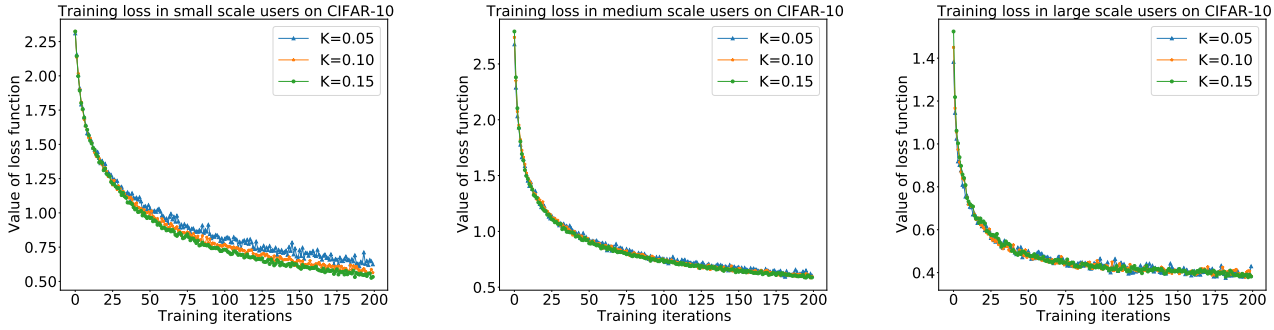
Authorized licensed use limited to: Iowa State University Library. Downloaded on March 06,2024 at 23:25:47 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Network Science and Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2024.3360311



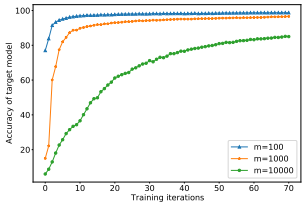
(a) (b) (c)

Fig. 6: Value of loss function of MNIST classifier trained by CNNs model when selecting different proportions of users among different scales of users. (a) shows results in small scale (i.e., $m = 100$), with the increase of training iterations. (b) shows results in medium scale (i.e., $m = 1,000$), with the increase of training iterations. (c) shows results in large scale (i.e., $m = 10,000$), with the increase of training iterations.

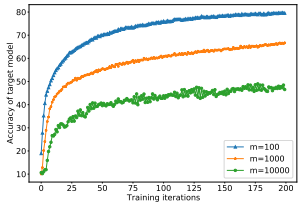


(a) (b) (c)

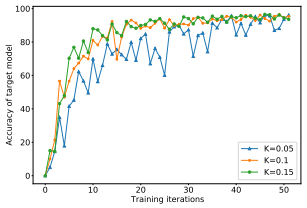
Fig. 7: Value of loss function of CIFAR-10 classifier trained by ResNet18 when selecting different proportions of users among different scales of users. (a) shows results in small scale (i.e., $m = 100$), with the increase of training iterations. (b) shows results in medium scale (i.e., $m = 1,000$), with the increase of training iterations. (c) shows results in large scale (i.e., $m = 10,000$), with the increase of training iterations.



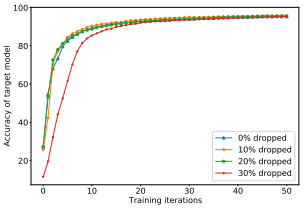
(a) Accuracy of MNIST classifier trained by CNNs model.



(b) Accuracy of CIFAR-10 classifier trained by ResNet18.



(c) Accuracy under non-iid MNIST.



(d) Accuracy under dropped users.

Fig. 8: Accuracy of model under different scenarios.

of selected users, and the quality of datasets (IID or non-IID). We use Convolutional Neural Networks (CNNs) and ResNet18 [28] to train classifiers for MNIST and CIFAR-10 respectively. The accuracy of experimental results is as follows.

- Fig. 6 and Fig. 7 show the loss of MNIST classifier and CIFAR-10 classifier during training process, respectively. Accordingly, Fig. 8(a) and Fig. 8(b) show the accuracy of the model. We observe LiPFed performs well in small and medium scale cases, where the model is trained well with only a few iterations. In large scale case, more training iterations are needed and the final model accuracy is not as high as that in the other cases, because each user holds fewer data examples. The more users participate in training, the more data examples are involved.
- We consider the case of Non-IID data, which often exists in reality. Fig. 8(c) shows the results on Non-IID MNIST datasets. Although the model accuracy rises in the fluctuation, LiPFed is still practical and the accuracy maintains at a high level, no matter in which scale case.
- LiPFed takes into dropped users which means that it does not require users to be online all the time. Fig. 8(d) shows the model accuracy under different proportions of dropped users. Even if there are dropped users, the impact on the final accuracy of the model can be almost ignored.

