SPEFL: Efficient Security and Privacy Enhanced Federated Learning Against Poisoning Attacks

Liyan Shen, Zhenhan Ke, Jinqiao Shi, Xi Zhang, Yanwei Sun, Jiapeng Zhao, Xuebin Wang, Xiaojie Zhao

***Abstract*—Federated learning (FL) is a distributed machine learning paradigm in the Internet of Things (IoT), which allows multiple devices to collaboratively train models without leaking local data. In the open scenario of IoT, malicious devices can launch poisoning attacks to compromise the final model by submitting crafted gradients. Some previous studies defend against poisoning attacks by analyzing the statistical characteristics of plaintext gradients. However, plaintext gradients would expose private information to malicious FL devices or servers. To simultaneously resist poisoning attacks and preserve privacy, cryptography technology can be utilized to obfuscate the gradients in defense methods, but the private calculation of resisting poisoning attack methods will cause efficiency problems, especially imposing unaffordable overhead on resource-limited IoT devices. Therefore, resisting poisoning attacks efficiently while protecting privacy remains a challenge. This paper proposes a secure and privacy-enhanced FL (SPEFL) framework for efficient privacy-preserving and poisoning-resistant federated learning in IoT. We design an efficient secure computation protocol based on a three-server architecture to facilitate the cryptographic computation of large linear and complex nonlinear operators in the method against poisoning attacks. In SPEFL, most of the calculations are efficiently performed on the servers, which will not impose too much burden on resource-limited IoT devices. In addition, we design a security-enhanced verifiable protocol to detect the malicious behavior of the server and guarantee the correctness of FL aggregation results. Experimental and theoretical results demonstrate that SPEFL can efficiently complete FL training meanwhile guaranteeing the accuracy of the model.**

***Index Terms*—Federated learning, Poisoning attack, Privacy-preserving, Internet of Things, Secure multi-party computation.**

I. Introduction

FEDERATED learning (FL) is a distributed machine learning paradigm that enables multiple Internet of Things (IoT) devices to collaboratively learn a global model without collecting raw data. For example, multiple mobile devices can train local models using local text data and upload updates to jointly improve text entry [1]. With the development of IoT systems, the number of devices and the volume of data are increasing rapidly [2]. In traditional machine learning, institutions need to collect large volumes of data and train models centrally, which is not only difficult to achieve but also violates privacy. Different from traditional machine learning, the data of the devices in FL is stored locally, and the devices only need to upload the local update of each iteration. Therefore, federated learning (FL) can be used to coordinate multiple devices in IoT and avoid collecting private data [3]–[5].

Due to the open and distributed nature of IoT, the application of FL will suffer from security and privacy threats [6], [7]. One of the typical security threats is poisoning attacks, where malicious devices among the large number of devices in the open IoT may corrupt training by intentionally modifying training data or uploading elaborate poisoning gradients [8]–[12]. Researches have shown that a single poisoning attacker can significantly modify training results and reduce model performance [8], [13]. Another concern is that the aggregation server could infer private information from gradients trained on device-local data [14]–[19], which violates strict regulations, such as General Data Protection Regulation (GDPR).

In order to prevent maliciously poisoned gradients from affecting training, several defense methods are proposed [8], [9], [20]–[25]. The main idea is to filter out outliers by analyzing statistics of gradients such as Euclidean distance [8], trimmed mean [9], median [9], etc. Additionally, some defense mechanisms compute trust scores and aggregate weights for gradients to take full advantage of benign gradients [21], [24]. However, most defense methods require the server to observe plaintext gradients, which can lead to the leakage of private data. To deal with privacy concerns, a variety of privacy-preserving schemes have been proposed in FL based on cryptographic protocols [26]–[29], such as Homomorphic encryption (HE) and Secure Multi-Party Computation (MPC). MPC enables several participants to collaboratively compute arbitrary functions and has been used in secure aggregation in FL [29], [30], where the server aggregates the global gradients without knowing the original gradients. The above studies address poisoning attacks and privacy issues separately, however, both of these issues affect the application of FL in IoT and need to be addressed simultaneously.

To simultaneously address poisoning attacks and privacy protection in FL, an intuitive approach is to directly integrate cryptographic techniques with defense methods to resist poisoning attacks without revealing plaintext gradients. However, defense methods that involve a large scale of linear (matrix multiplication) and complex nonlinear (comparison, minimum, etc.) operators, such as Krum [8], Mean [9] and Bulyan [20],

This work was supported partially by the National Natural Science Foundation of China under Grant 62302057 and the CAAI-Huawei MindSpore Open Fund. Compute services from Hebei Artificial Intelligence Computing Center.

