Privacy-Enhanced Federated Learning Against Poisoning Adversaries

Xiaoyuan Liu, *Graduate Student Member, IEEE*, Hongwei Li, *Senior Member, IEEE*, Guowen Xu, *Member, IEEE*, Zongqi Chen, *Graduate Student Member, IEEE*, Xiaoming Huang, *Member, IEEE*, and Rongxing Lu, *Fellow, IEEE*

***Abstract*— Federated learning (FL), as a distributed machine learning setting, has received considerable attention in recent years. To alleviate privacy concerns, FL essentially promises that multiple parties jointly train the model by exchanging gradients rather than raw data. However, intrinsic privacy issue still exists in FL, e.g., user's training samples could be revealed by solely inferring gradients. Moreover, the emerging poisoning attack also poses a crucial security threat to FL. In particular, due to the distributed nature of FL, malicious users may submit crafted gradients during the training process to undermine the integrity and availability of the model. Furthermore, there exists a contradiction in simultaneously addressing two issues, that is, privacy-preserving FL solutions are dedicated to ensuring gradients indistinguishability, whereas the defenses against poisoning attacks tend to remove outliers based on their similarity. To solve such a dilemma, in this paper, we aim to build a bridge between the two issues. Specifically, we present a privacy-enhanced FL (PEFL) framework that adopts homomorphic encryption as the underlying technology and provides the server with a channel to punish poisoners via the effective gradient data extraction of the logarithmic function. To the best of our knowledge, the PEFL is the first effort to efficiently detect the poisoning behaviors in FL under ciphertext. Detailed theoretical analyses illustrate the security and convergence properties of the scheme. Moreover, the experiments conducted on real-world datasets show that the PEFL can effectively defend against label-flipping and backdoor attacks, two representative poisoning attacks in FL.**

***Index Terms*— Federated learning, poisoning attack, privacy protection, cloud computing.**

Manuscript received April 8, 2021; revised June 26, 2021; accepted August 1, 2021. Date of publication August 30, 2021; date of current version September 16, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 62020106013, Grant 61972454, Grant 61802051, Grant 61772121, and Grant 61728102; in part by Sichuan Science and Technology Program under Grant 2020JDTD0007 and Grant 2020YFG0298; in part by Chongqing Science and Technology Commission under Grant cstc2018jcyjAX0703; and in part by the Fundamental Research Funds for Chinese Central Universities under Grant ZYGX2020ZB027. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Dali Kaafar. *(Corresponding author: Hongwei Li.)*

Xiaoyuan Liu and Zongqi Chen are with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: xiaoyuan.l@foxmail.com; czq5659502@foxmail.com).

Hongwei Li is with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China, and also with the Chongqing Key Laboratory of Big Data and Intelligent Computing, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing 400714, China (e-mail: hongweili@uestc.edu.cn).

Guowen Xu is with the School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798 (e-mail: guowen.xu@ntu.edu.sg).

Xiaoming Huang is with CETC Cyberspace Security Research Institute Company Ltd., Chengdu 610041, China (e-mail: apride@gmail.com).

Rongxing Lu is with the Faculty of Computer Science (FCS), University of New Brunswick (UNB), Fredericton, NB E3B 5A3, Canada (e-mail: rlu1@unb.ca).

Digital Object Identifier 10.1109/TIFS.2021.3108434

I. Introduction

FEDERATED learning (FL) has been emerging as a promising machine learning setting [1], in which multiple parties train a copy of the model locally and then submit the model updates (i.e. gradients) to the parameter server for aggregation, so as to construct an improved version. Benefitting from its ability to train the model without storing the training data in the cloud, FL has become one of the fundamental techniques in many security-sensitive tasks such as autonomous driving [2] and medical image analyses [3].

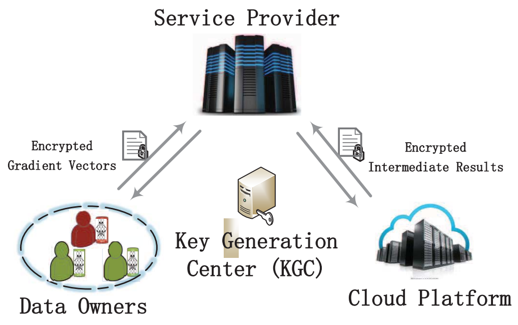
Despite its potential, security and privacy concerns in FL have also aroused widespread attention. One of the prominent concerns is privacy leakage from gradients. The semi-honest server can still recover some sensitive information (e.g. avatar, mood, and medical information) of the target users through the received gradients, as indicated in previous works [4], [5]. Another crucial security threat faced by FL is poisoning attacks. Because the server does not have access to the users' dataset and federated training process, malicious users may submit customized gradients so as to induce classification errors at the test phase. Notably, a poisoned update can take control of the entire training process, thereby rendering the final model invalid [6]–[8]. Such manipulation may also indirectly infringe upon users' data privacy. For example, by uploading inverted and amplified gradients, the adversary can infer whether the samples were used to train the target model [7]. Motivated by the aforementioned issues, a trustworthy FL must consider the following fundamental concerns: (1) how to ensure that users' data privacy is not leaked and (2) how to guarantee the robustness of the model against adversarial manipulations.

To protect users' privacy, existing works mainly focused on ensuring the confidentiality of gradients. The solutions are generally based on the following three underlying technologies: Differential Privacy (DP) [9]–[12], Secure Multi-Party Computation (SMC) [13]–[15] and Homomorphic Encryption (HE) [16]–[18]. DP is a mathematically-formalized standard for the privacy-preserving analysis and use of data with some advantages in terms of efficiency. DP-based FL, however, is typically accomplished by incorporating random noise into gradients, which inevitably drops the accuracy of model [19]. SMC realizes private and correct computation through multiple interactions between participants [13], [14]. However, this mode not only incurs huge communication cost but also requires participants to stay online, which induce new performance bottlenecks for the actual distributed scenarios. Compared with DP and SMC, HE allows to correctly aggregate gradients while supporting users dropouts during the training. Prior works [16]–[18] have also shown the superior performance of HE to achieve secure FL.

Despite extensive investigations into the topic of privacy-preserving FL, existing efforts have not concurrently considered a fundamental question, i.e. do all users submit benign gradients honestly and reliably? In reality, it is possible for a purpose-driven terrorist to misguide other autonomous vehicles at the stop sign to move forward by submitting maliciously crafted parameters, posing a serious threat to traffic safety. As a result, for a secure and robust FL, as mentioned above, it is crucial to resist poisoning attacks launched by malicious adversaries while protecting users' privacy. However, existing schemes for solving privacy issues and poisoning attacks in FL focus around two opposite directions: the privacy-preserving FL solutions seek to ensure data indistinguishability, whereas the defenses against poisoning attacks tend to remove malicious gradients based on their similarity to benign gradients. Currently, the similarity is mainly measured based on one or more of the following criteria: 1) The difference between gradients in Euclidean space, e.g., *Krum* [6]. 2) Variations in gradient distributions such as *Detox* [20]. 3) Difference in L_p-norms between gradients like *GeoMed* [21]. From this perspective, the above defenses for extracting valuable statistical information rely heavily on data distinguishability. The contradiction often makes it a major challenge to identify the poisoning behaviors in FL without compromising users' privacy. The closest to our method we noticed is [22], as it focuses on making the current distance-based outlier removal mechanism compatible with secure aggregation utilizing SMC. However, the scheme inherits the performance and costly computation overhead of the aggregation rule, as well as additional communication overhead required by SMC.

To address the aforementioned challenge, in this paper, we propose an effective and privacy-enhanced FL framework (PEFL), which can efficiently detect poisoning behaviors during the federated training process while protecting privacy. Specifically, the contributions of our work are threefold, as follows:

- We propose a new privacy-enhanced FL framework (PEFL) with HE as the underlying technology. The PEFL can ensure that malicious users cannot infer memberships by uploading malicious gradients, in addition to preventing the semi-honest servers from infringing users' privacy.
- We also propose a novel adaptive federated aggregation to mitigate poisoning attacks in FL, which assesses users' reliability by taking coordinate-wise medians as the benchmark and then adaptively adjusts the weights of corresponding users' gradients. To distinguish between malicious and benign gradients far from the benchmark, we further propose a logarithmic function to remove malicious gradients.
- We provide comprehensive security analysis and prove the convergence of the scheme. Besides, the experiments conducted on real-world datasets show that the PEFL can effectively resist label-flipping and backdoor attacks, both of which are typical poisoning attacks in FL.


Service Provider
Encrypted Gradient Vectors
Encrypted Intermediate Results
Key Generation Center (KGC)
Data Owners
Cloud Platform
Fig. 1. System model.

The rest of this paper is organized as follows. In Section II and III, we overview the problem statement and preliminaries. Then, we present the privacy-enhanced federated learning (PEFL) in detail in Section IV and carry out the theoretical analysis in Section V. Next, we evaluate the performance of the PEFL and discuss the related works in Section VI and VII, respectively. Finally, Section VIII summarizes this paper.

II. Problem Statement

A. System Model

As depicted in Fig.1, there are four basic entities in our system:

- *Key Generation Center (KGC)*: KGC is the independent and trusted agency that distributes and manages all the public and private keys (pk, sk).
- *Data Owners*: All data owners, also called users, collaboratively train a uniform model with the coordination of the service provider. For privacy reasons, each user trains the model locally over the private data on device, then uploads the encrypted gradients to the service provider. Besides, we assume that the data held by all data owners are independent and identically distributed, which is the same as many prior works [6], [17], [23].
- *Service Provider (SP)*: SP is responsible to receive all gradients submitted by users and aggregate them (usually by averaging) so as to obtain an optimized global model. Simultaneously, it is required to detect the poisoning attacks launched by potentially malicious users with the help of the cloud platform.
- *Cloud Platform (CP)*: CP provides service on the pay-per-use basis. It works with SP to perform the calculations in this paper. Besides, CP holds a private-public key pair

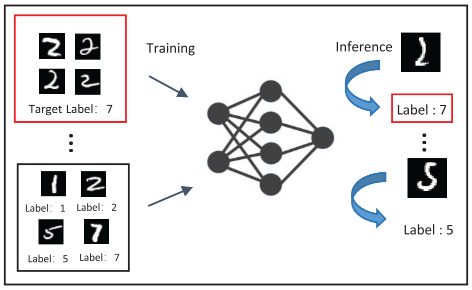


Fig. 2. Label-flipping attack.

(pk_c, sk_c) generated by a trusted authority (i.e., KGC in this paper), which can be used to encrypt data or decrypt the ciphertext.

B. Threat Model

In this paper, we focus on the privacy and security vulnerabilities of FL that could be abused by malicious participants. We assume that each poisoner has his/her own local dataset drawn from the same distribution as the training sets of other honest users. Malicious users can manipulate the private data stored on their devices for data poisoning. Similar to the prior works [6], [24], [25], to ensure the practicality and availability of the model, we make an assumption about the upper limit of the number of malicious users, i.e., $|F| \leq \frac{|M|-1}{2}$, where F and M are the number of malicious users and the total users, respectively, and $|\cdot|$ represents the number of users in the set. Besides, since the SP and CP have access to all users' local gradients, we consider them to be semi-honest adversaries. That means that both SP and CP are honest in performing all operations in compliance with the protocol, while striving to obtain more information based on the gradients they have mastered, thus compromising users' data privacy. Also, we assume that there is no collusion among the four entities (KGC, users, SP and CP) following previous works [22], [26].

C. Design Goals

With the above threats under consideration, the PEFL has the following three design goals:

- **Accuracy.** In this paper, the parameters submitted by poisoning adversary could cause the model to misclassify, degrading the accuracy of the trained model. Hence, a secure and robust FL needs to ensure that the accuracy of the model is within a reasonable range.
- **Robustness.** Robustness requires that the correct output is delivered to all protocol participants, no matter how the adversary misbehaves.
- **Privacy.** As prior works [4], [5] have shown, an adversary may recover users' sensitive information such as training samples or memberships by inferring the shared gradients. To protect users' data privacy, it is essential to keep each user's local gradients confidential.