As we can see, LiPFed trains high-accuracy models in scenarios of users of different scales, the existence of Non-IID datasets and dropped users, which means that it can be widely applied in various fields.

2) Efficiency: We evaluate training time overhead of LiPFed from training time on user side and aggregation time on the aggregator side (edge node side). Besides, we evaluate the communication time overhead in blockchain, which is related to the latency of invoking smart contracts to share and query data and communication between users and edge nodes.

- From Table III we can infer that the percentage of the

Authorized licensed use limited to: Iowa State University Library. Downloaded on March 06,2024 at 23:25:47 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Network Science and Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2024.3360311

TABLE III: Training Time (s) on User Side (US) and Aggregation Side (AS)

Number of Users	Percentage of Selected Users	Number of Edge Nodes											
		4			7			10			13		
		US	AS	Total	US	AS	Total	US	AS	Total	US	AS	Total
100	5%	3.05	0.001	3.051	3.39	0.001	3.391	3.03	0.001	3.031	3.01	0.001	3.011
	10%	2.97	0.001	2.971	3.13	0.001	3.131	3.08	0.001	3.081	3.02	0.001	3.021
	15%	3.02	0.001	3.021	3.03	0.001	3.031	3.12	0.001	3.121	3.14	0.001	3.141
1,000	5%	0.30	0.04	0.34	0.31	0.04	0.35	0.35	0.05	0.40	0.32	0.04	0.36
	10%	0.31	0.04	0.35	0.31	0.04	0.35	0.38	0.04	0.42	0.37	0.05	0.42
	15%	0.32	0.04	0.36	0.30	0.05	0.35	0.35	0.05	0.40	0.32	0.04	0.36
10,000	5%	0.04	0.30	0.34	0.045	0.29	0.335	0.045	0.32	0.365	0.047	0.29	0.337
	10%	0.04	0.32	0.36	0.045	0.33	0.375	0.04	0.32	0.36	0.05	0.37	0.42
	15%	0.04	0.37	0.41	0.04	0.4	0.44	0.04	0.32	0.36	0.047	0.35	0.39

TABLE IV: Blockchain Communication Time (s) per Iteration

Operations	Number of Edge Nodes			
	4	7	10	13
Upload (s)	0.283	0.34	0.38	0.413
Query (s)	0.083	0.086	0.088	0.087
Total (s)	0.366	0.426	0.468	0.500

selected users has little effect on the training time of both the user and aggregator side. This is because the process of additive secret sharing by a user does not require communication with other entities. It is evident that user side training is conducted with minimal time cost. Additionally, in all scenarios, the time cost generated by the aggregation process on the Aggregation side is also very low. This is primarily because the edge nodes solely focus on aggregating without performing resource-intensive tasks such as decryption or decoding.

- From Table IV, when the number of edge nodes increases, the overhead of both uploading and querying increases, which is caused by the characteristics of the consortium blockchain. However, since the local aggregation process of each blockchain node is independent of each other in parallel, there will be no significant changes in time.

C. Comparison with Existing Methods

We compare the LiPFed with the existing methods from three aspects: ability to defend against attack, model accuracy, and training efficiency. The comparison results are as follows:

1) Comparison Results in Defense of Membership Inference Attack: Membership inference attacks can attack both the local and global models. Therefore, comparative experiments include testing the attack accuracy of the membership inference attack on the local model and the attack accuracy on the global model. We employ a membership inference attack proposed in [5] to infer the membership privacy of LiPFed and five comparative methods. It is a white-box membership inference attack that exploits privacy vulnerabilities in SGD algorithm. The architecture combines gradients extracted from different layers of the target model to compute the membership probability of the target data point. The results are summarized in Table V. In CDPFed, since the distortion is allocated by the server, the server obtains the plaintext of each user's local model and obtains a high attack accuracy as unprotected FedAvg. Similarly, iVerifyNet and PSA ignore the protection of the global model. In Partially-HE, the attack accuracy is at a low level as the attacker can only get encrypted the local and global model. In LiPFed, the attack accuracy remains low on both local and global models, which indicates LiPFed's defense is comprehensive and effective. This is because LiPFed provides protection for both local and global models, and the attacker can only obtain the perturbed models.