Liyan Shen, Zhenhan Ke, Jinqiao Shi (Corresponding Author), Xi Zhang, Yanwei Sun, Jiapeng Zhao, and Xiaojie Zhao are with the Key Laboratory of Trustworthy Distributed Computing and Service (BUPT), Ministry of Education, Beijing University of Posts and Telecommunications, Beijing, 100088, China (e-mail: shijinqiao@bupt.edu.cn).

Xuebin Wang is with the Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China.

will incur a huge overhead due to the computation of heavy cryptographic operations. In order to improve efficiency, some work makes trade-offs in terms of privacy by revealing some intermediate values [25], [31] or in terms of effectiveness against poisoning attacks by utilizing relatively simple defenses [32]. However, in the IoT scenario, the computation of the server needs to be efficient enough to support large-scale IoT devices. And since the computation and bandwidth resources of IoT devices are particularly limited, the defense methods of poisoning attacks and privacy computation technologies should not bring too much burden to the devices.

Consequently, how to effectively resist FL poisoning attacks while sufficiently and efficiently protecting privacy in IoT remains a challenge. To address the above issues, this paper proposes an efficient security and privacy-enhanced FL framework SPEFL. In SPEFL, we perform secret sharing in a three-server architecture, which enables poisoning attack detection without exposing plaintext gradients. We design a series of computation protocols in which the overhead of the device is light and the computation of the servers can be performed efficiently. Furthermore, we provide a security-enhanced verifiable protocol to detect the malicious behavior of the server, which can guarantee the correctness of the aggregation results. The key contributions of our work are summarized as follows:

- We propose a private, lightweight and robust FL framework SPEFL, which facilitates the cryptographic computation of large linear and complex nonlinear operators in the method against poisoning attacks by introducing a three-server architecture.
- We complete the effective defense method [25] based on median and Pearson correlation coefficient on the ciphertext. We present a detailed security analysis, and experiments on three neural network models on real datasets demonstrate that SPEFL is two orders of magnitude faster than the HE-based method and an order of magnitude faster than the method based on general MPC peotocol while maintaining the accuracy of the model.
- We further design a verifiable FL protocol to guarantee the correctness of the aggregated results, when the server is corrupted by malicious adversaries. It is secure even if one server colludes with multiple devices in the malicious model.

The paper is organized as follows. In Section II, we introduce the related work. Next, we present the preliminaries, threat model and the SPEFL framework in Section III and perform the theoretical analysis in Section IV. In Section V, we evaluate the performance of SPEFL. After, we propose a security-enhanced protocol in Section VI. Finally, we summarize our work in Section VII.

II. Related Work

Recently, FL has received extensive attention in the IoT domain, but FL also faces security and privacy issues [6], [33]. In this section, we outline the related solutions to poisoning attacks and solutions that address these attacks and privacy-preserving issues simultaneously in FL.

Defenses against poisoning attacks: In federated learning, malicious devices may break training results by modifying data, constructing crafted gradients, etc. Even a single malicious device can significantly affect the training process and corrupt the training results [8]. Researchers have proposed many solutions to resist poisoning attacks, one of which is to analyze statistical characteristics of gradients to identify abnormal behaviors and remove suspicious outliers. For example, Krum [8] finds outliers based on the Euclidean distance between gradients. To further improve the robustness, Bulyan [20] combines Euclidean distance and trimmed mean to determine the similarity between user gradients and Yin et al. [9] proposed coordinate-wise trimmed mean and median schemes. The above works aggregate only a few representative gradients into the global model. To allow benign gradients to participate more fully in aggregation, some works compute trust scores for user gradients. In AFA [21], the author proposed a Hidden Markov Model to evaluate the quality of user gradients, and Cao et al. [24] proposed FLTrust, which assigns different weights to different gradients through the cosine similarity between the gradient and the benchmark. Mao et al. proposed Romoa [34], which combines multiple criteria such as cosine similarity and linear correlation to identify poisoning attacks. Zhang et al. proposed FLDetector [35], which uses L-BFGS to predict gradient updates and judge malicious clients based on Euclidean distance. However, these defense methods are designed and executed based on plaintext without considering privacy issues.