III. Preliminaries

A. Federated Learning

Unlike traditional deep learning that centralizes all training data, FL is a promising distributed setting that allows all data owners to keep data local [27]. In FL, the server orchestrates the whole lifecycle of training until the model accuracy reaches the desired level, or the number of iterations reaches the preset value. The goal of learning is to find optimal model parameters $\boldsymbol{\omega}^\star$ so that the output of the model $\bar{y}$ is infinitely close to the true label for given feature vector $\mathbf{x}$.

In this paper, we focus on the supervised setting, i.e., the user U_x holds a private dataset D_x, where $D_x = \{\langle \mathbf{x}_i, y_i\rangle; i = 1, 2, \ldots, s\}$, and $\mathbf{x}_i \in \mathcal{R}^v$ represents v-dimensional feature vector, y_i is the corresponding class label. At the beginning of each iteration, the server selects a subset of m users that meet eligibility requirements, such as unoccupied or connected. Each selected user downloads current model parameters, and then locally trains the model over their private data. To achieve that, we exploit the stochastic gradient descent (SGD) algorithm, which is a popular method for iteratively optimizing the learning model. Instead of taking the whole dataset, the user randomly selects a small batch B of training data at each iteration to calculate a loss as:

$$L(B, \boldsymbol{\omega}) \leftarrow \frac{1}{|B|} \sum_{\langle \mathbf{x}_i, y_i\rangle \in B} L_f(\boldsymbol{\omega}, \mathbf{x}_i, y_i) \tag{1}$$

where $L_f(\boldsymbol{\omega}, \mathbf{x}_i, y_i)$ is a loss function that computes the gap between the current output of the model and the real label. Then, the gradient $\boldsymbol{G} \leftarrow \nabla_{\boldsymbol{\omega}} L(B, \boldsymbol{\omega})$ is calculated and shared to the server. At last, the server aggregates all local gradient vectors and updates the global model as follows:

$$\boldsymbol{\omega}^{t} \leftarrow \boldsymbol{\omega}^{t-1} - \eta \frac{\sum_{x \in [1,m]} \boldsymbol{G_x}}{m} \tag{2}$$

where $\boldsymbol{\omega}^t$ represents the model parameters after the t-th iteration and η is the learning rate, $\{\boldsymbol{G_x}, x \in [1, m]\}$ (also described as $\{\boldsymbol{G_x}\}_{x=1}^{x=m}$ in the following) refers to $\{\boldsymbol{G_1}, \boldsymbol{G_2}, \ldots, \boldsymbol{G_m}\}$, the set of m users' gradients.

Note that in this paper, we use bold symbols to represent vectors, unless there is a special explanation.

B. Poisoning Attack

In the poisoning attacks, the adversary intends to change the classification boundary of the trained model so that the specified feature space is mapped to the target class. It has been demonstrated that a single poisoner can control the entire training process and compromise users' data privacy [6], [7].

In this paper, we mainly focus on two types of representative poisoning attacks: One is label-flipping attack [28]. As shown in Fig.2, the label of the normal features is flipped to the target class. For example, the adversary incorrectly marks the samples with the true label 2 to label 7 for misclassification. The other is the backdoor attack [29] as illustrated in Fig.3, which aims to seek a set of parameters to establish strong links between the trigger and the target label while minimizing the impact on the classification of benign inputs. For instance, the adversary expects a model so that any pictures with a hat as a trigger will be classified as Captain Jack, while the pictures without a hat will be correctly classified. To achieve this, Bagdasaryan *et al.* [29] proposed an optimized model that minimizes the distance from the original model parameters

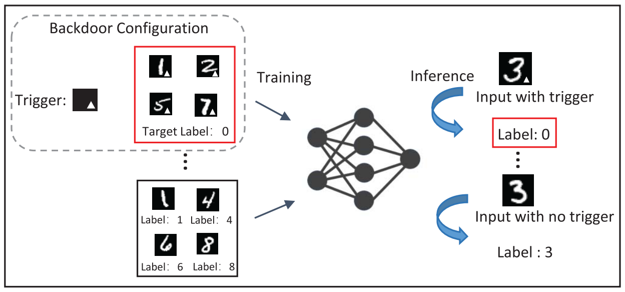


Fig. 3. Backdoor attack.

by adding an item to the loss function of the local model: $\alpha l_{backdoor} + (1-\alpha)l_{\Delta}$, where $l_{backdoor}$ is used to introduce the backdoor, and l_{Δ} is the Mean Square Error between the original parameters and new ones.

C. Linearly Homomorphic Encryption

Homomorphic encryption allows computation to be performed directly on encrypted data without requiring access to a secret key. In this paper, we implement privacy-enhanced FL by exploiting the linearly homomorphic encryption (LHE) [30], which is a public-key encryption scheme supporting linearly homomorphic operations over the ciphertexts.

In general, the LHE consists of a tuple of algorithms (KeyGen, Enc, Dec, Eval) as below:

- HE.KeyGen(1^{k}) $\rightarrow$ (pk, sk). HE.KeyGen denotes as a probabilistic polynomial time (PPT) algorithm, whose input is security parameter 1^{k}, output is public key pk and private key sk.
- HE.Enc(pk, x) $\rightarrow$ c. Upon receiving the public key pk and a plaintext x, the encryption function HE.Enc outputs the ciphertext c corresponding to x. For sake of brevity, $[\![x]\!]_{pk}$ is used to stand for the ciphertext form of encrypting x with the public key pk in the following paper.
- HE.Dec(sk, c) $\rightarrow$ x. HE.Dec is a decryption function, which takes as input the private key sk and a ciphertext c and returns the plaintext x corresponding to c.
- HE.Eval(pk, c_1, c_2, f_L) $\rightarrow$ c'. Given the public key pk and ciphertexts c_1, c_2, as well as a linear function f_L, the evaluate function HE.Eval outputs the ciphertext c' such that HE.Dec$(sk, c') = f_L(x_1, x_2)$, where $c_i =$ Enc(pk, x_i) for $i \in \{1, 2\}$.

The LHE satisfies the semantic security and function privacy. The semantic security means that given two messages x_1 and x_2 sampled from $\{0,1\}^{l}$ and a ciphertext c, the polynomial-time adversary $\mathcal{A}$ distinguishes the ciphertext c from either x_1 or x_2 with the probability of at most negligibly better than $\frac{1}{2}$. The function privacy requires that the homomorphic operations reveal no unnecessary information on their functionality. We take the Paillier cryptosystem [30] for example, which is a representative additive homomorphic encryption with the following two basic homomorphic properties, i.e., given two plaintexts x_1 and x_2, a constant r, and a public key pk, we have: (a) $[\![x_1]\!]_{pk} \cdot [\![x_2]\!]_{pk} = [\![x_1 + x_2]\!]_{pk}$; (b) $[\![x_1]\!]_{pk}^{r} = [\![r \cdot x_1]\!]_{pk}$. For more details about the LHE, please refer to the paper [30]. We emphasize that optimizing LHE is an orthogonal direction that is out of the scope of this paper, and our proposed scheme can obtain direct gains from any performance improvement of LHE.

TABLE I
SYMBOLS

Symbol	Description
κ	Security parameter
pk	Public key
sk	Private key
$\mathbf{x}$	Vector of data feature
y	Data label
$\boldsymbol{\omega}$	Weight of the model
η	Learning rate
$\boldsymbol{G}$	Gradient vector
n	Dimensionality of gradient vector
$[\![x]\!]_{pk}$	Ciphertext encrypted with public key pk
ρ	Pearson correlation coefficient
μ	Re-scaled Pearson correlation coefficient

IV. Proposed Scheme

In this section, we first discuss an overview of our proposed scheme, then present our scheme on how to mitigate the poisoning attacks in detail. For ease of reference, the symbols that appeared in this paper and corresponding descriptions are listed in TABLE I.

A. Overview

In the very heart of federated learning lies the assumption that the local training data held by users are independent and identically distributed (i.i.d.), which allows the local gradients to be the unbiased estimate of global update. The assumption is also the basis for the PEFL which attempts to identify and block the gradient vectors that deviate from the honest majority's gradients.

Our key insight is that the corrupted users submit malicious gradients for seeking a set of parameters $\boldsymbol{\omega}^{\tau}$ with a malicious goal, which differs from the target model $\boldsymbol{\omega}^{\star}$ of honest users, i.e., $\boldsymbol{\omega}^{\star} \neq \boldsymbol{\omega}^{\tau}$. This means that there is a perceptible difference between the malicious gradient vectors and benign ones. Actually, the low similarity with the benign gradients means that the gradient is malicious with high probability, which is the consensus for many related works [6], [24], [25] and us to identify and block the malicious gradients. In this paper, we measure the similarity through computing the Pearson correlation coefficient [31] between gradients, which is one of the most commonly-used similarity metrics between high-dimensional variables. The formula is as follows:

$$\rho_{x,y} = \frac{Cov(X, Y)}{\sigma(X)\sigma(Y)} \tag{3}$$

We can observe from the above formula that the Pearson correlation coefficient equals the covariance $Cov(\boldsymbol{X}, \boldsymbol{Y})$ divided by the product of the standard deviation of the two variables $\sigma(\boldsymbol{X})\sigma(\boldsymbol{Y})$.

To improve efficiency, we establish a benchmark (i.e., the coordinate-wise medians in this paper) to distinguish abnormal behaviors from the honest majority. Then, we design a logarithmic function to extract effective information and give weight to each gradient adaptively. In particular, the gradients with lower correlation are considered to be abnormal, while their weights should be set to zero for the gain in accuracy and robustness of the model. To protect the users' data privacy, we further propose four secure protocols for aggregating gradient vectors and optimizing the shared model in the ciphertext domain, which are implemented by adopting the additive homomorphic properties of the Paillier cryptosystem [30].

B. Construction of PEFL

1) System Setup: We require a trusted Key Generation Center (KGC) to generate a pair of asymmetric key (pk_c, sk_c) of the LHE for the Cloud Platform (CP), where the private key sk_c is kept only by the CP. All authorized users, meanwhile, hold the same pair of asymmetric key (pk_x, sk_x) of LHE generated by the KGC. Besides, at the beginning of the protocol, the Service Provider (SP) randomly initializes the parameters of global model $\boldsymbol{\omega_{init}}$.

2) Securely Training: The process of secure training consists of two phases: the local training phase and the robust aggregation phase. Detailed steps are described as follows:

a) Local training phase: In this phase, we assume that the upper limit of the proportion of poisoners in the subset is the same as that in the full, i.e., less than 50%. At round t, for all $x \in [1, m]$, user U_x^t receives the encrypted model $[\![\boldsymbol{\omega^t}]\!]_{pk_x}$ with the public key pk_x. After decrypting them, U_x^t trains and obtains the local gradient vector $\boldsymbol{G_x^t}$.

Instead of uploading the current gradients alone, we use the SGD with the momentum [32] to smooth out the update, which adds a series of previous gradients using an exponential decay factor γ $(0 < \gamma < 1)$. Therefore, the parameters submitted by users become: $\boldsymbol{G_x} \triangleq \sum_{\ell\in[0,t]} \gamma^{t-\ell}\boldsymbol{G_x^\ell}$. El-Mhamdi *et al.* [33] have shown that the momentum can not only accelerate convergence but also reduce the variance-norm ratio, which is of benefit to the robustness of the model. The PEFL inherits this advantage in mitigating the poisoning attacks in FL.

