SCFL:A Robust Privacy-Preserving Federal Learning Model Based on Secure Clustering

1st Lida Xiang, 2nd Jinglin Zeng, 3rd Xiaofen Wang, 4th Yongqiang Wang
University of Electronic Science and Technology of China
Chengdu, China
xfwang@uestc.edu.cn
5th Bo Zhang, 6th Xiaosong Ding
China Telecom Sichuan Branch
Chengdu, China
7th Lei Zheng
China Mobile Internet Co., LTD
Guangzhou, China

***Abstract*—Federated learning(FL) has been developed as a distributed machine learning which utilizes data distributed across various terminals. In FL, as the gradient of each client is shared, privacy leakage problems arise, which have led to the development of privacy-preserving federated learning(PPFL) as an emerging secure FL approach. Nevertheless, current PPFL methods, particularly for nonIID data, encounter challenges such as heavy computation and communication overhead. In addition, they are susceptible to model poisoning attacks. To address this problem, we design a cosine similarity-based defense strategy for DBSCAN clustering and a secure federated learning model based on secure clustering (SCFL), which resists encrypted model poisoning attacks and protects privacy. We first construct KD-tree based on the cosine similarity between local gradients. The DBSCAN clustering based on the KD-tree acceleration can be applied to detect malicious gradients and address data heterogeneity. We propose Byzantine-tolerance aggregation using cosine similarity, and we show this technique achieves robustness for both IID and nonIID data settings. The experiment results show that SCFL outperforms the prevailing defense strategy such as ShieldFL in terms that SCFL reduces communication cost by about 50% and encryption and decryption runtime by about 80%, and achieves 5%-10% accuracy improvement.**

***Index Terms*—Privacy-Preserving, Federated Learning, Homomorphic Encryption, Secure Clustering, Model Poisoning Attack**

I. Introduction

Federated learning [5] is an advanced distributed machine learning framework that effectively solves the data silo problem by allowing a global model to be trained with locally trained gradients without sharing data. However, federated learning faces multiple privacy threats from malicious users or curious centralized servers. To cope with privacy leakage problems, multiple privacy-preserving strategies [23] are proposed by using cryptographic primitives or differential privacy techniques.

However, the existing PPFL schemes have some shortcomings and potential security risks [3], [20]. Some schemes [15] that apply homomorphic encryption or multiparty secure computation suffer from high computational or communication overhead. Differential privacy technique [4], [19] makes the model accuracy declined. Due to the invisibility of local training, PPFL is susceptible to model poisoning attacks [2]. The model poisoning attacks are initiated by a Byzantine adversary who manipulates an arbitrary percentage of malicious users to submit toxic local model gradients, thus making the global model deviate from the correct trend. Model poisoning attacks are classified into targeted attacks and untargeted attacks [8]. The existing defense strategies [22] against poisoning attacks will compromise privacy in PPFL as they need to access local gradients. Furthermore, due to the unpredictability of heterogeneous data (especially nonIID data), PPFL inherently suffers from non-negligible local bias, making it more difficult to detect and defend against model poisoning attacks [15].

To address these issues, we propose secure federated learning model based on secure clustering (SCFL), which has a privacy-preserving defense strategy against model poisoning attacks. In SCFL, a cosine similarity-based secure clustering strategy is designed to detect model poisoning attacks against the PPFL that is based on a homomorphic encryption, and a robust Byzantine-tolerant aggregation mechanism is constructed for heterogeneous data scenarios. The main contributions are as follows.

- We propose a cosine similarity-based secure clustering defense strategy against encrypted model poisoning attacks, which can identify encrypted malicious gradients based on cosine similarity while guaranteeing privacy protection and efficient clustering.
- We design a Byzantine-tolerant aggregation mechanism to make SCFL robust against Byzantine attacks, which supports heterogeneous data scenarios including IID and nonIID data settings, and it makes SCFL maintain high model accuracy if no more than 50% users are compromised.
- We evaluate SCFL on two real datasets against model poisoning attacks. The results of the experiment show that SCFL can recognize poisoned gradients even when they are encrypted, and it achieves better accuracy and higher

efficiency compared to state-of-the-art PPFL scheme.

II. Related Work

To resist privacy leakage in federated learning, privacy-preserving federated learning schemes [5], [14] are developed in recent years. Google first proposed a PPFL [5], which is based on stochastic gradient descent (SGD). In 2019, WeBank's AI team released the self-developed Federated Learning Open Source Project FATE [14](Federated AI Technology Enabler), which provides a distributed secure computing framework and provides privacy computing support for machine learning and migration learning algorithms.