2) Comparison Results in Accuracy: We conduct experiments to compare the accuracy of the final model trained by LiPFed with other protection methods and the FedAvg without protection. We get the highest accuracy and list the results in Table VI. The accuracy of the model protected by the LiPFed has the same level as Partially-HE, VerifyNet, and PSA. The commonness of these methods is that the methods they employed will not cause any loss of model accuracy. It is worth noting that the training effect of the CDPFed model is not ideal. We set the same privacy budget in our experiment with CDPFed in [12]. The only difference from [12] is that we replace the training model with a CNNs model which is the same as other methods. Experiments show that when the differential privacy budget is exhausted, the model accuracy has not reached convergence.

3) Comparison Results in Efficiency: We present time overhead of different methods on user side and aggregation side respectively in Table VII. Specifically, CDPFed continues training until the privacy budget boundary is reached. LiPFed has obvious advantages in time overhead on user side. It outperforms Partially-HE which needs complex homomorphic operations on user side. Compared with VerifyNet and PSA, LiPFed still has less computation time on user side. Time overhead on aggregation side is comparable to other methods. With the scale of users getting larger, LiPFed has a small increase in total time cost, but the total time overhead of Partially-HE, DPFed, VerifyNet, and PSA increases significantly in large-scale scenarios

Summary. Through extensive experimental results on the performance of the LiPFed and the comparison with other privacy preserving methods, we can claim that LiPFed shows superiority over the existing schemes in terms of privacy protection against membership inference attacks, model accuracy, and especially training overhead on the user side.

VII. Related Work

Attacks on membership privacy and defense have attracted increasing research attention in recent years. In this section,

Authorized licensed use limited to: Iowa State University Library. Downloaded on March 06,2024 at 23:25:47 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Network Science and Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2024.3360311

TABLE V: Attack Accuracy (%) of Membership Inference Attack on Local Model (LM) and Global Model (GM) on Different Defense Schemes

Methods	100 Users		1,000 Users		10,000 Users	
	Attack on LM	Attack on GM	Attack on LM	Attack on GM	Attack on LM	Attack on GM
FedAvg [1]	82.8	88.6	87.4	89.7	82.3	86.3
Partially-HE [16]	46.2	45.3	45.4	46.7	44.9	43.5
CDPFed [12]	89.3	57.1	83.8	57.5	86.4	55.5
VerifyNet [10]	46.5	89.2	49.0	89.1	51.2	85.9
PSA [11]	42.3	89.1	48.6	89.1	50.3	86.1
LiPFed	**48.0**	**47.2**	**48.9**	**46.7**	**49.7**	**48.5**

[1] In the experience of CDPFed, the setting of the privacy budget (δ) is the same as that of CDPFed in [12]. δ is equal to $e-3$ in the scenario of 100 users, δ is equal to $e-5$ in the scenario of 1,000 users and δ is equal to $e-6$ in the scenario of 10,000 users.

TABLE VI: Accuracy (%) of Global Models of Different FL Schemes

Methods	Number of Users		
	100	1,000	10,000
FedAvg [1]	98.53	96.55	92.07
Partially-HE [16]	98.01	96.13	91.51
CDPFed [12]	47.95	35.88	12.74
VerifyNet [10]	98.04	96.26	91.13
PSA [11]	98.12	96.30	91.28
LiPFed	**98.47**	**95.64**	**91.96**

TABLE VII: Comparison of Computational Time (s) on User Side (US) and Aggregation Side (AS)

Methods	Number of Users					
	100		1,000		10,000	
	US	AS	US	AS	US	AS
FedAvg [1]	2.380	0.002	0.270	0.004	0.030	0.013
Partially-HE [16]	34.940	0.001	35.210	0.113	53.120	1.009
CDPFed [12]	2.370	1.637	9.320	1.893	79.970	10.350
VerifyNet [10]	16.570	0.002	16.380	0.009	19.650	0.084
PSA [11]	17.010	0.012	17.580	0.024	19.620	0.090
LiPFed	**3.050**	**0.366**	**0.310**	**0.406**	**0.040**	**0.736**

we briefly review the existing attacks and defenses.