To protect data privacy, U_x^t encrypts the gradient vector $[\![\boldsymbol{G_x}]\!]_{pk_c}$ with the public key pk_c of the CP. Due to the essential requirements of the cryptographic primitives, for $i \in [1, n]$, each entry G_{xi} of the gradient vector should be encoded into integer form as follows:

$$\boldsymbol{G_x} \leftarrow \{\lceil 2^{prec} \cdot G_{xi} \rfloor\}_{i=1}^{i=n} \tag{4}$$

where $\lceil a \rfloor \in \mathbb{Z}$ represents the nearest integer to the real number a, and $prec$ is the bits of precision. Besides, G_{xi} is the i-th entry in the gradient vector $\boldsymbol{G_x}$.

To reduce the communication cost, we exploit the technique of ciphertext packing which packs multiple plaintexts into one ciphertext, and use the Single Instruction Multiple Data (SIMD) technique to perform operations on these values in parallel [34]. Assume the public key $pk_c = p$ (a large positive integer), for the plaintext space $\log_2 p \geq 2048$ bits, each user packs d plaintexts into a single ciphertext as follows:

$$\{[\![\overbrace{\underbrace{\lceil 2^{prec} \cdot G_{xi} \rfloor \mathbf{0}}_{prec+pad\ bits} \cdots \lceil 2^{prec} \cdot G_{x(i+d)} \rfloor \mathbf{0}}^{\lfloor \log_2 p \rfloor\ bits}]\!]_{pk_c}\}_{i=1}^{i=\lceil \frac{p}{d} \rceil} \tag{5}$$

where $\mathbf{0}$ denotes the zero-padding of pad bits to prevent from overflowing in ciphertexts additions. Besides, $d = \left\lfloor \frac{\log_2 p}{prec+pad} \right\rfloor$, which takes the integer portion of the real number, and $\lceil \frac{p}{d} \rceil = \lfloor \frac{p}{d} \rfloor + 1$. For simplicity, we use G_{xi} to represent $\lceil 2^{prec} \cdot G_{xi} \rfloor \mathbf{0}$ in the following. U_x^t eventually sends the packed ciphertext $[\![\boldsymbol{G_x}]\!]_{pk_c} = \{[\![\ G_{xi} \cdots G_{x(i+d)}]\!]_{pk_c}\}_{i=1}^{i=\lceil \frac{p}{d} \rceil}$ to the SP.

b) Robust aggregation phase: In this phase, the SP interacts with the CP to identify and block the poisoning attacks launched by malicious users. It is worth noting that we are the first to achieve robust aggregation under ciphertext.

First of all, upon receiving the encrypted gradient vectors $\{[\![\boldsymbol{G_x}]\!]_{pk_c}\}_{x=1}^{x=m}$ from m selected users, the SP works with the CP for calculating the coordinate-wise medians $\{[\![G_{(med)i}]\!]_{pk_c}\}_{i=1}^{i=n}$ without revealing $\{\boldsymbol{G_x}\}_{x=1}^{x=m}$. To achieve this, we design the secure protocol *SecMed*. The specific steps of *SecMed* are as follows: for $x \in [1, m]$, the SP firstly obscures $\{[\![G_{xi}]\!]_{pk_c}\}_{i=1}^{i=n}$ by multiplying the ciphertexts of randomly sampled value r_i, and sends the obscured values R'_{xi} to the CP. The CP then calculates a median for each coordinate, i.e., for n-dimensional gradient vectors, n medians need to be calculated. Next, the CP encrypts the medians, and sends them to the SP. Finally, the SP eliminates the noise to obtain the desired results. Fig.4 shows the details of protocol *SecMed*.

Input: SP has $[\![\boldsymbol{G_x}]\!]_{pk_c}$ for $x \in [1, m]$; CP holds the private key sk_c.
Output: For $i \in [1, n]$, the median of $[\![G_{xi}]\!]_{pk_c}$: $[\![G_{(med)i}]\!]_{pk_c}$
Procedure:

```
SP:
  1. Randomly selects $n$ nonzero integers $r_i$ for $i \in [1, n]$.
  2. For $i \in [1, n]$, calculates as follow:
         $R'_{xi} \leftarrow [\![G_{xi}]\!]_{pk_c} \cdot [\![r_i]\!]_{pk_c}$
  3. Sends $\{R'_{xi}\}_{i=1}^{i=n}$ to CP.
CP:
  1. For $i \in [1, n]$, decrypts with the private key $sk_c$:
         $d'_{xi} \leftarrow Dec(sk_c, R'_{xi})$
  2. For $i \in [1, n]$, obtains the medians:
         $d'_{(med)i} \leftarrow \text{median}(d'_{1i}, d'_{2i}, \ldots, d'_{mi})$
  3. For $i \in [1, n]$, encrypts $[\![d'_{(med)i}]\!]_{pk_c}$ with the public key $pk_c$.
  4. Sends $\left\{[\![d'_{(med)i}]\!]_{pk_c}\right\}_{i=1}^{i=n}$ to SP.
SP:
  1. For $i \in [1, n]$, computes:
         $G_{(med)i} \leftarrow [\![d'_{(med)i}]\!]_{pk_c} \cdot [\![r_i]\!]_{pk_c}^{-1}$
```

Fig. 4. *SecMed*: Protocol to calculate the coordinate-wise medians without revealing privacy.

```
Input: SP has $[\![\boldsymbol{G_x}]\!]_{pk_c}$ and $[\![\boldsymbol{G_y}]\!]_{pk_c}$; CP holds the private key $sk_c$.
Output: The Pearson correlation coefficient $\rho_{x,y}$.
Procedure:
SP:
  1. Randomly selects two nonzero integers $r_i$ for $i \in \{x, y\}$.
  2. For $i \in \{x, y\}$, calculates: $R_i \leftarrow [\![\boldsymbol{G_i}]\!]_{pk_c}^{r_i}$.
  3. Sends $\boldsymbol{R_x}$ and $\boldsymbol{R_y}$ to CP.
CP:
  1. For $i \in \{x, y\}$, decrypts with the private key $sk_c$:
         $\boldsymbol{d_i} \leftarrow \mathrm{Dec}(sk_c, \boldsymbol{R_i})$
  2. Calculates the Pearson correlation coefficient:
         $\rho_{x,y} = \frac{Cov(\boldsymbol{d_x}, \boldsymbol{d_y})}{\sigma(\boldsymbol{d_x}) \cdot \sigma(\boldsymbol{d_y})}$
  3. Sends $\rho_{x,y}$ to SP.
```

Fig. 5. *SecPear*: Protocol to calculate the Pearson correlation coefficient without revealing privacy.

In the PEFL, the coordinate-wise medians are considered as the benchmark, i.e., we calculate the Pearson correlation coefficient $\rho_{x,y}$ between the coordinate-wise medians and the gradient of the user U_x^t. To calculate the correlation without revealing privacy, i.e., the gradient vectors still keep confidential for both SP and CP, we propose a secure protocol *SecPear*. Specifically, the SP firstly obscures the gradient vectors by calculating the ciphertext of the gradient vectors to the r-th power, where r is a randomly sampled nonzero integer. After decrypting the above ciphertexts, the CP calculates the correlation of obscured variables and sends the results to the SP. Fig.5 depicts the protocol *SecPear*. The correctness and security are discussed in Section IV-C2 and V-A, respectively.

For m local gradients, the SP calls m times the protocol *SecPear* to obtain m correlation coefficients. Then, each coefficient is re-scaled as below:

$$\mu_x \leftarrow \max\{0, \ln(\frac{1+\rho_{x,y}}{1-\rho_{x,y}}) - 0.5\} \tag{6}$$

Recall that we consider the users with lower correlation to be poisoners. The logarithmic function in (6) further encourages a higher divergence for values that are near the two tails of the function, so that the malicious behaviors are impeded. Simultaneously, honest users with low correlation are avoided from being punished as far as possible.

Given the re-scaled values, each gradient vector is assigned a weight. The SP then updates the shared model under the ciphertext domain as below:

$$\boldsymbol{\omega^t} \leftarrow \boldsymbol{\omega^{t-1}} - \eta \sum_{x\in[1,m]} \frac{\mu_x}{m\sum_{x\in[1,m]}\mu_x}\boldsymbol{G_x} \tag{7}$$

The protocol *SecAgg* is designed to implement the above process without revealing $\boldsymbol{G_x}$. Fig.6 illustrates the details.

Finally, the SP communicates with the CP for obtaining re-encrypted global parameters $[\![\boldsymbol{\omega^t}]\!]_{pk_x}$ with the public key pk_x held by all users, and then broadcasts to all users for further training. Fig.7 shows the details of the protocol *SecExch*.

```
Input: SP has $[\![\boldsymbol{\omega^{t-1}}]\!]_{pk_c}$ and $[\![\boldsymbol{G_x}]\!]_{pk_c}$ for $x \in [1, m]$; CP holds
the private key $sk_c$ and $\mu_x$ for $x \in [1, m]$.
Output: $[\![\boldsymbol{\omega^t}]\!]_{pk_c}$.
Procedure:
SP:
  1. Randomly selects $m$ nonzero integers $r_x$ for $x \in [1, m]$.
  2. For$x \in [1, m]$, calculates as follow:
         $\boldsymbol{R'_x} \leftarrow [\![\boldsymbol{G_x}]\!]_{pk_c} \cdot [\![r_x]\!]_{pk_c}$
  3. Sends $\{\boldsymbol{R'_x}\}_{x=1}^{x=m}$ to CP.
CP:
  1. For $x \in [1, m]$, decrypts with the private key $sk_c$:
         $\boldsymbol{d'_x} \leftarrow Dec(sk_c, \boldsymbol{R'_x})$
  2. For $x \in [1, m]$, calculates:
         $\boldsymbol{e_x} \leftarrow \eta \cdot \frac{\mu_x}{m \cdot \sum_{x\in[1,m]}\mu_x} \cdot \boldsymbol{d'_x}$
  3. For $x \in [1, m]$, encrypts $\boldsymbol{E_x} \leftarrow [\![\boldsymbol{e_x}]\!]_{pk_c}$ with the public
     key $pk_c$.
  4. Sends $\{\boldsymbol{E_x}\}_{x=1}^{x=m}$ to SP.
SP:
  1. For $x \in [1, m]$, eliminates the noise as follows:
         $\boldsymbol{F_x} \leftarrow \boldsymbol{E_x} \cdot [\![\frac{\eta \cdot \mu_x}{m \cdot \sum_{x\in[1,m]}\mu_x} \cdot r_x]\!]_{pk_c}^{-1}$
  2. Updates the global model as:
         $[\![\boldsymbol{\omega^t}]\!]_{pk_c} \leftarrow [\![\boldsymbol{\omega^{t-1}}]\!]_{pk_c} \cdot (\prod_{x\in[1,m]} \boldsymbol{F_x})^{-1}$
```

Fig. 6. *SecAgg*: Protocol to aggregate the parameters without revealing privacy.

```
Input: SP has $[\![\boldsymbol{\omega^t}]\!]_{pk_c}$; CP holds the private key $sk_c$.
Output: Encrypted parameters $[\![\boldsymbol{\omega^t}]\!]_{pk_x}$ with the public key of
participants $pk_x$.
Procedure:
SP:
  1. Randomly selects $n$-dimensional nonzero integer vector $\boldsymbol{r}$.
  2. Calculates: $\boldsymbol{R} \leftarrow [\![\boldsymbol{\omega^t}]\!]_{pk_c} \cdot [\![\boldsymbol{r}]\!]_{pk_c}$.
  3. Sends $\boldsymbol{R}$ to CP.
CP:
  1. Decrypts with the private key $sk_c$: $\boldsymbol{d} \leftarrow \mathrm{Dec}(sk_c, \boldsymbol{R})$.
  2. Re-encrypts with the public key $pk_x$: $\boldsymbol{R'} \leftarrow [\![\boldsymbol{d}]\!]_{pk_x}$.
  3. Sends $\boldsymbol{R'}$ to SP.
SP:
  1. Eliminates the noise: $[\![\boldsymbol{\omega^t}]\!]_{pk_x} \leftarrow \boldsymbol{R'} \cdot [\![\boldsymbol{r}]\!]_{pk_x}^{-1}$.
  2. Sends $[\![\boldsymbol{\omega^t}]\!]_{pk_x}$to all participants.
```

Fig. 7. *SecExch*: Protocol to exchange the parameters without revealing privacy.