The PPFL schemes that can protect data and model privacy are susceptible to model poisoning attacks [9]. By injecting encrypted poisoned updates instead of injecting poisoned samples, model poisoning attack is more stealthy and powerful than data poisoning attack. Model poisoning attacks can be classified into two categories, namely untargeted attacks [8] and targeted attacks [2]. Researchers have proposed many solutions to resist model poisoning attacks in FL. Median-based defense strategy [22] is based on a median statistics approach for Byzantine failures, where the median of the local gradient is computed as the global update in the FL. Bulyan [11] uses Krum [12] to identify outliers using Euclidean distance, and the gradient that has the closest distance to its neighboring gradients is selected as the global gradient. Auror [17] uses k-means to make local gradients' clustering and identify the outliers. Sybils [9] uses cosine similarity to identify the outliers. Since local gradients trained on nonIID data have different distributions [21], these approaches may make wrong classification of benign local updates trained on nonIID data and the outliers detected are not correct. Ma et al. propose the ShieldFL scheme based on Two-Trapdoor Homomorphic Encryption [15], which can resist encrypted model poisoning and achieve robust aggregation without compromising privacy. However, this scheme suffers from high computation and communication overhead, and its model accuracy greatly declines when the poisoning rates goes higher.

Resisting encrypted model poisoning attacks in both IID and nonIID data scenarios in PPFL is a major challenge. As shown in **Table 1**, most of the existing strategies cannot effectively defend against poisoning attacks due to the limited ability in distinguishing benign from malicious gradients and they do not provide robust aggregation. Although ShieldFL [15] is capable to resist poisoning attacks in both IID and nonIID data scenarios, but it is not efficient enough for practical FL.

III. Preliminaries

A. Federated Learning

Federated Learning (FL) is a distributed machine learning architecture, which assumes that heterogeneous data is distributed among n users, each of which trains local gradients using stochastic gradient descent(SGD) and then uploads them to the server for aggregation and updating the global model. The model reaches convergence after multiple iterations of training.

1) Local training.: In the t-th training iteration, each user trains new iteration of local gradient $g_i^{(t)}$ based on individual local data $D_i (i \in [1, n])$ and the global model $g^{(t-1)}$ of the previous iteration assigned by the server as follows

$$g_i^{(t)} = g_i^{(t-1)} - \eta \cdot \nabla J_i(g^{(t-1)}, D_i) \tag{III.1}$$

$$J(g^{(t-1)}, D) = \frac{1}{|D|} \sum J(g^{(t-1)}, D) \tag{III.2}$$

where η is the learning rate in local training.

2) Aggregation.: The server aggregates the gradient $g_i^{(t)}$ $(i \in [1, n])$ uploaded by the users to compute the global gradient $g^{(t)}$ as follows

$$g^{(t)} = \frac{1}{n} \sum_{i=1}^{n} g_i^{(t)} \tag{III.3}$$

In the PPFL system, to prevent the privacy leakage, each user's local gradient is encrypted before being uploaded to the server and the aggregation process is conducted within an encrypted manner. In our scheme, Joint Homomorphic Encryption(JHE) is used in PPFL.

B. KD-DBSCAN

KD-DBSCAN [18] can cluster data faster than other methods, utilizing KD-tree to divide them and accelerating the finding of neighboring nodes. Since the KD-tree is constructed based on data attributes, data objects with high similarity are more closely connected in the KD-tree. Compared to DBSCAN [6], [7], KD-DBSCAN exhibits a reduction of time complexity from $O(n^2)$ to $O(nlog_2 n)$.

1) Build KD-Tree.: Initially, the data points are sorted according to the first dimension, and the median value is selected as the root node to create the left and right subtrees. Compared with the root node, the nodes of the left subtree have smaller values, while the right subtree has larger node values. The process of constructing the left and right subtrees is repeated from the second dimension until no further subdivision is possible.

2) KD-DBSCAN Cluster.: In KD-DBSCAN Cluster, there are two steps, which are shown in the following.

a) Search neighbor node.: The KD-tree compares the value at each dimension with the boundary range eps, and finds the nearest neighbor node [13]. It also calculates the distance between the two points based on the Euclidean distance and adds it to the data set that satisfies

$$\mathbf{Dist}(KnownNode, SearchedNode) \leq eps. \tag{III.4}$$

where $KnownNode$ is the current cluster core point and $SearchedNode$ is the node in the KD-tree being searched.

b) Cluster.: When the number of nodes in the dataset found in the previous step is no less than $MinPts$, these nodes are classified into the same cluster.

The above two processes are continued until all the nodes are traversed.

TABLE I
A COMPARATIVE SUMMARY OF EXISTING DEFENSE STRATEGIES

Approach	Data Setting	Defense Strategy	Privacy	Byzantine Tolerance
Median [22]	IID	median statistics	×	×
Krum [12]	IID	Euclidean distance	×	×
Auror [17]	IID	k-means	×	×
Sybils [9]	nonIID	cosine similarity	×	✓
ShieldFL [15]	both	cosine similarity	✓	✓
SCFL	both	DBSCAN & cosine similarity	✓	✓

IV. JOINT HOMOMORPHIC ENCRYPTION

In this section, we describe the process of Joint Homomorphic Encryption (JHE) in detail and its full homomorphism.