A. Attacks on Membership Privacy in FL

Membership inference stands as one of the most severe security threats in FL today. Based on the attacker's knowledge and access to the target model, membership inference attacks can be classified into two types: white-box attacks and black-box attacks. Nasr *et al.* [5] first introduce white-box membership inference attacks, where attackers have complete access to the structure and parameters of the FL model. This level of access allows attackers to compute gradients of the objective function with respect to the information source input. They can analyze model outputs, weights, and other information to determine whether a specific training sample is included in the model. Leino and Fredriksonden *et al.* [6] point out that the capabilities of attackers proposed in [5] are too strong and deviate from realistic scenarios. Therefore, in [6], attackers are assumed to know a significant portion of the target model's private training dataset and possess a shadow dataset about the private training data.

Shokri *et al.* [7] conduct membership inference attacks in a black-box environment, where attackers can only observe the input and output of the target model without accessing its internal details or gradients. Attackers infer whether the model contains a specific training sample by observing the model's outputs and other publicly available information, such as prediction results. Shokri *et al.* initially focus on attacks against binary classifiers. Ye *et al.* [8] provide interpretability to membership inference attacks. Liu *et al.* [9] judge member and non-member data by evaluating the sensitivity of different data to the target model.

The aggregation of local models reduces the contribution of individual user's local model. To enhance the inference of user's membership privacy in FL, the aggregator launches isolating attacks [5] by individually sending carefully crafted fake models to victim users. This approach allows attackers to gain more information about local models, increasing the success rate of membership inference attacks in FL.

B. Privacy preserving FL

To defend against privacy attacks in FL, various privacy preserving methods have been proposed. Differential Privacy-based FL protects sensitive information by introducing perturbations to the model, offering lightweight computation but potentially impacting convergence and availability of global model. Centralized differential privacy [12] involves the aggregator adding noise to local models, while local differential privacy methods [13, 14] have users add noise to their own local models. Hybrid differential privacy [15, 29] integrates local and centralized differential privacy, wherein the role of adding perturbations to local models is chosen based on the credibility from users to the aggregator. Homomorphic Encryption (HE)-based FL [16–19, 30] allows computations on encrypted models or parameters without decryption, but incurs substantial computational and communication overhead. Efficient algorithms and hardware support are crucial to maintain performance. Commonly used secret sharing schemes (SSS) include Shamir Secret Sharing and Additive Secret Sharing. Shamir Secret Sharing-based FL [10, 11] utilizes non-linear mapping for model reconstruction. Model parameters are partitioned into multiple shares and distributed to other users. The original local model can only be reconstructed when

Authorized licensed use limited to: Iowa State University Library. Downloaded on March 06,2024 at 23:25:47 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Network Science and Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2024.3360311

a sufficient number of shares are collected. Additive Secret Sharing-based FL involves linear mapping for model reconstruction. For example, ABY [26] supports two-party secure computation. Trusted Execution Environment (TEE) [31, 32] provides a hardware-level protected execution environment for FL. It safeguards the privacy of local model training and parameter aggregation, ensuring that the code and data running within it are highly isolated and protected.

VIII. Conclusion

In this paper, we proposed LiPFed, a computationally lightweight privacy preserving FL scheme by secure decentralized aggregation. LiPFed introduced consortium blockchain to decentralize the authority of a central server to multiple edge nodes. It designed and employed an additive secret sharing algorithm to protect the privacy of both local and global models throughout the FL process. Security analysis theoretically proved the security of this approach. Extensive experiments demonstrated that LiPFed can ensure model's accuracy, protect data sufficiently, and improve efficiency significantly.

In our future works, we plan to extend our work in two aspects. First, we will reduce communication overhead between users and edge nodes. Second, we will enhance the robustness to defend against malicious participants.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[2] H. Zhou, Y. Zheng, H. Huang, J. Shu, and X. Jia, "Toward robust hierarchical federated learning in internet of vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 5, pp. 5600–5614, 2023.

[3] B. Ghimire and D. B. Rawat, "Recent advances on federated learning for cybersecurity and cybersecurity for federated learning for internet of things," *IEEE Internet Things J.*, vol. 9, no. 11, pp. 8229–8249, 2022.

[4] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *arXiv preprint arXiv:1811.03604*, 2018.

[5] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in *2019 IEEE Symposium on Security and Privacy (SP)*, 2019, pp. 739–753.

[6] K. Leino and M. Fredrikson, "Stolen memories: Leveraging model memorization for calibrated white-box membership inference," in *29th USENIX security symposium (USENIX Security 20)*, 2020, pp. 1605–1622.

[7] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 3–18.