Remark: In this paper, we consider the privacy of the Pearson correlation coefficient does not need to be protected. It is clear that the servers cannot invert the magnitude and sign of gradients even if the coefficient is public. Recently, Geiping *et al.* [5] proposed that the similarity between gradients than the gradient value contains more information about the training data. They also designed an optimization scheme based on the cosine similarity between gradients to reconstruct the training samples. However, the scheme relies heavily on the sign of the gradient, which is hidden information in the PEFL. Therefore, our scheme invalidates the data reconstruction attack. We admit that the coefficients could also be used to carry out some attacks, e.g., model steal attack. However, it is beyond the scope of this paper. The issue remains open.

C. Discussion

Here, we discuss how to optimize the above protocols, and give detailed proof of correctness for the protocol *SecPear*.

1) Optimization: We reduce the communication overhead of our basic protocols using the following optimizations.

- Removing redundancy: Observe that there exists redundancy in m calls to the protocol *SecPear* for calculating the Pearson correlation coefficient between the gradient vectors submitted by the users and the coordinate-wise medians. The SP repeatedly sends the medians to the CP on each invocation of the protocol *SecPear*. Hence, in the optimized protocol, the SP only sends once for the duplicate item.
- Reducing the communication round: We observe that there exist two independent relationships in the protocol *SecPear* and *SecAgg*. One is that for all $x \in [1, m]$, $\boldsymbol{R_x}$ is independent of each other in protocol *SecPear*. The other is that $\boldsymbol{R_x}$ in protocol *SecPear* and $\boldsymbol{R'_x}$ in protocol *SecAgg* are mutually independent. Therefore, for $x \in [1, m]$, the SP reduces the communication round by sending $\boldsymbol{R_x}$ and $\boldsymbol{R'_x}$ in a single round.

2) Correctness: To ensure that the PEFL can effectively identify malicious gradients, we need to ensure that the correlation coefficient between the gradients can still be correctly obtained from the obscured data in the protocol *SecPear*.

Proposition 1 (Correctness): Given perturbed gradient vectors, the protocol SecPear can correctly obtain the Pearson correlation coefficient between the gradient vectors submitted by the users and their coordinate-wise medians.

Proof: According to the homomorphism of encryption primitives [30], we have: $\boldsymbol{d_x} = r_x \boldsymbol{G_x}$ in the protocol *SecPear*. To prove the correctness, it is equivalent to prove: $\rho_{x,y} = \rho_{d_x,d_y}$.

According to the definition of the Pearson correlation coefficient in Section IV-A, we consider the covariance and the standard deviation of the variables separately. At first, the formula of the covariance is as follows:

$$\begin{aligned} Cov(\boldsymbol{d_x}, \boldsymbol{d_y}) &= Cov(r_x \cdot \boldsymbol{G_x}, r_y \cdot \boldsymbol{G_y}) \\ &= E[(r_x \boldsymbol{G_x} - \overline{r_x \boldsymbol{G_x}})(r_y \boldsymbol{G_y} - \overline{r_y \boldsymbol{G_y}})] \\ &= r_x r_y E[(\boldsymbol{G_x} - \overline{\boldsymbol{G_x}})(\boldsymbol{G_y} - \overline{\boldsymbol{G_y}})] \\ &= r_x r_y Cov(\boldsymbol{G_x}, \boldsymbol{G_y}) \end{aligned}$$

where $E(\boldsymbol{G_x})$ denotes the expectation of the variable $\boldsymbol{G_x}$, and $\overline{\boldsymbol{G_x}}$ is the average of variable $\boldsymbol{G_x}$. Similarly, the standard deviation scales with the magnitude of the random variable as follows:

$$\begin{aligned} \sigma(r_x \boldsymbol{X}) &= \sqrt{E((r_x \boldsymbol{G_x})^2) - (E(r_x \boldsymbol{G_x}))^2} \\ &= r_x \sqrt{E((\boldsymbol{G_x})^2) - (E(\boldsymbol{G_x}))^2} \\ &= r_x \sigma(\boldsymbol{G_x}) \end{aligned}$$

Thus, the equation is satisfied as below:

$$\rho_{\boldsymbol{d_x},\boldsymbol{d_y}} = \frac{Cov(\boldsymbol{d_x}, \boldsymbol{d_y})}{\sigma(\boldsymbol{d_x})\sigma(\boldsymbol{d_y})} = \frac{r_x r_y Cov(\boldsymbol{G_x}, \boldsymbol{G_y})}{r_x r_y \sigma(\boldsymbol{G_x})\sigma(\boldsymbol{G_y})} = \rho_{x,y}$$

In summary, the Pearson correlation coefficient can be correctly calculated even given the obscured variables.

V. Analysis

In this section, we provide comprehensive proof of the security property and convergence property of our scheme, and briefly analyze the efficiency and functionality of the PEFL.

A. Security Property

The function privacy and semantic security of the LHE against chosen-plaintext attacks (or IND-CPA security for short) [30] guarantee the strong security of the protocol. In this section, we provide a hybrid argument that relies on simulators so as to further demonstrate that during the execution of the protocol, the joint view of the servers does leak nothing about the users' private data except for the information inferred from the results of the computation.

Proposition 2 (Honest But Curious Security, With Curious Servers): Given a security parameter κ, any subset of users U and $C = \{SP, CP\}$, let $\mathbf{REAL}_{\mathbf{C}}^{\mathbf{U},\kappa}$ be a random variable representing the joint view of parties in C in the real execution of above protocols. There exists a probabilistic polynomial-time (PPT) simulator **SIM** *such that the output of* **SIM** *is computationally indistinguishable from $\mathbf{REAL}_{\mathbf{C}}^{\mathbf{U},\kappa}$:*

$$\mathbf{REAL}_{\mathbf{C}}^{\mathbf{U},\kappa} \equiv \mathbf{SIM}_{\mathbf{C}}^{\mathbf{U},\kappa}$$

where "≡" represents computationally indistinguishable.

Proof: According to the definition of $\mathbf{REAL}_{\mathbf{C}}^{\mathbf{U},\kappa}$, it consists of all internal state and messages received by the parties in C during the execution of the protocols. We adopt the standard hybrid argument used in [13], [17] to prove this proposition, i.e., given the security parameters κ, we define a PPT simulator **SIM** through a series of (polynomially many) subsequent modifications to the random variables in $\mathbf{REAL}_{\mathbf{C}}^{\mathbf{U},\kappa}$, so that the output of **SIM** is computationally indistinguishable from $\mathbf{REAL}_{\mathbf{C}}^{\mathbf{U},\kappa}$. The detailed proof is described below.

$\mathbf{Hyb_1}$: We initialize a random variable whose distribution is indistinguishable from $\mathbf{REAL}_{\mathbf{C}}^{\mathbf{U},\kappa}$, the joint views of parties in C in the real protocol execution.

$\mathbf{Hyb_2}$: In this hybrid, we change the behavior of simulated honest users $U_x \in U$, so that each user U_x encrypts a randomly selected vector $\boldsymbol{\beta_x}$ with public key pk_c using the LHE, instead of the original gradient vector $\boldsymbol{G_x}$. Since only the contents of the ciphertexts are changed, the IND-CPA security property of the LHE [30], as well as the two non-collusive SP and CP setting guarantees that this hybrid is indistinguishable from the previous one.

$\mathbf{Hyb_3}$: In this hybrid, we simulate the SP to perturb the β_{xi} by the noise ζ_i, which is sampled uniformly at random, instead of $[\![G_{xi}]\!]_{pk_c} \cdot [\![r_i]\!]_{pk_c}$. It is well known that the parameters added by uniformly random numbers are also uniformly random. Since the distribution of the noise ζ is uniformly random which is exactly the same as the previous one, and the perturbed parameters are also uniformly random, this hybrid here and the previous one are sampled from identical distribution, i.e., uniformly random. Besides, the IND-CPA security property of the LHE, as well as the two non-collusive SP and CP setting guarantees that this hybrid is indistinguishable from the previous one.

Hyb$_4$: In this hybrid, we change the input of protocol *SecPear* executed by SP and CP with $[\![\boldsymbol{\beta_x}]\!]_{pk_c}$ and $[\![\theta_i]\!]_{pk_c}$ instead of $[\![\boldsymbol{G_x}]\!]_{pk_c}$ and $[\![G_{(med)i}]\!]_{pk_c}$. The IND-CPA security property of the LHE, as well as the two non-collusive SP and CP setting guarantees that this hybrid is indistinguishable from the previous one.

Hyb$_5$: In this hybrid, for all users $U_x \in U$, we simulate the SP to compute $[\![\boldsymbol{\beta_x}]\!]_{pk_c}^{\zeta_x}$ instead of $[\![\boldsymbol{G_x}]\!]_{pk_c}^{r_x}$, where $x \in \{1, m\}$. Although the CP holds the private key sk_c which can be used to decrypt the above ciphertexts, the parameters multiplied by a random number are still uniformly random, which is consistent with the previous hybrid. Hence, this hybrid is indistinguishable from the previous one.

Hyb$_6$: In this hybrid, we change the input of protocol *SecAgg* executed by SP and CP with $[\![\boldsymbol{\xi^{t-1}}]\!]_{pk_c}$ and $[\![\boldsymbol{\beta_x}]\!]_{pk_c}$ instead of $[\![\boldsymbol{\omega^{t-1}}]\!]_{pk_c}$ and $[\![\boldsymbol{G_x}]\!]_{pk_c}$. Since only the contents of the ciphertexts are changed, the IND-CPA security of the LHE, as well as the two non-collusive SP and CP setting guarantees that this hybrid is indistinguishable from the previous one.

Hyb$_7$: This hybrid is similar to **Hyb$_3$**, for all users $U_x \in U$, we simulate the SP to perturb the $\boldsymbol{\beta_x}$ by the random noise ζ_x instead of $[\![\boldsymbol{G_x}]\!]_{pk_c} \cdot [\![r_x]\!]_{pk_c}$. The IND-CPA security property of the LHE, as well as the two non-collusive SP and CP setting guarantees that this hybrid is identically distributed to the previous one. Hence, this hybrid is indistinguishable from the previous one.

Hyb$_8$: In this hybrid, we change the input of protocol *SecExch* executed by SP and CP with $[\![\boldsymbol{\xi^t}]\!]_{pk_c}$ instead of $[\![\boldsymbol{\omega^t}]\!]_{pk_c}$. The IND-CPA security property of the LHE, as well as the three non-collusive entities (i.e., SP, CP and users) setting guarantees that this hybrid is indistinguishable from the previous one.

Hyb$_9$: Similar to **Hyb$_3$**, instead of sending: $[\![\boldsymbol{r} + \boldsymbol{\omega^t}]\!]_{pk_c} = [\![\boldsymbol{\omega^t}]\!]_{pk_c}[\![\boldsymbol{r}]\!]_{pk_c}$, we substitute the parameters sent to the CP for re-encrypting with $[\![\boldsymbol{\xi^t}]\!]_{pk_c}[\![\boldsymbol{\zeta^t}]\!]_{pk_c}$. Even the CP holding the private key sk_c can decrypt the above ciphertexts, the hybrid and $(\boldsymbol{r} + \boldsymbol{\omega^t})$ have identical distribution, i.e., uniformly random. Thus, this hybrid is indistinguishable from the previous one.