A. JHE Scheme

1) KeyGen(λ).: The system's public parameters are denoted by p,g, where p is a large prime number and $g \in Z_p^*$. Then KGC randomly chooses $k, x \in Z_p^*$ and computes a and (k_1,k_2) as follows

$$a = g^x \mod p \tag{IV.1}$$
$$k_1 = g^k \mod p \tag{IV.2}$$
$$k_2 = k \cdot a^k \mod p \tag{IV.3}$$

2) Blinding($\mathbf{m}$): Given a random diagonal matrix $\mathbf{n}$ whose elements belong to $[1, p]$. Assuming that a plaintext vector $\mathbf{m}$, it is blinded to

$$\mathbf{x} = \mathbf{n} \cdot \mathbf{m}. \tag{IV.4}$$

Note that $\mathbf{n}$ and $\mathbf{m}$ have the same dimension.

3) Enc_Ser($\boldsymbol{x}$): Given a data matrix $\mathbf{x}$ and k_2, $\mathbf{x}$ is encrypted as

$$[\mathbf{x}] = k_2 \cdot \mathbf{x}. \tag{IV.5}$$

4) Enc_User($[\boldsymbol{x}]$): Given a partially encrypted data $[\mathbf{x}]$ and k_1, x, and the inverse matrix of $\mathbf{n}$ as $\mathbf{n^{-1}}$, it yields the final encryption $[[\mathbf{x}]]$ such that

$$[[\mathbf{x}]] = \mathbf{n^{-1}} \cdot [\mathbf{x}] \cdot k_1^{-x}. \tag{IV.6}$$

5) PartDec($[[\boldsymbol{x}]]$): Given the ciphertext $[[\mathbf{x}]]$ and k_2, it yields the corresponding decryption as follows

$$[\mathbf{m}] = [[\mathbf{m}]] \cdot k_2^{-1}. \tag{IV.7}$$

6) FullDec($[\boldsymbol{m}]$): Given a partially decrypted data $[\mathbf{m}]$ and k_1, x, the plaintext $\mathbf{m}$ is decrypted as

$$\mathbf{m} = k_1^x \cdot [\mathbf{m}]. \tag{IV.8}$$

B. Full Homomorphism Mechanism

In order to decrypt the ciphertext, it is necessary to run the PartDec and FullDec algorithms. JHE implements secure computation in ciphertext state based on full homomorphism [1]. Given two ciphertexts $[[x_1]]$, $[[x_2]]$, JHE satisfies the full homomorphism such that

$$[[\mathbf{x_1}]] \cdot [[\mathbf{x_2}]] = [[\mathbf{x_1} \cdot \mathbf{x_2}]] \tag{IV.9}$$
$$[[\mathbf{x_1}]] + [[\mathbf{x_2}]] = [[\mathbf{x_1} + \mathbf{x_2}]] \tag{IV.10}$$

As JHE operates on integers, floating numbers are converted to integers before encrypting it. The approximation Ftol is utilized to convert a floating number x to an integer ix by running $\mathsf{Ftol}(x)$ as follows:

$$\mathsf{Ftol}(x) = \lfloor x \cdot P \rceil, \tag{IV.11}$$

where P is the employed precision value.

V. SCFL FRAMEWORK

In this section, we describe the system framework, adversary model, and security goals for SCFL respectively. The notations are described in **Table 2**.

A. System Framework

As shown in **Fig. 1**, the system framework comprises a key generation center KGC, two servers $Server1$ and $Server2$ who are separately responsible for cryptosystem, clustering and aggregation, and n users U_i, where $i \in [1, n]$.

- **KGC.** KGC generates system parameters, computes and distributes system keys for $Server1$, $Server2$ and $Users$.
- **Servers.** $Server1$ is responsible for encrypting and decrypting gradients from $Users$ and $Server2$, while $Server2$ is responsible for the clustering and aggregation.
- **Users.** In the t-th training iteration, U_i trains local gradients which are encrypted and uploaded to $Server2$, and receives the gradients' ciphertext from $Server1$ for decryption, and finally downloads the global gradient for the next training iteration.

TABLE II
NOTATION DESCRIPTIONS

Notations	Descriptions	Notations	Descriptions
$epoch$	model training iteration	$\mathbf{n}_i$	random diagonal matrix
$[[x]]$	fully encrypted ciphertext	$[x]$	partially encrypted ciphertext
g_i	local gradient	g	global gradient
g_i^*	poisoned gradient	P	precision value
FtoI	approximation function	Blinding	blinding process in encryption
C	clusters	Cen	cluster centers
$Score$	each user's score in cluster	$MaxScore$	maximum score
Att	attack ratio	cos	cosine similarity function
con	weight for aggregation within cluster	siz	weight for aggregation between cluster
Acc	model accuracy	$AccImp$	improvement of Acc

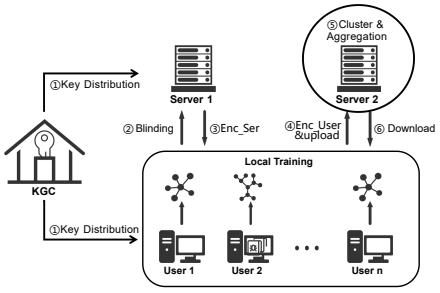


Fig. 1. System model.