[8] J. Ye, A. Maddi, S. K. Murakonda, V. Bindschaedler, and R. Shokri, "Enhanced membership inference attacks against machine learning models," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 3093–3106.

[9] L. Liu, Y. Wang, G. Liu, K. Peng, and C. Wang, "Membership inference attacks against machine learning models via prediction sensitivity," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 3, pp. 2341–2347, 2023.

[10] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, "Verifynet: Secure and verifiable federated learning," *IEEE Transactions on Information Forensics and Security*, vol. 15, no. 99, pp. 911–926, 2020.

[11] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, p. 1175–1191.

[12] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," *arXiv: Cryptography and Security*, 2017.

[13] C. Wang, X. Wu, G. Liu, T. Deng, K. Peng, and S. Wan, "Safeguarding cross-silo federated learning with local differential privacy," *Digital Communications and Networks*, vol. 8, no. 4, pp. 446–454, 2022.

[14] Y. Zhao, J. Zhao, M. Yang, T. Wang, N. Wang, L. Lyu, D. Niyato, and K.-Y. Lam, "Local differential privacy-based federated learning for internet of things," *IEEE Internet of Things Journal*, vol. 8, no. 11, pp. 8836–8853, 2021.

[15] B. Avent, A. Korolova, D. Zeber, T. Hovden, and B. Livshits, "Blender: Enabling local search with a hybrid differential privacy model," in *26th USENIX Security Symposium (USENIX Security 17)*, 2017, pp. 747–764.

[16] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1333–1345, 2018.

[17] X. Tang, M. Shen, Q. Li, L. Zhu, T. Xue, and Q. Qu, "Pile: Robust privacy-preserving federated learning via verifiable perturbations," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 6, pp. 5005–5023, 2023.

[18] Z. Ma, J. Ma, Y. Miao, Y. Li, and R. H. Deng, "Shieldfl: Mitigating model poisoning attacks in privacy-preserving federated learning," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 1639–1654, 2022.

[19] N. M. Hijazi, M. Aloqaily, M. Guizani, B. Ouni, and F. Karray, "Secure federated learning with fully homomorphic encryption for iot communications," *IEEE Internet of Things Journal*, pp. 1–1, 2023.

[20] L. S. Sankar, M. Sindhu, and M. Sethumadhavan, "Survey of consensus protocols on blockchain applications," in *2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS)*, 2017, pp. 1–5.

[21] M. Shen, J. Duan, L. Zhu, J. Zhang, X. Du, and M. Guizani, "Blockchain-based incentives for secure and collaborative data sharing in multiple clouds," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 6, pp. 1229–1241, 2020.

[22] M. Shen, H. Liu, L. Zhu, K. Xu, H. Yu, X. Du, and M. Guizani, "Blockchain-assisted secure device authentication for cross-domain industrial iot," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 5, pp. 942–954, 2020.

[23] Y. Liu, S. Liu, Y. Wang, F. Lombardi, and J. Han, "A survey of stochastic computing neural networks for machine learning applications," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 7, pp. 2809–2824, 2020.

[24] O. Goldreich, *Foundations of Cryptography Volume II Basic Applications*. The Edinburgh Building, Cambridge CB2 8RU, UK: Cambridge University Press, 2009.

[25] Y. Long, V. Bindschaedler, L. Wang, D. Bu, X. Wang, H. Tang, C. A. Gunter, and K. Chen, "Understanding membership inferences on well-generalized learning models," *arXiv preprint arXiv:1802.04889*, 2018.

[26] D. Demmler, T. Schneider, and M. Zohner, "Aby-a framework for efficient mixed-protocol secure two-party computation." in *22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, February 8-11, 2015*, 2015.

[27] O. Goldreich, "Foundations of cryptography: Basic tools, cambridge u," 2001.

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[29] T. Wang, J. Q. Chen, Z. Zhang, D. Su, Y. Cheng, Z. Li, N. Li, and S. Jha, "Continuous release of data streams under both centralized and local differential privacy," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 1237–1253.

[30] M. Shen, J. Wang, H. Du, D. Niyato, X. Tang, J. Kang, Y. Ding, and L. Zhu, "Secure semantic communications: Challenges, approaches, and opportunities," *IEEE Network*, pp. 1–1, 2023.

[31] F. Mo, H. Haddadi, K. Katevas, E. Marin, D. Perino, and N. Kourtellis, "Ppfl: privacy-preserving federated learning with trusted execution environments," in *Proceedings of the 19th annual international conference on mobile systems, applications, and services*, 2021, pp. 94–108.