The argument proves that there is a simulator **SIM** sampled from the distribution described above so that its output is computationally indistinguishable from the output of **REAL**. Hence, the PEFL holds the security property that the curious SP and CP learn nothing about users' private data.

Proposition 3: The PEFL holds the security property that the malicious users cannot compromise other users' privacy.

Proof: In our threat model, the malicious users as active adversaries launch attacks and compromise other users' privacy. As indicated in [7], they deduce if there exist the target samples in the training dataset by exploiting the erroneous gradient vectors. The attack depends heavily on the aggregation rule by average. In this paper, each gradient vector is adaptively given a weight instead of treating them equally. According to experimental results in Section VI-B2, the PEFL reduces the success rate of the attack to 0.04, i.e., the malicious users fail to infer whether the target sample is used during the training.

Hence, the PEFL holds the security property that malicious users can't compromise other users' privacy.

B. Convergence Property

Now, we prove the convergence of the proposed scheme.

Claim 4 (Error Term): There exists an error term $\boldsymbol{\epsilon}$ *between the malicious gradients and benign ones such that* $\sum_{x\in F} \boldsymbol{G_x^\tau} = \sum_{x\in H} \boldsymbol{G_x} + \boldsymbol{\epsilon}$ *holds.*

Proof: We generalize the process of poisoning attacks to seek a set of model parameters $\boldsymbol{\omega^\tau}$ with malicious goals, which differs from the target model $\boldsymbol{\omega^\star}$ of honest users, i.e., $\boldsymbol{\omega^\star} \neq \boldsymbol{\omega^\tau}$. Let the distance between $\boldsymbol{\omega^\star}$ and $\boldsymbol{\omega^\tau}$ be $\boldsymbol{\hbar}$, i.e., $\boldsymbol{\hbar} = \boldsymbol{\omega^\star} - \boldsymbol{\omega^\tau}$. Note that malicious users and honest users can only achieve their goals by submitting gradients.

Now we formally define the classic poisoning attack model. Suppose that in a specific round, the correct gradient vectors $\{\boldsymbol{G_x}, x \in [1, m]\}$ are independent and identically distributed (i.i.d.) samples drawn from the random variable $\boldsymbol{G} \leftarrow \triangledown_{\boldsymbol{\omega}} L(B, \boldsymbol{\omega})$, where $E[\boldsymbol{G}] = \boldsymbol{g}$ is an unbiased estimator of the gradient. Thus, $E[\boldsymbol{G_x}] = E[\boldsymbol{G}] = \boldsymbol{g}$, for any $x \in [1, m]$. With the malicious users, the actual vectors $\{\tilde{\boldsymbol{G}}_{\boldsymbol{x}}, x \in [1, m]\}$ received by the SP are as follows:

$$\tilde{\boldsymbol{G}}_{\boldsymbol{x}} = \begin{cases} \boldsymbol{G_x}, & \text{if the } x\text{-th user is honest,} \\ \boldsymbol{G_x^\tau}, & \text{if the } x\text{-th user is malicious.} \end{cases}$$

Note that a very important assumption for the convergence properties of SGD algorithm is that each benign gradient is an unbiased estimation of the actual gradient, which is typically ensured through uniform random sampling. However, the gradients submitted by malicious users break the restriction of uniform random sampling. Therefore, there is a clear difference between malicious gradient vectors and benign gradient vectors.

According to the definition of the SGD algorithm used to update the model parameters, we have:

$$\boldsymbol{\omega^t} \leftarrow \boldsymbol{\omega^{t-1}} - \eta \frac{\sum_{x\in[1,m]} \tilde{\boldsymbol{G}}_{\boldsymbol{x}}}{m}$$

Given the gradients of malicious users $\{\boldsymbol{G_x^\tau}, x \in F\}$, where F represents the set of malicious users, the goal of malicious users is to obtain parameters of the following form: $\boldsymbol{\omega^\tau} \leftarrow \boldsymbol{\omega^{t-1}} - \eta \frac{\sum_{x\in F} \boldsymbol{G_x^\tau}}{|F|}$. Analogously, given the gradients of honest users $\{\boldsymbol{G_x}, x \in H\}$, where H represents the set of honest users, honest users aim to obtain parameters of the following form: $\boldsymbol{\omega^\star} \leftarrow \boldsymbol{\omega^{t-1}} - \eta \frac{\sum_{x\in H} \boldsymbol{G_x}}{|H|}$. According to the parameter distance defined previously (i.e., $\boldsymbol{\hbar} = \boldsymbol{\omega^\star} - \boldsymbol{\omega^\tau}$), we have:

$\boldsymbol{\hbar} = \eta\frac{\sum_{x\in F}\boldsymbol{G_x^\tau}}{|F|} - \eta\frac{\sum_{x\in H}\boldsymbol{G_x}}{|H|}$. And then we get:

$$\begin{aligned}\sum_{x\in F}\boldsymbol{G_x^\tau} &= \frac{|F|}{\eta}\boldsymbol{\hbar} + \frac{\sum_{x\in H}\boldsymbol{G_x}}{|H|}|F| \\ &= \frac{|F|}{\eta}\boldsymbol{\hbar} + (\frac{|F|}{|H|} - 1)\sum_{x\in H}\boldsymbol{G_x} + \sum_{x\in H}\boldsymbol{G_x} \\ &= \frac{|F|}{\eta}\boldsymbol{\hbar} + (\frac{|F|}{|H|} - 1)|H|\boldsymbol{g} + \sum_{x\in H}\boldsymbol{G_x}\end{aligned}$$

Thus, there exists an error term $\boldsymbol{\epsilon}$ between the malicious gradients and benign ones because of the different purposes, i.e.,

$$\sum_{x\in F}\boldsymbol{G_x^\tau} = \sum_{x\in H}\boldsymbol{G_x} + \boldsymbol{\epsilon}$$

where $\boldsymbol{\epsilon} = \frac{|F|}{\eta}\boldsymbol{\hbar} + (\frac{|F|}{|H|} - 1)|H|\boldsymbol{g}$.

Proposition 5 (Convergence): Given the detailed steps of the PEFL as described in Section IV-B, the convergence rate of malicious or/and honest users is $O(\frac{1}{T^2})$ over T iterations.

Proof: The PEFL follows the outline of the adaptive learning rate method [35], which has been applied to the SGD algorithm and provided a convergence guarantee. As we know, it is $O(\frac{1}{T^2})$ that the convergence rate of the SGD algorithm with a constant learning rate.

Assume that the adaptive learning rate $\eta_x = \eta \times \frac{\mu_x}{\sum_{x\in[1,m]}\mu_x}$ of user U_x is decided by a function $f_s(x,t)$, where t represents the current iteration. U_x comes from either F or H, which denote the set of malicious users and honest ones in the protocol, respectively. As mentioned earlier, the percentage of the malicious users is less than 50% in this paper, i.e., $2|F| < |F|+|H|$. The convergence property of SGD can be applied to our solution as long as the training is performed with the honest users' data, which satisfies the following conditions:

$$\begin{aligned}&\textit{Condition 1:}\ \ \forall x \in F, f_s(x,t) \to 0 \\ &\textit{Condition 2:}\ \ \forall x \in H, f_s(x,t) \to 1\end{aligned}$$

We consider $\boldsymbol{G_x^\star}$ to be the ideal gradient from the initial model $\boldsymbol{\omega_{init}}$ to optimized model $\boldsymbol{\omega_x^\star}$ relative to U_x's local data. According to Claim 4, we know that: $\sum_{x\in F}\boldsymbol{G_x^\tau} = \sum_{x\in H}\boldsymbol{G_x^\star}+\boldsymbol{\epsilon}$. In this regard, the PEFL can identify abnormalities based on the difference and reduce the weight of malicious gradients. If the malicious goal has been achieved, $\boldsymbol{\epsilon}$ increases as the number of iterations increases. That causes that the rescaled coefficient approaches 0, i.e., $f_s(x,t) \to 0$. Hence, the weights of poisoners are 0, which satisfies the *Condition 1*. Accordingly, the effect of the PEFL on honest users satisfies:

$$\lim_{t\to\infty}\sum_{x\in H}\frac{\mu_x}{\sum_{x\in[1,m]}\mu_x} = 1$$

That is, the sum of the weights of the honest users approaches 1, which satisfies the *Condition 2*.

The above proof demonstrates that the PEFL fulfills both conditions. Therefore, the convergence rate of the PEFL is the same as the SGD with the adaptive learning rate, i.e., $O(\frac{1}{T^2})$.

C. Efficiency Evaluation

We now discuss the communication and computation overhead of the PEFL. As discussed in Section IV-C1, we optimize the communication overhead of the basic PEFL. In the following, we denote PEFL$_{imd}$ as the optimized scheme.

1) Communication Overhead: In the PEFL, the SP interacts with the CP to perform four secure protocols for secure aggregation. What needs to be mentioned is that the increased factor in communication overhead using ciphertext packing technology is around $\sigma = 2(1+\frac{pad}{prec})$ as discussed in [16]. For example, we take $pad = 15$, and $prec = 32$, so the increased communication factor is 2.93. In the *SecMed*, the SP interacts with the CP to obtain n coordinate-wise medians, where n represents the dimensionality of the gradient vector. Hence, the communication overheads of the SP and CP are $O(\sigma Tnm)$ and $O(\sigma Tn)$, respectively, where T is the number of iterations, and m represents the number of users. For each user, the protocol *SecPear* is invoked once at each iteration, so the communication round is $O(Tm)$. We note that the SP sends two encrypted vectors to the CP in the *SecPear*, which incurs the communication cost of $O(2\sigma Tnm)$. For the CP, it sends m Pearson correlation coefficients in plaintext to the SP. Thus, the communication round and cost should be $O(Tm)$ and $O(Tm)$, respectively. Then, the protocol *SecAgg* is called to securely update model parameters.

For each iteration, both SP and CP communicate m obscured gradient vectors in the form of ciphertext to each other, so the communication rounds and costs are the same, $O(T)$ and $O(\sigma Tnm)$. Finally, for both SP and CP, the protocol *SecExch* incurs the same communication rounds and costs, i.e., $O(T)$ and $O(\sigma Tn)$, respectively.

In the PEFL$_{imd}$, the SP and CP only interact once to communicate all the data used in the protocol *SecPear* and *SecAgg*. Hence, the communication round of the SP and CP in protocol *SecPear* should be $O(T)$, and 0 for the SP to call protocol *SecAgg*. In this round of communication, the SP sends the encrypted gradient vectors that have been obscured in two ways, as well as their coordinate-wise medians. Therefore, the communication overhead of the SP is $O((2m+1)\sigma Tn)$. The data sent by the CP is the same as that in the PEFL. So the communication overhead is the same. As described in TABLE II, we can observe that PEFL$_{imd}$ is superior to basic PEFL in terms of communication overhead.