B. Adversarial Model

In our scheme, we assume KGC is fully trusted, and $Server1$ and $Server2$ are two honest-but-curious and non-colluding entities that honestly follow the pre-defined protocols but curiously compromise users' privacy. The non-collusion of the twin-server model has been formalized in previous work [16].

In SCFL, the users are curious about the data of other users, and the users are categorized as either benign or malicious, with poisoners potentially accessing the local datasets. As shown in **Fig.1**, we assume "User 1" and "User n" are benign, while "User 2" is malicious.

Benign Users. A user U_i is considered benign if and only if U_i truthfully provides local gradients g_i which is trained from the user's local dataset D_i and serves as an unbiased estimator of the authentic gradient.

Malicious Users. A user U_j is deemed malicious if and only if U_j is controlled by a Byzantine adversary who implements model poisoning attacks by submitting encrypted malicious gradients g_j^*, and aims to weaken the global model's accuracy or integrity.

C. Design Goals

The objective of our work is to design a PPFL scheme, i.e. SCFL, which can protect the users' privacy and train a joint model with high degree of accuracy and robustness, as well as high computational and communication efficiency, even in face of adversarial disruptions during training.

Accuracy. SCFL guarantees high model accuracy for both IID and nonIID data scenarios.

Robustness. SCFL provides consistently accurate output to all users if less than 50% users are compromised, i.e. $|U^*|/|U| \leq 50\%$, in both IID and nonIID contexts.

Privacy. In SCFL, each user's local gradient and the global model are protected even if the adversaries can access the transmitted information. Here the adversaries include the curious users and servers.

Efficiency. SCFL optimizes the performance of the system and minimizes both computation and communication costs.

VI. SCFL SCHEME

A. Technical Overview

Fig. 2 shows the overview of our SCFL scheme, that uses clustering and cosine similarity techniques to defend against model poisoning attacks. In our scheme, outliers are identified by checking the cosine similarity to the center of the cluster to see whether the local gradient is in the cluster.

To defense model poisoning attacks, in SCFL a cosine similarity-based secure clustering defense strategy is designed to identify outliers and Byzantine tolerance aggregation mechanism is designed to cope with heterogeneous data settings including IID and nonIID data. In terms of privacy protection, SCFL relies on JHE to encrypt the user's local gradient so that no information about the user's model is leaked during the data transmission and server-side clustering and aggregation.

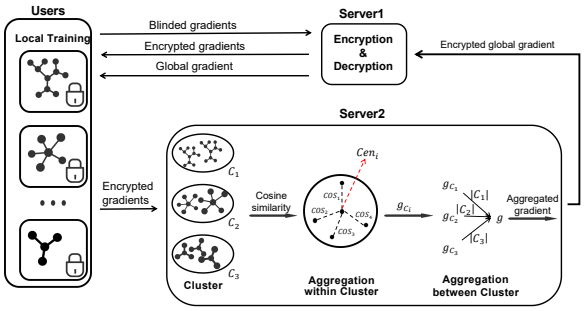


Fig. 2. Overview of SCFL

B. Construction of SCFL

Firstly, the general process of SCFL is shown in Algorithm 1, which consists of four processes, namely Setup, Local training, Cluster, and Aggregation.

Algorithm 1 General process of SCFL

Input: System parameters λ, number of clients n, model training iteration $epoch$
Output: Global gradient W

Setup: SCFL completes system initialization.
for t in $epoch$ **do**
  Each user U_i ($i \in [1, n]$) downloads the global gradient from $Server2$.
  Local training: U_i trains individual local gradient and encrypts it.
  Cluster: $Server2$ clusters users' encrypted gradients and detects the potentially poisoned gradients.
  Aggregation: $Server2$ successively implements aggregation within a cluster and between clusters to get the global gradient.
end for
return global model W.

1) Setup: In the setup process, KGC implements key generation and distribution for $Users$ and $Server1$. $Server2$ distributes the initial model to $Users$, and initializes each user's initial score $Score_i$ as the maximum score $MaxScore$, which is used in the Cluster phase. The details are shown in Algorithm 2.

Algorithm 2 Setup

Input: System parameters p, g
Output: System keys k_1, k_2

/*Key distribution*/
KGC generates x, k and implements $(a, k_1, k_2) \leftarrow$ KeyGen(λ);
return KGC broadcasts p, g and distributes x, k_1 to $Users$, k_2 to $Server1$ securely;
/*Model distribution*/
for $U_i \in U (i \in [1, n])$ **do**
  U_i downloads W_0 from $Server2$.
end for
/*Score Initialization*/
$Server2$ initializes each user U_i's $Score_i$ to $MaxScore$ which is the maximum score.

2) Local training: In the t-th training iteration, each user U_i trains local gradient $g_i^{(t)}$ using SGD, and sends its blinded result to $Server1$ for partial encryption, i.e. Enc_Ser. Then U_i receives the partially encrypted gradient $[g_i^{(t)}]$ from $Server1$, implements the user side encryption Enc_User, and finally uploads $[[g_i^{(t)}]]$ to $Server2$. The detailed procedures are shown in Algorithm 3, including benign training (for benign users U) or local poisoning (for malicious users U^*), gradient normalization and gradient encryption.