[32] Y. Zheng, S. Lai, Y. Liu, X. Yuan, X. Yi, and C. Wang, "Aggregation service for federated learning: An efficient, secure, and more resilient realization," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 2, pp. 988–1001, 2023.

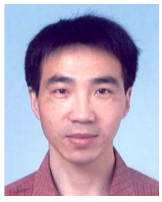
Authorized licensed use limited to: Iowa State University Library. Downloaded on March 06,2024 at 23:25:47 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Network Science and Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2024.3360311

Meng Shen (Member, IEEE) is a Professor at Beijing Institute of Technology, Beijing, China. He received the B.Eng degree from Shandong University, Jinan, China in 2009, and the Ph.D. degree from Tsinghua University, Beijing, China in 2014, both in computer science. His research interests include data privacy and security, blockchain applications, and encrypted traffic classification. He has authored over 50 papers in top-level journals and conferences, such as ACM SIGCOMM, IEEE JSAC, and IEEE TIFS. He has guest edited special issues on emerging technologies for data security and privacy in IEEE Network and IEEE Internet-of-Things Journal. He received the Best Paper Runner-Up Award at IEEE IPCCC 2014 and IEEE/ACM IWQoS 2020. Dr. Shen was selected by the Beijing Nova Program 2020 and was the winner of the ACM SIGCOMM China Rising Star Award 2019. He is a member of the IEEE.

Jing Wang received the B.Eng degree in computer science from Beijing University of Technology, Beijing, China in 2021. Currently she is a master student in the School of Cyberspace Science and Technology, Beijing Institute of Technology. Her research interests include machine learning privacy and blockchain applications.

Jie Zhang received the B.Eng degree in computer science from China University of Mining and Technology, Jiangsu, China in 2018, and the M.Sc degree in computer science in Beijing Institute of Technology, Beijing, China in 2021. His research interests include blockchain applications and machine learning privac5y.



Qinglin Zhao (Senior Member, IEEE) received the B.S. degree in mathematics education from the Hubei University, Wuhan, China, in 1998, the M.S. degree in applied mathematics from the Huazhong University of Science and Technology, Wuhan, in 2001, and the Ph.D. degree in computer architecture from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2005. From May 2005 to August 2009, he worked as a Postdoctoral Researcher with The Chinese University of Hong Kong and the Hong Kong University of Science and Technology. Since September 2009, he has been with the School of Computer Science and Engineering, Macau University of Science and Technology, where he is currently a Professor. He has published more than 80 peer-reviewed papers, including 20 IEEE Transactions papers, held more than 30 patents, including eight U.S. patents. His research interests include blockchain and decentralization computing, machine learning, Internet of Things, wireless communications and networking, cloud/fog computing, and software-defined wireless networking. He received the BOC Excellent Research Award of Macau University of Science and Technology in 2011 and 2015.

Bohan Peng received the B.Eng degree in science in electronic information engineering from Beijing Institute of Technology, Beijing, China in 2022. Currently, he is a Ph.D. student in the School of Cyberspace Science and Technology, Beijing Institute of Technology. His research interests include machine learning security and robust distributed learning.

Tong Wu received her Ph.D. degrees in computer science from University of Wollongong, Australia, in 2020. She is currently a postdoctoral research fellow at School of Cyberspace Science and Technology, Beijing Institute of Technology. Her research interests include applied cryptography, cloud security, security and privacy in blockchain.

Liehuang Zhu (Senior Member, IEEE) is a professor at the Department of Cyberspace Science and Technology at Beijing Institute of Technology. He is selected into the Program for New Century Excellent Talents in University from the Ministry of Education, P.R. China. His research interests include Internet of Things, Cloud Computing Security, Internet and Mobile Security.

Ke Xu (Fellow, IEEE) received his Ph.D. from the Department of Computer Science and Technology of Tsinghua University, Beijing, China, where he serves as a Full Professor. He has published more than 200 technical papers and holds 11 US patents in the research areas of next-generation Internet, Blockchain systems, Internet of Things, and network security. He is a member of ACM and senior member of IEEE. He has guest-edited several special issues in IEEE and Springer Journals. He is an editor of IEEE IoT Journal and has served as a steering committee chair of IEEE/ACM IWQoS.

Authorized licensed use limited to: Iowa State University Library. Downloaded on March 06,2024 at 23:25:47 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.