Methods against poisoning attacks and privacy preserving: To simultaneously address the security and privacy concerns of federated learning in IoT, Domingo-Ferrer et al. [7] proposed an FL framework that can provide unlinkable anonymity and identify malicious peers. However, the framework still submits plaintext gradients and does not solve the data leakage problem. In [19], the authors proposed robust aggregation in FL based on differential privacy, but the scheme based on differential privacy needs to sacrifice model accuracy in exchange for privacy. Besides that, there have been several works that can resist poisoning attacks and protect privacy while maintaining accuracy by obfuscating gradients and analyzing the statistics of these obfuscated gradients. From the perspective of cryptographic techniques, most of these works can be divided into two categories: based on homomorphic encryption and secret sharing.

The schemes based on homomorphic encryption: Liu et al. [25] proposed PEFL based on a new defense method and linearly homomorphic encryption. The defense method adopted the Pearson correlation coefficient and the coordinate-wise medians as the benchmark. It has a better effect than traditional schemes such as Krum and Bulyan in resisting poisoning attacks. However, in order to efficiently calculate the Pearson correlation coefficient, the authors involved multiplication operations in obscuring the gradient vectors, which leak users' private information. Concretely, the server can obtain the obfuscated gradient for calculation, but we found that due to the insecure obfuscation, user privacy can still be inferred from the obfuscated gradient using Deep Leakage from Gradients (DLG) [14]. We have conducted experiments

to demonstrate this problem in Section V. Ma et al. [36] also proposed ShieldFL based on HE, which calculates the cosine similarity and detects the poisoning gradient on the ciphertext. Hidde et al. proposed RoFL [37], which combines cryptographic techniques such as HE, zero-knowledge argument of knowledge, and commitments to achieve the robustness of secure federated learning by enforcing constraints such as norm bounds on clients' updates. However, these schemes used homomorphic encryption to complete aggregation and poisoning attack detection, the frequent homomorphic encryption operations will reduce the efficiency of the servers. Besides, HE technology is computationally expensive and adds high overhead to edge devices with limited resources, which makes the FL frameworks unsuitable for IoT scenarios.

The schemes based on secret sharing: So et al. [32] proposed BREA which utilized Shamir's secret sharing to protect privacy and Krum to resist poisoning attacks. In BREA, the high computational complexity of aggregation rules aggravates the overhead problem on the ciphertext computation, and FL users need to perform complex calculations such as polynomial reconstruction, which makes the framework execution inefficient. Several works proposed utilizing two servers to increase performance and reduce overhead for users. Concretely, the users sent shares to the two non-colluding servers using additive secret sharing, and they cooperated to complete the aggregation operation. Khazrak et al. [38] proposed a method to mitigate poisoning attacks in privacy-preserving distributed collaborative learning by employing secret sharing. However, this method has high computational complexity and assumes that each user will normalize the parameter vector, which may not be obeyed by malicious users. Hao et al. [39] proposed SecureFL which adopted the robust aggregation scheme of [24] and additive secret sharing. For the nonlinear comparison operations, it adopted complex garbled circuits, resulting in a relatively large overhead. Besides, it required the central server to have a clean seed dataset for detecting malicious users, which limits the application of the framework.

In IoT scenarios, resource-limited devices have higher requirements for protocol performance. It is not suitable to directly apply general privacy computation operators to the defense methods in federated learning [29], [30], [40]. To solve this problem, this paper proposes a robust and private FL framework SPEFL based on the three-server architecture in IoT, which can be implemented using trusted hardware [41] or trusted execution environments such as Intel SGX [42] and can be used in IoT-based healthcare scenarios. Specifically, inspired by existing work that introducing an additional assistant computing server can significantly reduce computation and communication overhead [25], [43], [44], we introduce an additional assistant computing server based on the secure two-party computation and form the three-server architecture. Further, we design a series of customized computation protocols to improve the performance of private and robust FL under the three-server architecture. In our framework, three servers will bear the main overhead. By introducing an Assistant Computing Server, complex privacy computation operators can be executed efficiently in IoT scenarios.

SPEFL requires no prior knowledge and is efficient on both the device side and the server side. A coarse-grained comparison between different methods is shown in Table I, and we will present the implementation of SPEFL in Section III.

III. Framework of SPEFL

In this section, we introduce the preliminaries, threat model, the overview of the SPEFL framework, and illustrate the specific protocol constructions.