Algorithm 3 Local training

Input: Global gradient $g^{(t-1)}$ and each user's training data $D_i (i \in [1, n])$
Output: Encrypted local gradients $[[g_i^{(t)}]]$

if $U_i \notin U^*$ **then**
  U_i obtain $g_i^{(t)}$ trained on local data;
  U_i normalizes local gradients $g_i^{(t)}$ before sending them to $Server2$;
else
  U_i^* launches model poisoning, and yields poisonous gradients $g_i^{*(t)}$;
end if
/*Encryption of the users' gradients*/
for $i = 1$ to n **do**
  U_i blinds the gradient $g_i^{(t)}$ or $g_i^{*(t)}$ with a randomly chosen matrix $\mathbf{n}_i$ and uploads to $Server1$;
  $Server1$ uses k_2 to encrypt the blinded gradients and sends the ciphertext $[g_i^{(t)}]$ or $[g_i^{*(t)}]$ to the user U_i;
  U_i uses $\mathbf{n}_i^{-1}$ to remove the blindness and further encrypts it with k_1 and x;
  return the encrypted local gradients $[[g_i^{(t)}]]$ or $[[g_i^{*(t)}]]$.
end for

3) Cluster: In the t-th training iteration, after receiving the encrypted gradients uploaded by the users, $Server2$ clusters them, which includes three steps: constructing the KD-tree, clustering gradients, and detecting malicious gradients. We improve the KD-DBSCAN algorithm by constructing KD-tree based on cosine similarity instead of Euclidean distance, as this method is more suitable for high-dimensional data and more robust to distributional offsets and direction-sensitive malicious attacks (e.g., gradient poisoning). In this phase, users' gradients of different distributions $[[g_i^{(t)}]]$ are clustered into k different clusters $C = (C_1, C_2, ..., C_k)$ whose centers are noted as $Cen = (Cen_1, Cen_2, ..., Cen_k)$. If one's gradient does not belong to any cluster or deviates from the cluster which it belonged to in the previous round, it is adjudged as outlier and the $Score_i$ is reduced by one. When $Score_i = 0$, the user U_i is identified as a malicious user and disqualified for subsequent training. The details of Cluster are shown in Algorithm 4.

4) Aggregation: After clustering, $Server2$ performs the aggregation operation. In order to prevent interference of malicious gradients to the global model before the malicious users are removed, we propose a Byzantine-tolerant aggregation mechanism, where gradients are aggregated using the credibility con and the size of the cluster siz as the weights of the aggregation within and between the clusters respectively. In each cluster, $Server2$ calculates the cosine similarity $cosine_i$ between the gradient of each user U_i and its cluster center Cen_j as well as the credibility con_i, and uses con_i ($i \in [1, |C_j|]$) as the weight to aggregate the gradients in the cluster. After obtaining the aggregated gradients of each cluster, $Server2$ then aggregates them into global gradient

Algorithm 4 Cluster

Input: Encrypted local gradients $[[g_i^{(t)}]]$
Output: Cluster C and cluster center Cen

/*Construction of the KD-tree*/
Assuming that the dimension of the gradient is m;
$Server2$ randomly selects one of local gradients as the base-gradient and calculates the cosine similarity between the consecutive binary vectors of base-gradient and the received gradient $[[g_i^{(t)}]]$, forming n new $(m-1)$ dimensional cos-gradients;
/*Cluster*/
$Server2$ builds the KD-tree based on the cos-gradients and clusters accelerated by KD-tree;
return Cluster $C = (C_1, ..., C_k)$ and cluster centers $Cen = (Cen_1, ..., Cen_k)$;
/*Detection of malicious gradients */
for $i = 1$ to n **do**
    if $Score_i \leq 0$ **then**
        Move $[[g_i^{(t)}]]$ out of C and disable U_i participating in subsequent training;
    else
        if $[[g_i^{(t)}]] \notin C$ **then**
            $Score_i = Score_i - 1$;
        end if
    end if
end for

using the size of each cluster siz as the weight. Finally, $Server2$ returns the global gradient. The details of aggregation are shown in Algorithm 5.

Algorithm 5 Aggregation

Input: Cluster C and cluster centers Cen
Output: Global gradients $g^{(t)}$

for C_j in C **do**
    $cosine_i = \textbf{cos}([[g_i^{(t)}]], Cen_j)$
    $con_i = cosine_i / \sum_{k=1}^{|C_j|} cosine_k$
    $[[g_j^{(t)}]] = \sum_{i=1}^{|C_j|} con_i \cdot [[g_i^{(t)}]]$
end for
$siz_j = |C_j|/|C|$
$[[g^{(t)}]] = \sum_{j=1}^{|C|} siz_j \cdot [[g_j^{(t)}]]$
return global gradients $[[g^{(t)}]]$.

VII. Security Analysis and Performance Comparison

In this section, we analyze the security of our SCFL scheme, and show the experimental performance of SCFL from three perspectives including the communication and computation overhead, the performance across heterogeneous data settings, i.e. nonIID and IID data, and the performance against different model poisoning attacks including targeted attacks and untargeted attacks. To highlight the advantages of SCFL, we construct the poisonous PPFL without defense(FedAvg) as the baseline, and compare our approach with ShieldFL [15] and the baseline.