2) Computation Overhead: In the PEFL, the main limitation lies in the computation costs of the robust aggregation phase. In this phase, the SP interacts with the CP for calculating the coordinate-wise medians, Pearson correlation coefficient and weights of m gradients vectors in the ciphertext domain. To compute the coordinate-wise medians, the easiest method is to exploit some sorting algorithm with the complexity of $O(nm\log m)$ to each dimension. In the PEFL, we obtain the medians by exploiting the BFPRT algorithm, which is a famous algorithm to solve the classical problem of choosing the k-th largest or k-th smallest number from m numbers, with the worst time complexity of $O(m)$. Hence, the computation cost of the protocol *SecMed* on average takes linear time $O(nm)$. For each user, we only need to calculate the

TABLE II
COMPARISON OF COMMUNICATION OVERHEAD BETWEEN PEFL AND PEFL$_{imd}$

Protocol	Scheme	SP		CP	
		# of Round	Communication Complexity	# of Round	Communication Complexity
SecPear	PEFL	$O(Tm)$	$O(2\sigma Tnm)$	$O(Tm)$	$O(Tm)$
	PEFL$_{imd}$	$O(T)$	$O((2m+1)\sigma Tn)$	$O(T)$	$O(\sigma Tnm)$
SecAgg	PEFL	$O(T)$	$O(\sigma Tnm)$	$O(T)$	$O(Tm)$
	PEFL$_{imd}$	0	0	$O(T)$	$O(\sigma Tnm)$

TABLE III
COARSE-GRAINED COMPARISON

Scheme	# of Poisoners	Computation Complexity	Redundancy or Not	No prior knowledge about # of Poisoners	Protect Privacy against Users	Protect Privacy against Servers
Krum [6]	$< 50\%$	Medium	$\surd$	$\times$	$\times$	$\times$
Bulyan [24]	$< 25\%$	Medium $\sim$ High	$\surd$	$\times$	$\times$	$\times$
Trimmed Mean [25]	$< 50\%$	Low	$\surd$	$\times$	$\times$	$\times$
DETOX [20]	$< 50\%$	Low	$\times$	$\surd$	$\times$	$\times$
Client Detection [36]	$< 50\%$	Low	$\times$	$\surd$	$\surd$	$\times$
PEFL	$< 50\%$	Low	$\surd$	$\surd$	$\surd$	$\surd$

correlation once. Therefore, the computation overhead incurred by the protocol *SecPear* is also $O(nm)$. Besides, the parameter update phase of PEFL does not introduce additional computational complexity, except for adjusting the weight which requires $O(m)$ computational complexity. Overall, the computation complexity incurred in the defense phase is $O(Tnm)$. On the users' side, the computation cost increases linearly with the number of the model parameters. This makes the computation cost of each user as $O(Tns)$, where s is the number of samples held by each user.

Compared with *Krum* [6] and *Bulyan* [24], whose computation cost increases quadratically with the increase in the number of users, the PEFL is more advantageous in terms of computation cost. We also provide a coarse-grained comparison with the other five state-of-the-art works in terms of computation overhead, which is illustrated in TABLE III.

D. Functionality

Here, we analyze the functional advantages of the PEFL by comparing with five state-of-the-art robust schemes, which are *Krum* [6], *Bulyan* [24], *Trimmed Mean* [25], *Detox* [20] and *Client Detection* [36]. TABLE III presents the coarse-grained comparison in terms of the number of poisoners, computation complexity, redundancy or not, prior knowledge of the numbers of poisoners or not, and privacy.

Since *Bulyan* is an improved version of *Krum*, *Bulyan* has stricter limits on the number of poisoners and incurs higher computation complexity. The runtime per iteration of both *Krum* and *Bulyan* is often quadratic in the number of users, or even higher. While the *Trimmed Mean* and PEFL are computationally effective, almost nearly linear in the number of users. The computation complexity of *Detox* and *Client Detection* is also almost remarkably correlated linearly with the number of users, since both of them require heavy redundancy to filter out almost all malicious gradients.

Besides, the design of *Krum*, *Bulyan* and *Trimmed Mean* requires the number of poisoners as prior knowledge, which contributes to improving the effectiveness of the defenses. In real scenarios, this assumption is unrealistic, i.e., the identity and number of the adversary cannot be perceived in advance. For each poisoned sub-model in both *Detox* and *Client Detection*, more than half of the users need to report exceptions. So they do not need this information as prior knowledge. In the PEFL, we focus on designing a practical and effective robust aggregation rule, our scheme thus does not consider the assumption to be reasonable.

The solution of *client detection* [36] claims to protect users' data privacy against malicious users, which achieves by a trusted server using differential privacy to perturb the model parameters from being compromised. However, both the differential privacy mechanism and client-side detection cause an increase in the probability of false positives. Besides, the server is not always trusted. According to prior works [17], the server could compromise the users' data privacy. In the PEFL, the IND-CPA security property of the LHE and two non-collusive servers setting guarantee users' data privacy. Security proof is given in Section V-A.

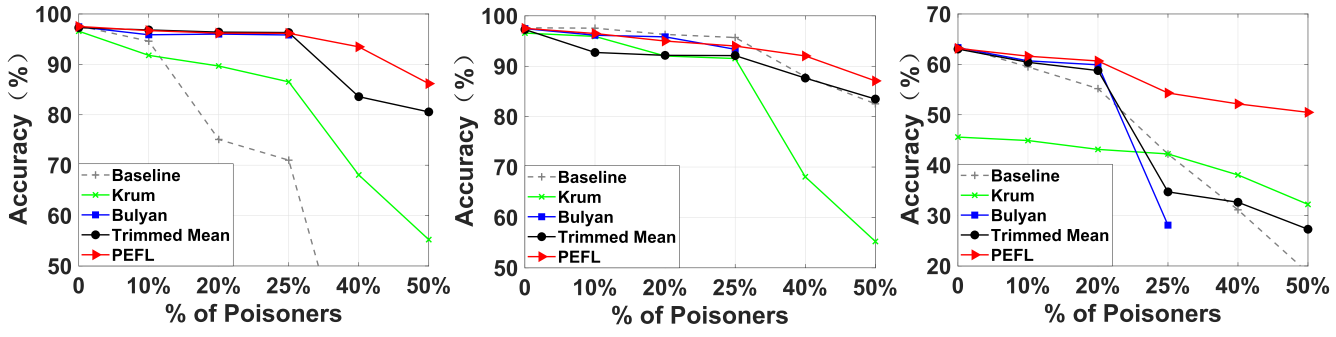
VI. PERFORMANCE EVALUATION

In this section, we conduct experiments with real-world datasets to evaluate the performance of the PEFL. All experiments are run in a Lenovo server with the configuration of Ubuntu 18.04, Intel(R) Xeon(R) E5-2620 2.10 GHz CPU and 16GB RAM.

A. Experiment Setup

We present the performance of the PEFL from two aspects: a. Validating the claim regarding the accuracy. b. Validating the robustness of the PEFL against real attacks. To highlight the advantages of the PEFL, except for three typically robust schemes, i.e., *Krum* [6], *Bulyan* [24], *Trimmed Mean* [25], we establish a control group − unmodified FL as a baseline which takes average as the aggregation rule.

1) Attacks: We mainly focus on two representative poisoning attacks: label-flipping attack and backdoor attack. To simulate the label-flipping attack, we re-label the source class held by the malicious users as the target class. The source class



(a) T_{hdr} with the label-flipping attack (b) T_{hdr} with the backdoor attack (c) T_{ir} with the backdoor attack

Fig. 8. Comparison of accuracy with different number of poisoners.

and target class are 1 and 9 in our experiments, respectively. To reproduce the backdoor attack, in each round of training, malicious users randomly sample a certain number of images (e.g., 1000) from the private dataset, and cover the 5 × 5 pixels at the bottom right with the maximum intensity. All these modified pictures reset their label to the target class, e.g., the airplane for the CIFAR-10 dataset. The 5 × 5 pattern is considered as the trigger.

2) Datasets and Model Architectures: To evaluate the performance of the PEFL, we simulate two scenarios: handwritten digit recognition (T_{hdr}) and image recognition (T_{ir}). T_{hdr} aims to classify handwritten digits into one of 10 classes (0-9) according to the learned features. In this task, we use a simple two-layer fully connected network as the model structure, and the parameter settings of each layer are 724×100 and 100×10 in sequence. The data set is the classic MNIST dataset, including 60,000 handwritten digit pictures with a size of 28 × 28, and 10,000 test samples. For the T_{ir}, our experiments are implemented on the CIFAR-10 dataset which has a total of 60,000 color images. These images are 32 × 32 in size and are divided into 10 categories (such as airplanes, cars and birds), each with 6000 images. There are 50,000 sheets for training and another 10,000 for testing. The model architecture of this task is composed of two layers of convolution and three layers of fully connected. Moreover, the training data is distributed to each user by executing torch.utils.data.distributed.DistributedSampler().

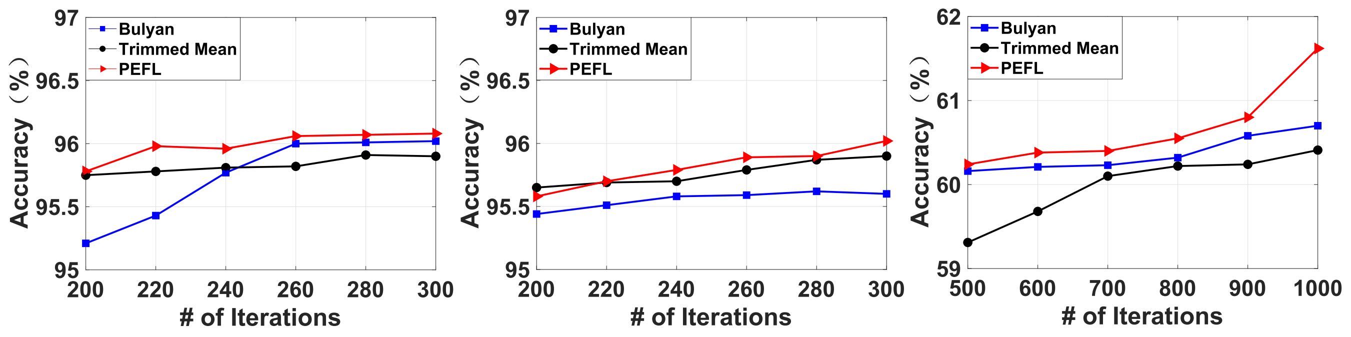
3) Hyper-Parameters: The total number of users we used in the experiment is 51. For each experiment, we set the batch size of 2^7, the momentum of 0.9 and the initial learning rate to 0.1. Each reported data takes an average of five experiments.

B. Experiment Results

1) Accuracy Evaluation: Many factors affect the model accuracy, including the model capacity, the quality and quantity of data, the number of iterations, and the proportion of malicious users. In our experiments, we use public datasets to evaluate the performance of our solution in two application scenarios, we thus only show the accuracy of the model by sliding the value of the latter two factors, freezing the capacity of the model and ignoring the problem of data quality and quantity. It is worth mentioning that the model architecture for T_{ir} is relatively simple, while the CIFAR-10 dataset with more complex features is under-fitting, resulting in low model accuracy. However, we only focus on a difference in accuracy between different defenses here. In the following, we describe the influence of these two factors on model accuracy.

a) Effect of different proportions of poisoners: Intuitively, the more benign data hold by honest users in the training brings about better model accuracy. Prior works have shown that the vanilla federated learning with the aggregation rule of average is vulnerable even if there is only one malicious participant. Fig.8 outlines the comparison of test accuracy with the different number of poisoners. *Bulyan* in the figure is incomplete because the upper limit of the scheme for the proportion of poisoners is 25%. We observe that the accuracy of all defenses decreases as the number of poisoners increases. We analyze this because as the proportion of poisoners increases, the amount of valuable data in the training process decreases, which leads to a drop in model accuracy. After the proportion of poisoning is more than 25%, we can obtain that the degree of decline became larger. However, no matter in which application scenario, the PEFL can still maintain a comparative advantage compared with other defenses. We attribute it to the benefit of the coordinate-wise medians and aggregating parameters around the medians. Another interesting thing is that the accuracy of the PEFL, *Bulyan* and *Trimmed Mean* is significantly higher than *Krum*. A reasonable explanation is the number of aggregated parameters. As already mentioned, *Krum* only selects a set of candidate parameters for updating, discarding the remaining $(m-1)$ sets of gradient vectors. While others aggregate benign parameters as many as possible.

b) Effect of different iterations: As the number of iterations increases, the model can extract more effective data features, which also leads to more accurate test accuracy of the model. Here, we ignore the problems of under-fitting and over-fitting caused by too small or large model capacity. Fig.9 shows the comparison of accuracy with different iterations. Moreover, due to the restriction of *Bulyan* on the proportion of poisoners, Fig.9a and Fig.9b fix 25% of poisoners, i.e., 12 poisoners are mixed in a total of 51 users. Fig.9c fixes 20% of poisoners since Fig.8c shows that the accuracy of the three schemes is very close when the proportion of poisoners is 20%. We can observe that the accuracy of the PEFL is slightly higher than *Bulyan* and *Trimmed Mean*. This is because the PEFL



(a) T_{hdr} with the label-flipping attack (b) T_{hdr} with the backdoor attack (c) T_{ir} with the backdoor attack

Fig. 9. Comparison of accuracy with different iterations.