A. Preliminaries

1) Federated Learning: FL is a distributed machine learning framework that enables various parties to collaboratively train a global model w using local data. In this paper, we focus on supervised horizontal federated learning. Suppose there are n devices/clients jointly training a global model w, and each device P_i has a private dataset $D_i, i \in [n]$. In each iteration, P_i locally trains the model to obtain the gradient $\boldsymbol{G}_i$ and sends $\boldsymbol{G}_i$ to the central server. Finally, the server aggregates the gradients into a global gradient and updates the global model as follows:

$$\boldsymbol{G}^{\text{global}} = \text{agg}\left(\boldsymbol{G}_1, \boldsymbol{G}_2 \ldots, \boldsymbol{G}_n\right) \tag{1}$$

$$w = w - \eta \cdot \boldsymbol{G}^{\text{global}} \tag{2}$$

where $\text{agg}(\cdot)$ is the aggregation function and η is the learning rate. In order to prevent malicious devices from affecting training, a robust aggregation function $\text{agg}(\cdot)$ that can defend against poisoning attacks is required.

2) Cryptographic Primitives: SPEFL mainly uses *additive secret sharing* to achieve privacy protection, as in many traditional privacy-preserving machine learning works [43], [45], [46]. In the two-party secure computation, the shares of a secret x can be denoted as $\langle x \rangle := (\langle x \rangle_0, \langle x \rangle_1)$, where $x = \langle x \rangle_0 + \langle x \rangle_1$. The parties generate two shares of x in the ring $\mathbb{Z}_{2^L}$, where $\langle x \rangle_0 = r$, $\langle x \rangle_1 = x - r \pmod{\mathbb{Z}_{2^L}}$ and $r \in \mathbb{Z}_{2^L}$ is a random number. They can reconstruct the value of x by computing $\langle x \rangle_0 + \langle x \rangle_1$. Participants can further perform $\langle x + y \rangle$, $\langle x \cdot y \rangle$ on the shares. To compute $\langle z \rangle = \langle x + y \rangle$, they can set $\langle z \rangle_i = \langle x \rangle_i + \langle y \rangle_i$. And the calculation of $\langle z \rangle = \langle x \cdot y \rangle$ requires Beaver's multiplication triplets, the details are shown in Algorithm 1. The above algorithms can be used to compute matrix multiplication by sharing the elements of matrix X component-wise and expanding the triples to matrix format [44].

Algorithm 1 Multiplication Protocol Based on Secret Sharing.

Input: Party P_0 has shares $\langle x \rangle_0, \langle y \rangle_0$ and triples $\langle a \rangle_0$, $\langle b \rangle_0$, $\langle c \rangle_0$; P_1 has shares $\langle x \rangle_1, \langle y \rangle_1$ and triples $\langle a \rangle_1$, $\langle b \rangle_1$, $\langle c \rangle_1$, where $c = a \cdot b$.

Output: P_0 gets share $\langle x \cdot y \rangle_0$ and P_1 gets share $\langle x \cdot y \rangle_1$.

1: P_i computes $\langle e \rangle_i = \langle x \rangle_i - \langle a \rangle_i$ and $\langle f \rangle_i = \langle y \rangle_i - \langle b \rangle_i$, $i = 0, 1$.
2: P_i sends $\langle e \rangle_i$, $\langle f \rangle_i$ to each other and reconstruct e, f.
3: For $i = 0, 1$, P_i outputs $\langle x \cdot y \rangle_i = f \cdot \langle x \rangle_i + e \cdot \langle y \rangle_i + \langle c \rangle_i - i \cdot e \cdot f$.

TABLE I
COARSE-GRAINED COMPARISON BETWEEN SCHEMES

FL Scheme	Proportion of Poisoners	Computational Complexity	Privacy Protection	No prior knowledge required	Light burden on devices	High efficiency of the server(s)
Krum [8]	<50%	Medium	✗	✓	✓	✓
Bulyan [20]	<25%	Medium	✗	✓	✓	✓
FLTrust [24]	<50%	Low	✗	✗	✓	✓
PEFL [25]	<50%	Low	✓	✓	✗	✗
SecureFL [39]	<50%	Low	✓	✗	✓	✗
SPEFL	<50%	Low	✓	✓	✓	✓

B. Threat Model

SPEFL involves three servers and several IoT devices. This paper mainly focuses on privacy data leakage to the servers and the destruction of FL models by malicious devices. We consider three semi-honest servers and a part of malicious devices outside of honest devices. We assume that all three servers are semi-honest and do not collude with each other, nor with other IoT devices, following previous works [25], [43], [46]. Honest and malicious devices have independent and identically distributed local datasets. Honest devices can only access their own local data sets, while malicious devices have the same attack target and can collude with each other to launch attacks. We assume that malicious devices can arbitrarily control their local datasets and gradients uploaded to servers to launch poisoning attacks. We follow the settings of existing works to limit the number of malicious devices [8], [20], [25], specifically, $t < \frac{n-1}{2}$, where t and n are the number of malicious devices and the total devices.