A. Security Analysis

In the following we will show that our SCFL scheme satisfies privacy-preserving of users' and global model.

For external adversaries, since the whole training process of the model is carried out in the JHE encrypted environment, all gradients are encrypted using JHE and JHE satisfies the IND-CPA security [10], the external adversaries cannot obtain any valid information of U_i's gradient or the global model from the ciphertexts.

For honest-but-curious $Server1$, as the user has blinded his gradient before it is uploaded, $Server1$ cannot access U_i's plaintext gradient information. When $Server2$ sends the encrypted global model to $Server1$, as it can only partially decrypt it with k_2 and the JHE is IND-CPA secure, $Server1$ cannot access the plaintext of the global model without the user's key k_1.

For honest-but-curious $Server2$, as the Cluster and Aggregation processes are done in encrypted environment, it can only access the gradients' ciphertexts. As JHE is IND-CPA secure, $Server2$ can not access the plaintext information of users' gradients and the global model without the keys k_1, k_2.

For curious users, since each user individually uses his random matrix to blind his gradient and the JHE satisfies the IND-CPA security, each user can only learn his own gradient and the global model, but can not access the other users' local gradients.

B. Experiment Setup

The evaluation of our experiments is performed using the PyTorch framework running on an Ubuntu 22.04.5 LTS server. It uses the JHE scheme with a security level of 80 bits ($\|p\|$= 1024 bits) implemented in Python 3.7. Furthermore, we have simulated multiple users with lightweight threads, where each thread implements a PyTorch regression classifier for PPFL in SCFL that is relevant to real-world applications.

1) Datasets, Data Settings and Model Architectures: We introduce the datasets, data settings, and model architectures used in our experiments in the following.

a) Datasets: We evaluate the performance of SCFL using two benchmark datasets. Each dataset is divided into a training set and a testing set, and the ratio of the training set to the testing set is 0.8 to 0.2.

MNIST: The dataset contains handwritten digits with 10 classes, and it provides 60, 000 training samples, each of which is a 28×28 pixel grayscale image.

CIFAR-10: CIFAR-10 contains real objects in the real world, which are not only noisy, but also have different proportions and features of the objects. Each sample is a 32×32 color RGB images.

b) Data settings: To evaluate SCFL, we build both IID setting and nonIID setting.

IID setting: The training data are uniformly partitioned among U and each user stores IID data samples.

NonIID setting: The training data are divided into different classes, and each user stores the training data samples that belong to a single class.

c) Model architectures: We use the widely used logistic regression classifier as a global model on different datasets with a mini-batch size of 100. The number of training iterations in SCFL is 290 for MNIST and CIFAR-10. Besides, we set the precision value $P = 10^5$ for the FtoI.

2) Model Poisoning Attacks in PPFL: To evaluate SCFL against model poisoning, we construct model poisoning attacks. In particular, our adversarial attack is set to the highest level. All the data samples stored by a malicious user U^* are poisoned and are used to train encrypted local poisonous gradients $[[g^*]]$. To evaluate the performance of SCFL, we simulate different levels of model poisoning attack, where local gradients are encrypted. To implement various attack ratios Att, the Byzantine adversary is designed to corrupt different ratio of malicious users $U^* \in U$, i.e. $Att = |U^*|/|U|$, where $|U|$ is the total number of users and $|U^*|$ is the number of malicious users. In the experiment, Att various from 0.1 to 0.5, which is the theoretic upper bound of Byzantine attacks.

C. Performance Comparison

In this section, we compare communication complexity, computation complexity, accuracy and accuracy improvement of our scheme compared with baseline and ShieldFL [15].

1) Communication Complexity: To evaluate the communication overhead of SCFL compared to ShieldFL, we discuss the communication cost in each training iteration. The comparison result is shown in **Table 3**, where n denotes the number of users, m denotes the number of dimensions in local gradients, and $|X|$ denotes the communication complexity of an encrypted vector. From **Table 3** we can find that the communication complexity of our scheme is twice as high as ShieldFL's in local training phase, and extra communication overhead is added in the Cluster phase of our scheme, but the communication complexity in Aggregation phase is only a quarter of ShieldFL's. As tremendous gradients of the users are aggregated at the server side, it is very essential to reduce the communication cost in Aggregation phase. Therefore, our SCFL scheme is more efficient in communication.

TABLE III
COMMUNICATION COMPLEXITY IN SCFL AND SHIELDFL

Phase	SCFL	ShieldFL
Local Training	$2nm\|X\|$	$nm\|X\|$
Cluster	$2nm\|X\|$	–
Aggregation	$3nm\|X\|$	$12nm\|X\|$
Total	$7nm\|X\|$	$13nm\|X\|$

2) Computation Complexity: To compare the computation overhead of the cryptographic algorithms in SCFL and ShieldFL, we test the run time of a single 1000-dimensional gradient encrypted by JHE in SCFL and Two-trapdoor HE in ShieldFL. As shown in **Table 4**, for both encryption and decryption our SCFL is more efficient compared with ShieldFL, and SCFL reduces total run time by about 80%.