TABLE IV
ACCURACY OF T_{hdr} UNDER THE LABEL-FLIPPING ATTACK

		No Poisoner	10% Poisoners	20% Poisoners	25% Poisoners	40% Poisoners	50% Poisoners
Baseline	%Eq(Pred(S),T)[1]	0.001	1.	1.	1.	1.	1.
	%Eq(Pred(S),S)[2]	0.98	0.10	0.03	0.01	0	0
	%Eq(Pred(R),R)[3]	0.95	0.91	0.89	0.77	0.68	0.51
Krum	%Eq(Pred(S),T)	0	0	0.002	0.008	0.02	0.05
	%Eq(Pred(S),S)	0.95	0.94	0.92	0.91	0.89	0.04
	%Eq(Pred(R),R)	0.90	0.89	0.90	0.90	0.89	0.11
Bulyan	%Eq(Pred(S),T)	0	0	0	0.02		
	%Eq(Pred(S),S)	0.98	0.97	0.97	0.95	–	–
	%Eq(Pred(R),R)	0.96	0.95	0.94	0.94		
Trimmed Mean	%Eq(Pred(S),T)	0	0	0	0	0.01	0.03
	%Eq(Pred(S),S)	0.95	0.95	0.94	0.93	0.92	0.76
	%Eq(Pred(R),R)	0.95	0.94	0.94	0.94	0.91	0.72
PEFL	%Eq(Pred(S),T)	0	0	0	0	0	0.02
	%Eq(Pred(S),S)	0.95	0.98	0.98	0.98	0.97	0.88
	%Eq(Pred(R),R)	0.97	0.95	0.95	0.95	0.92	0.76

[1] %Eq(Pred(S), T) means the proportion of the source class classified as the target class, that is, the attack success rate.
[2] %Eq(Pred(S), S) refers to the prediction accuracy of the source class as the source class, that is, the true positive rate.
[3] %Eq(Pred(R), R) is the classification accuracy of other classes except the source class, where R represents non-source classes.

adjusts the weight of each gradient, which further improves the reliability of the remaining parameters and benefits the convergence of the model.

2) *Robustness Evaluation:* The robustness in this paper is considered to be the ability to defend against poisoning attacks. We evaluate the robustness of defense schemes by describing the success rate of the attack. For the label-flipping attacks, we test the success rate of the attack by observing the proportion of the target class identified in the test set. In TABLE IV, we present the adversary's attack success rate, the classification accuracy of the source and non-source classes under different proportions of poisoners, where non-source classes refer to the set of all classes except the source class. We observe that the baseline is very vulnerable even if there are only a few poisoners, the attack success rate achieves as high as 100%. This also highlights the importance of our research direction. Observing the experimental results, we obtain that even if the source class held by a malicious user is flipped and labeled as the target class, the classification accuracy of the source class is slightly higher than that of other non-source classes. We analyze this because the data features of the source class with the label of 1 are easier to identify than other data in the experiment. To verify the defense effect of PEFL on different source and target classes pairs, Fig.10 enumerates the attack success rate (i.e., The ratio of the source class misclassified as the target class) of different source classes and target classes when the probability of the poisoner reaches 50%, where the Y-axis lists the source class, and the X-axis refers to the target class designated by the adversary. We can observe that the maximum attack success rate is 0.04, which also demonstrates that the PEFL is robust under the label-flipping attack.

Different from the label-flipping attack, for the backdoor attack, we add the trigger to the test samples and observe the classification of pictures with triggers. TABLE V lists three sets of data for each scheme under different proportions of malicious users: the proportion of target classes with triggers classified as target classes, the proportion of pictures with triggers classified as target classes, and the proportion of correct classification of non-target pictures with triggers. Our experiment results show that as the proportion of poisoning increases, the PEFL can still effectively defend against backdoor attacks and maintain high accuracy.

In summary, according to the experimental results, we can clearly obtain that the PEFL has shown superior performance than other solutions (i.e., *Krum*, *Bulyan* and *Trimmed Mean*)

TABLE V
ACCURACY OF T_{hdr} UNDER THE BACKDOOR ATTACK

		No Poisoner	10% Poisoners	20% Poisoners	25% Poisoners	40% Poisoners	50% Poisoners
Baseline	%Eq(Pred(S+□),T)[1]	0.001	1.	1.	1.	1.	1.
	%Eq(Pred(S+□),S)[2]	0.98	0.01	0.01	0.02	0.009	0.00
	%Eq(Pred(R+□),R)[3]	0.95	0.94	0.92	0.90	0.77	0.60
Krum	%Eq(Pred(S+□),T)	0.008	0.02	0.10	0.18	0.26	0.95
	%Eq(Pred(S+□),S)	0.95	0.91	0.84	0.82	0.71	0.04
	%Eq(Pred(R+□),R)	0.90	0.90	0.91	0.89	0.89	0.001
Bulyan	%Eq(Pred(S+□),T)	0.002	0.01	0.02	0.02		
	%Eq(Pred(S+□),S)	0.98	0.95	0.90	0.90	–	–
	%Eq(Pred(R+□),R)	0.94	0.93	0.91	0.89		
Trimmed Mean	%Eq(Pred(S+□),T)	0.002	0.10	0.13	0.17	0.28	0.49
	%Eq(Pred(S+□),S)	0.98	0.92	0.89	0.90	0.51	0.09
	%Eq(Pred(R+□),R)	0.94	0.86	0.90	0.84	0.58	0.07
PEFL	%Eq(Pred(S+□),T)	0.001	0.001	0.01	0.01	0.04	0.04
	%Eq(Pred(S+□),S)	0.96	0.95	0.96	0.95	0.93	0.86
	%Eq(Pred(R+□),R)	0.95	0.95	0.94	0.92	0.84	0.64

[1] %Eq(Pred(S+□),T) means the proportion of the source class with the trigger classified as the target class, where +□ represents the image with the trigger.
[2] %Eq(Pred(S+□),S) refers to the prediction accuracy of the source class with the trigger as the source class.
[3] %Eq(Pred(R+□),R) is the classification accuracy of non-source classes with the trigger.

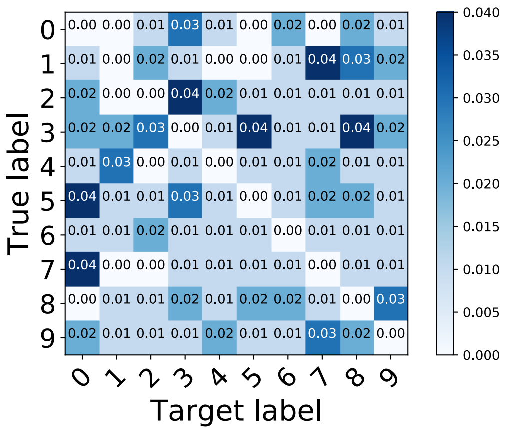
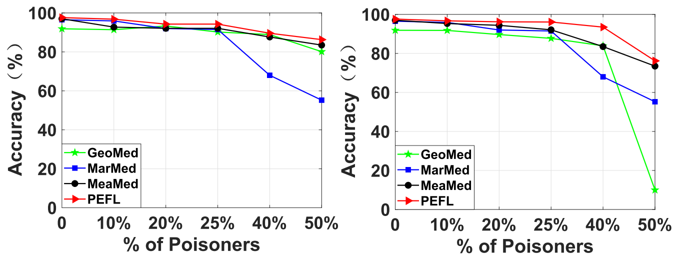


Fig. 10. Attack success rate with different source and target class.



(a) T_{hdr} with the label-flipping attack (b) T_{hdr} with the backdoor attack

Fig. 11. Comparison of accuracy with different number of poisoners.

in defense ability and classification accuracy under the label-flipping attack and backdoor attack.

3) Comparison With Median-Based Schemes: The effectiveness of the PEFL mainly depends on the reliability of the coordinate-wise medians. We thus provide a comparison with three median-based schemes, i.e., *GeoMed*, *MarMed*, *MeaMed*. By definition in [37], *GeoMed* means the geographic median of all gradient vectors, *MarMed* takes the coordinate-wise medians as the update, and *MeaMed* averages the top-$(m - f)$ values nearest to the coordinate-wise medians, where f is malicious users in the subset of users. Fig. 11 depicts the accuracy of T_{hdr} of four schemes with different proportions of poisoners under two attack methods. Since T_{ir} has similar results, we only use T_{hdr} to show the performance advantages of the PEFL here. We analyze that the reason for the low accuracy of *GeoMed* and *MarMed* is mainly each iteration only takes one user' gradient as the update. Similar to *Krum*, it ignores the contribution of the remaining $(m - f - 1)$ honest users. When up to 50% of poisoners in the protocol, *GeoMed* does not converge. Compared with *GeoMed* and *MarMed*, *MeaMed* has higher prediction accuracy under both attacks. However, the design of *MeaMed* requires the number of poisoners as prior knowledge. As discussed earlier, this is an unrealistic assumption.

Compared with three median-based schemes, the PEFL has the best accuracy advantage under both attacks and does not need to know the number of poisoners in advance. In summary, the PEFL outperforms the above median-based schemes.

VII. Related Works

A. Robust Aggregation Method

In response to the poisoning attacks in federated learning, the machine learning community has proposed a few defense schemes [6], [20]. In general, existing solutions mainly involve the following three methods. From the data level, statistics-based aggregation allows the parameter server to aggregate parameters based on statistics extracted from training data or gradients, rather than averaging them. Representative examples include *Krum* [6] based on Euclidean distance, and *GeoMed* [21] based on geographic median. These methods, however, are at considerable cost to the training accuracy and efficiency. From the model level, the parameter server assigns the updated sub-model to the data owners to determine whether there is an abnormality [20], [36]. Obviously, this method inevitably incurs redundant communication overhead. Worse of all, it can only tolerate a limited number of poisoners. Recently, some researchers have proposed to disconnect the end-to-end mapping from the specific feature space to the target label space from the neuron level [38], [39]. For example, Wang *et al.* [39] proposed the *Neural Cleanse*, which identifies the existence of trojaned behaviors by discovering small input disturbances that continuously change the output of the model, and then prunes the neurons via the model patching algorithm. However, this approach requires experts in neural networks and vast quantities of clean samples.

Due to relevance to this paper, we present more statistics-based defenses here, whose key idea is to focus the aggregation rules on the benign gradient vectors as much as possible. For example, *Blanchard* designed *Krum* [6], which selects one of the gradients most similar to the other gradients as the global update. The similarity here is measured by calculating the sum of the Euclidean distance of $(m - f - 2)$ gradients closest to the gradient vector. The gradient with the smallest sum is selected by *Krum*. *Bulyan* [24] is essentially a variant that combines *Krum* and *Trimmed mean*. *Bulyan* first chooses less than $(m - 2f)$ gradients with the same rule as the *Krum*, then averages the parameters closest to the median of the gradient vectors as the global update. However, Fang *et al.* [40] suggested that both of the above aggregation rules are not effective enough for an adversary with a certain amount of knowledge, who can carefully design a set of similar gradients to befuddle the aggregation rules. *Trimmed mean* [25] removes some extreme values, and then averages the remaining parameters as the global update. *GeoMed* [21] takes the median of the gradients as global update. However, their computational complexity exponentially increases with the dimension of gradients. Besides, Fang *et al.* [40] proposed a scheme to pull the above aggregation rules into the trap by forging the maximum and minimum values of the gradients based on the mean and variance of the user' parameters.