TABLE IV
A COMPARISON OF RUN TIME FOR ENCRYPTION AND DECRYPTION

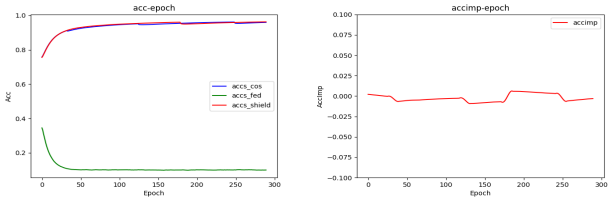
Phase	SCFL	ShieldFL
Encrypt	$6.6ms$	$10ms$
Decrypt	$3.4ms$	$40ms$
Total	$10ms$	$50ms$

3) Model Accuracy: In our experiment, we evaluate the accuracy of our scheme by comparing it with baseline(FedAvg) and ShieldFL in two aspects: Acc, i.e. the model accuracy, and $AccImp$, which means the accuracy improvement in SCFL compared to ShieldFL. We only show the accuracy of SCFL, FedAvg and ShieldFL when poison attacks are present in IID and nonIID data settings. And we compare the accuracy of these three schemes in IID data setting when $Att = 0.5$, and in nonIID data setting when $Att = 0.1$ and $Att = 0.5$.

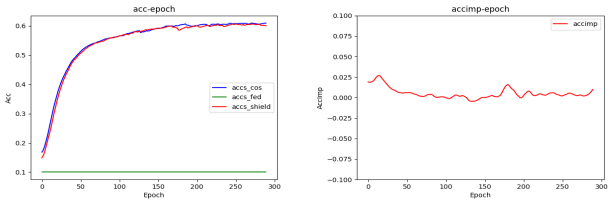
a) Accuracy in IID setting with Poison Attack: When we set $Att = 0.5$ in IID data setting, as shown in **Fig. 3** we find that in both MNIST and CIFAR-10 datasets Acc of baseline is less than 10%. In CIFAR-10 dataset, Acc of SCFL is improved compared to ShieldFL, preferably by 2.5%.

b) Accuracy in nonIID settings with Poison Attack: We test the model accuracy of SCFL, ShieldFL and the baseline in nonIID data setting with different attack ratios (i.e. Att = 0.1 and 0.5). The results are shown in **Fig. 4**. When $Att = 0.1$, the accuracy of baseline is about 20%, and Acc of SCFL is about 90%, which has 1% to 2% improvement compared with ShieldFL's accuracy. When Att = 0.5, the accuracy of baseline is less than 10%, while the accuracy of SCFL maintains more than 80% and compared to ShieldFL's, Acc of SCFL is improved by 1% to 5%.

c) Comparative Analysis: From **Fig. 3** and **Fig. 4**, it can be found that when there is poisoning attack, the accuracy of FedAvg, which has no defense, is around 10%, while SCFL provides a steady and much higher accuracy and consistently

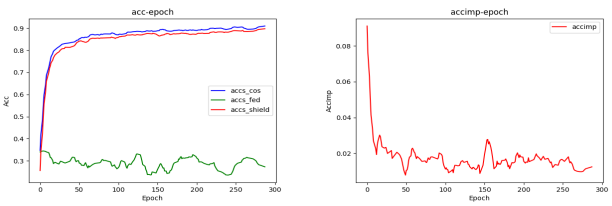


(a) MNIST and $Att = 0.5$

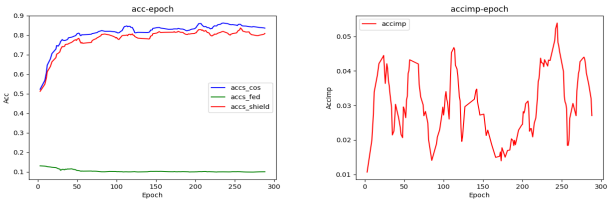


(b) CIFAR-10 and $Att = 0.5$

Fig. 3. Accuracy Comparison in IID settings



(a) MNIST and $Att = 0.1$



(b) MNIST and $Att = 0.5$

Fig. 4. Accuracy Comparison in nonIID settings

has better accuracy than ShieldFL. As shown in **Fig. 4**, ShieldFL can reach 85% accuracy when Att=0.1, but the accuracy drops to less than 80% when $Att = 0.5$, and our SCFL achieves 90% accuracy when $Att = 0.1$ and about 85% accuracy even when $Att = 0.5$. Therefore, our SCFL realizes better robustness even when half of the users are compromised.

VIII. Conclusion

In this paper, we propose a privacy-preserving defense strategy SCFL against model poisoning attacks in PPFL. In SCFL, the proposed defense strategy is based on secure KD-DBSCAN clustering and the cosine similarity of encrypted gradients. It provides a feasible solution for detecting encrypted malicious gradients and thus can resist encrypted model poisoning attacks in PPFL. SCFL utilizes joint homomorphic encryption to prevent the leakage of model privacy in PPFL. In SCFL, a Byzantine tolerant aggregation mechanism is realized to maintain high model accuracy in both IID and nonIID data settings. The experiment results show that SCFL achieves better performance and model accuracy compared with state-of-the-art PPFL scheme.

References

[1] Asharov, G., Jain, A., López-Alt, A., Tromer, E., Vaikuntanathan, V., Wichs, D.: Multiparty computation with low communication, computation and interaction via threshold fhe. In: Advances in Cryptology–EUROCRYPT 2012: 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings 31. pp. 483–501. Springer (2012)

[2] Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., Shmatikov, V.: How to backdoor federated learning. In: International conference on artificial intelligence and statistics. pp. 2938–2948. PMLR (2020)

[3] Bhagoji, A.N., Chakraborty, S., Mittal, P., Calo, S.: Analyzing federated learning through an adversarial lens. In: International conference on machine learning. pp. 634–643. PMLR (2019)

[4] Bi, M., Wang, Y., Cai, Z., Tong, X.: A privacy-preserving mechanism based on local differential privacy in edge computing. China Communications **17**(9), 50–65 (2020)

[5] Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H.B., Patel, S., Ramage, D., Segal, A., Seth, K.: Practical secure aggregation for privacy-preserving machine learning. In: proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 1175–1191 (2017)

[6] Chen, Y., Zhou, L., Pei, S., Yu, Z., Chen, Y., Liu, X., Du, J., Xiong, N.: Knn-block dbscan: Fast clustering for large-scale data. IEEE transactions on systems, man, and cybernetics: systems **51**(6), 3939–3953 (2019)

[7] Chen, Y., Ruys, W., Biros, G.: Knn-dbscan: a dbscan in high dimensions. ACM Transactions on Parallel Computing **12**(1), 1–27 (2025)

[8] Fang, M., Cao, X., Jia, J., Gong, N.: Local model poisoning attacks to {Byzantine-Robust} federated learning. In: 29th USENIX security symposium (USENIX Security 20). pp. 1605–1622 (2020)

[9] Fung, C., Yoon, C.J., Beschastnikh, I.: The limitations of federated learning in sybil settings. In: 23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020). pp. 301–316 (2020)

[10] Gao, J., Yu, H., Zhu, X., Li, X.: Blockchain-based digital rights management scheme via multiauthority ciphertext-policy attribute-based encryption and proxy re-encryption. IEEE Systems Journal **15**(4), 5233–5244 (2021)

[11] Guerraoui, R., Rouault, S., et al.: The hidden vulnerability of distributed learning in byzantium. In: International conference on machine learning. pp. 3521–3530. PMLR (2018)

[12] Hou, B., Gao, J., Guo, X., Baker, T., Zhang, Y., Wen, Y., Liu, Z.: Mitigating the backdoor attack by federated filters for industrial iot applications. IEEE Transactions on Industrial Informatics **18**(5), 3562–3571 (2022)

[13] Jia, C., Li, R., Wang, Y., et al.: Privacy protection scheme of dbscan clustering based on homomorphic encryption. Journal on Communications **42**(2), 1–11 (2021)

[14] Liu, Y., Fan, T., Chen, T., Xu, Q., Yang, Q.: Fate: An industrial grade platform for collaborative learning with data protection. Journal of Machine Learning Research **22**(226), 1–6 (2021)

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

[16] Mohassel, P., Zhang, Y.: Secureml: A system for scalable privacy-preserving machine learning. In: 2017 IEEE symposium on security and privacy (SP). pp. 19–38. IEEE (2017)

[17] Shen, S., Tople, S., Saxena, P.: Auror: Defending against poisoning attacks in collaborative deep learning systems. In: Proceedings of the 32nd annual conference on computer security applications. pp. 508–519 (2016)

[18] Shibla, T., Shibu Kumar, K.B.: Improving efficiency of dbscan by parallelizing kd-tree using spark. In: 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS). pp. 1197–1203 (2018)

[19] Soria-Comas, J., Domingo-Ferrer, J., Sánchez, D., Megías, D.: Individual differential privacy: A utility-preserving formulation of differential privacy guarantees. IEEE Transactions on Information Forensics and Security **12**(6), 1418–1429 (2017)

[20] Truex, S., Baracaldo, N., Anwar, A., Steinke, T., Ludwig, H., Zhang, R., Zhou, Y.: A hybrid approach to privacy-preserving federated learning. In: Proceedings of the 12th ACM workshop on artificial intelligence and security. pp. 1–11 (2019)

[21] Wang, H., Kaplan, Z., Niu, D., Li, B.: Optimizing federated learning on non-iid data with reinforcement learning. In: IEEE INFOCOM 2020-IEEE Conference on computer communications. pp. 1698–1707. IEEE (2020)

[22] Yin, D., Chen, Y., Kannan, R., Bartlett, P.: Byzantine-robust distributed learning: Towards optimal statistical rates. In: International conference on machine learning. pp. 5650–5659. PMLR (2018)

[23] Yin, X., Zhu, Y., Hu, J.: A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions. ACM Computing Surveys (CSUR) **54**(6), 1–36 (2021)