However, the aforementioned statistics-based defenses impair the model accuracy. In this paper, we presented the PEFL with satisfactory accuracy, which mainly benefits from the coordinate-wise medians.

B. Privacy-Preserving Federated Learning

Recently, enabling privacy-preserving FL mainly bases on the following three underlying technologies: Differential Privacy (DP) [12], Secure Multi-Party Computation (SMC) [14], Homomorphic Encryption (HE) [18], and Trusted Execution Environment (TEE) [41], [42]. However, each of the above methods has its limitations, which remains the implementation of privacy-preserving FL being an open problem.

Specifically, Shokri and Shmatikov [12] first proposed a privacy-preserving FL framework, which is implemented by selectively sharing small subsets of parameters of the model and perturbing them by exploiting the DP mechanism. However, this scheme has to make a trade-off between accuracy and privacy. To further improve the utility of the model, Abadi *et al.* [10] designed the Moments Accountant to keep track of a bound on the moments of the privacy loss during the training process with DP. However, Jayaraman and Evans [19] showed that current DP based works are rarely to offer acceptable utility-privacy trade-offs. *Payman* [26] and Bell *et al.* [14] protected the privacy of training data by using secret sharing, which is representative technology to implement SMC. However, it requires users to stay online, which introduces a new performance bottleneck for the actual distribution scenarios. Recently, Tramer and Boneh [42] proposed a private execution of neural network in TEE, which isolates the sensitive computations from untrusted software by using specialized hardware. Unfortunately, this approach is difficult to extend on a large scale due to its expensive cost in hardware and scarcity of scalability.

The above schemes are only limited to privacy issues in FL. In this paper, we build a feasible bridge between privacy protection and defense against poisoning attacks, which provides new idea for the design of future FL frameworks.

VIII. CONCLUSION

In this paper, we have proposed a novel framework called PEFL, which can provide privacy-preserving federated learning while guaranteeing robustness against representative data poisoning attacks. The PEFL is proved secure and convergent with the comprehensive theoretical analysis. Moreover, the experimental results demonstrated that the comparable performance of PEFL in terms of accuracy and robustness. In future work, we will focus on improving the accuracy of the PEFL and further exploit ways to optimize the efficiency.

REFERENCES

[1] P. Kairouz *et al.*, "Advances and open problems in federated learning," 2019, *arXiv:1912.04977*. [Online]. Available: http://arxiv.org/abs/1912.04977

[2] S. Niknam, H. S. Dhillon, and J. H. Reed, "Federated learning for wireless communications: Motivation, opportunities, and challenges," *IEEE Commun. Mag.*, vol. 58, no. 6, pp. 46–51, Jun. 2020.

[3] W. Zhu, M. Baust, Y. Cheng, S. Ourselin, M. J. Cardoso, and A. Feng, "Privacy-preserving federated brain tumour segmentation," in *Machine Learning in Medical Imaging*, vol. 11861. Springer, 2019, pp. 133–141.

[4] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the GAN: Information leakage from collaborative deep learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 603–618.

[5] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients—How easy is it to break privacy in federated learning?" 2020, *arXiv:2003.14053*. [Online]. Available: http://arxiv.org/abs/2003.14053

[6] P. Blanchard *et al.*, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Proc. NeurIPS*, 2017, pp. 119–129.

[7] 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 *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2019, pp. 739–753.

[8] W. Jiang, H. Li, S. Liu, X. Luo, and R. Lu, "Poisoning and evasion attacks against deep learning algorithms in autonomous vehicles," *IEEE Trans. Veh. Technol.*, vol. 69, no. 4, pp. 4439–4449, Apr. 2020.

[9] X. Liu, H. Li, G. Xu, S. Liu, Z. Liu, and R. Lu, "PADL: Privacy-aware and asynchronous deep learning for IoT applications," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 6955–6969, Aug. 2020.

[10] M. Abadi *et al.*, "Deep learning with differential privacy," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2016, pp. 308–318.

[11] X. Liu, H. Li, G. Xu, R. Lu, and M. He, "Adaptive privacy-preserving federated learning," *Peer Peer Netw. Appl.*, vol. 13, no. 6, pp. 2356–2366, Nov. 2020.

[12] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proc. 53rd Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sep. 2015, pp. 1310–1321.

[13] K. Bonawitz *et al.*, "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 1175–1191.

[14] J. H. Bell, K. A. Bonawitz, A. Gascón, T. Lepoint, and M. Raykova, "Secure single-server aggregation with (Poly)Logarithmic overhead," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2020, pp. 1253–1269.

[15] Y. Li, H. Li, G. Xu, T. Xiang, X. Huang, and R. Lu, "Toward secure and privacy-preserving distributed deep learning in fog-cloud computing," *IEEE Internet Things J.*, vol. 7, no. 12, pp. 11460–11472, Dec. 2020.

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

[17] G. Xu, H. Li, Y. Zhang, S. Xu, J. Ning, and R. Deng, "Privacy-preserving federated deep learning with irregular users," *IEEE Trans. Depend. Sec. Comput.*, early access, Jun. 30, 2020, doi: 10.1109/TDSC.2020.3005909.

[18] Q. Zhang, C. Xin, and H. Wu, "GALA: Greedy ComputAtion for linear algebra in privacy-preserved neural networks," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2021, pp. 1–16. [Online]. Available: https://www.ndss-symposium.org/ndss-paper/gala-greedy-computation-for-l%inear-algebra-in-privacy-preserved-neural-networks/

[19] B. Jayaraman and D. Evans, "Evaluating differentially private machine learning in practice," in *Proc. USENIX Secur.*, 2019, pp. 1895–1912.

[20] S. Rajput, H. Wang, Z. Charles, and D. Papailiopoulos, "DETOX: A redundancy-based framework for faster and more robust gradient aggregation," in *Proc. NeurIPS*, 2019, pp. 10320–10330.

[21] Y. Chen, L. Su, and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 1, no. 2, pp. 1–25, 2017.

[22] J. So, B. Guler, and A. S. Avestimehr, "Byzantine-resilient secure federated learning," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 2168–2181, Jul. 2021, doi: 10.1109/JSAC.2020.3041404.

[23] G. Baruch, M. Baruch, and Y. Goldberg, "A little is enough: Circumventing defenses for distributed learning," in *Proc. NeurIPS*, 2019, pp. 8632–8642.

[24] R. Guerraoui *et al.*, "The hidden vulnerability of distributed learning in byzantium," in *Proc. ICML*, 2018, pp. 3521–3530.

[25] D. Yin, Y. Chen, K. Ramchandran, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *Proc. ICML*, 2018, pp. 5650–5659.

[26] P. Mohassel and Y. Zhang, "SecureML: A system for scalable privacy-preserving machine learning," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 19–38.

[27] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, "VerifyNet: Secure and verifiable federated learning," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 911–926, 2020.

[28] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," 2012, *arXiv:1206.6389*. [Online]. Available: http://arxiv.org/abs/1206.6389

[29] E. Bagdasaryan *et al.*, "How to backdoor federated learning," in *Proc. PMLR AISTATS*, 2020, pp. 2938–2948.

[30] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.*, 1999, pp. 223–238.

[31] J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," in *Noise Reduction in Speech Processing*. Springer, 2009, pp. 1–4.

[32] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural Netw.*, vol. 12, no. 1, pp. 145–151, Jan. 1999.

[33] E.-M. El-Mhamdi, R. Guerraoui, and S. Rouault, "Distributed momentum for byzantine-resilient learning," 2020, *arXiv:2003.00010*. [Online]. Available: http://arxiv.org/abs/2003.00010

[34] H. Chen, W. Dai, M. Kim, and Y. Song, "Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2019, pp. 395–412.

[35] M. D. Zeiler, "ADADELTA: An adaptive learning rate method," 2012, *arXiv:1212.5701*. [Online]. Available: http://arxiv.org/abs/1212.5701

[36] L. Zhao *et al.*, "Shielding collaborative learning: Mitigating poisoning attacks through client-side detection," *IEEE Trans. Depend. Sec. Comput.*, early access, Apr. 14, 2020, doi: 10.1109/TDSC.2020.2986205.

[37] C. Xie, O. Koyejo, and I. Gupta, "Generalized Byzantine-tolerant SGD," 2018, *arXiv:1802.10116*. [Online]. Available: http://arxiv.org/abs/1802.10116

[38] Y. Liu, W.-C. Lee, G. Tao, S. Ma, Y. Aafer, and X. Zhang, "ABS: Scanning neural networks for back-doors by artificial brain stimulation," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2019, pp. 1265–1282.

[39] B. Wang *et al.*, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2019, pp. 707–723.

[40] M. Fang, X. Cao, J. Jia, and N. Z. Gong, "Local model poisoning attacks to byzantine-robust federated learning," in *Proc. USENIX Secur.*, 2020, pp. 1605–1622.

[41] X. Zhang, F. Li, Z. Zhang, Q. Li, C. Wang, and J. Wu, "Enabling execution assurance of federated learning at untrusted participants," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Jul. 2020, pp. 1877–1886.

[42] F. Tramer and D. Boneh, "Slalom: Fast, verifiable and private execution of neural networks in trusted hardware," in *Proc. ICLR*, 2018, pp. 1–19. [Online]. Available: https://openreview.net/forum?id=rJVorjCcKQ

Xiaoyuan Liu (Graduate Student Member, IEEE) received the B.S. degree from the School of Control and Computer Engineering, North China Electric Power University, in 2016. She is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, China. Her research interests include applied cryptography and privacy issues in machine learning.

Hongwei Li (Senior Member, IEEE) received the Ph.D. degree from the University of Electronic Science and Technology of China in June 2008. He is currently the Head and a Professor with the Department of Information Security, School of Computer Science and Engineering, University of Electronic Science and Technology of China. He worked as a Post-Doctoral Fellow with the University of Waterloo from October 2011 to October 2012. His research interests include network security and applied cryptography. He is also a Distinguished Lecturer of the IEEE Vehicular Technology Society.

Guowen Xu (Member, IEEE) received the Ph.D. degree from the University of Electronic Science and Technology of China in 2020. He is currently a Research Fellow with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. As the first author, he has published more than 15 papers in reputable venues, including ACM ACSAC, ACM ASIACCS, IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, and IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING. His research interests include applied cryptography, and AI security and privacy issues. He was a recipient of the Best Paper Award of ICPADS 2020 and the INFOCOM 2021 Student Conference Award.

Zongqi Chen (Graduate Student Member, IEEE) received the B.S. degree in information security from Hunan University, Hunan, China, in 2019. He is currently pursuing the master's degree with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China. His research interests include cryptography and AI security.

Xiaoming Huang (Member, IEEE) is currently the General Manager of the Technology Marketing Department, CETC Cyberspace Security Research Institute Company Ltd. His research interests include cognitive domain security, cryptography and information security theory, trusted computing and trusted network technology, and computer and communication security issues. He is a member of Sichuan Electronic Information Expert Group.

Rongxing Lu (Fellow, IEEE) received the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Waterloo, Canada, in 2012. He is currently an Associate Professor with the Faculty of Computer Science (FCS), University of New Brunswick (UNB), Canada. He was awarded the most prestigious Governor General's Gold Medal in 2012 and won the Eighth IEEE Communications Society (ComSoc) Asia Pacific (AP) Outstanding Young Researcher Award in 2013. He was the Winner of the 2016–2017 Excellence in Teaching Award, FCS, UNB. He also serves as the Vice-Chair (Publication) of IEEE ComSoc CIS-TC.