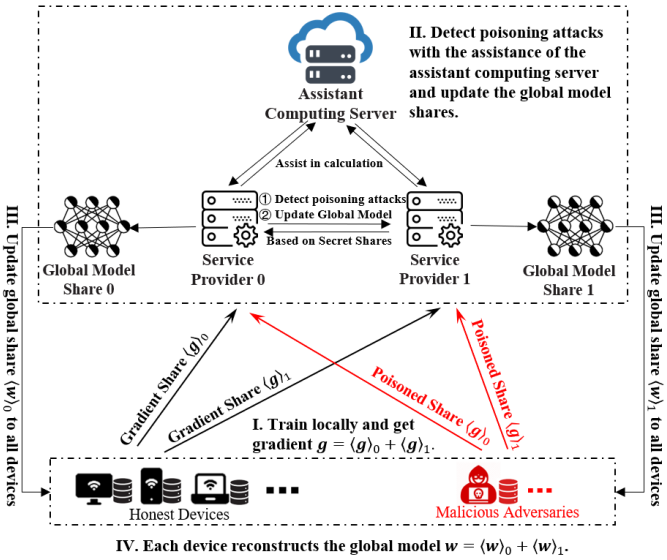


Fig. 1. System Model of SPEFL

C. Overview

In SPEFL, there are n devices/clients P_i holding private local data and three servers. Two of the three servers are Service Providers $\mathrm{SP}_0, \mathrm{SP}_1$, which receive secret shares of gradients, and perform poisoning attack detection and secure gradients aggregation on the secret shares. The third server is an Assistant Computing Server CS, which assists SP_0 and SP_1 to perform related calculations.

The system model of SPEFL is shown in Fig. 1. It can be divided into preprocessing, local training, and aggregating phase. The preprocessing phase is performed only once in the entire protocol. The CS generates Beaver's multiplication triplets and sends them to the two service providers. SP_x randomly initializes the secret shares of global model $\langle \boldsymbol{w}_{init}\rangle_x, x = 0, 1$, and sends it to all devices. The devices reconstruct and set the global model $\boldsymbol{w}_{init} = \langle \boldsymbol{w}_{init}\rangle_0 + \langle \boldsymbol{w}_{init}\rangle_1$. In the local training phase, each device P_i trains the model locally and computes the gradient $\boldsymbol{G}_i$ using stochastic gradient descent (SGD) on the local dataset D_i. Then, P_i sends the secret shares $\langle \boldsymbol{G}_i\rangle_0$ and $\langle \boldsymbol{G}_i\rangle_1$ to SP_0 and SP_1 respectively. Finally, in the aggregating phase, SP_0 and SP_1 detect poisoning attacks, aggregate gradients, and update parameters under secret sharing with the assistance of CS. And in each iteration, SP_0 and SP_1 broadcast the calculated secret shares of the global model to all devices.

To prevent malicious devices from affecting the training process, SPEFL uses a robust aggregation protocol based on the median and Pearson correlation coefficients from [25]. The main idea is to use the median $\boldsymbol{G}_{med}$ of device gradients as the benchmark and determine its weight μ according to the Pearson correlation coefficient ρ between the gradient and the benchmark. Therefore, SPEFL mainly includes three protocols, namely the secure median calculation protocol $SecMed$, the secure correlation coefficient calculation protocol $SecCorr$, and the secure gradients aggregation protocol $SecAgg$, which will be introduced in detail in Section III-D.

In our threat model, benign devices are the majority, so the median gradient is closer to the gradients of benign devices. Therefore, the benign gradient will have a stronger correlation with the median benchmark, while the correlation coefficient ρ between the poisoned gradient and the median benchmark will be lower. Further, we use a logarithmic function to calculate the weight as $\max\left\{0, \ln\left(\frac{1+\rho}{1-\rho}\right) - 0.5\right\}$. This function encourages a higher divergence for values that are near the two tails of the function and when the correlation coefficient is smaller than a certain value, the aggregation weight is clipped to 0. So the poisoned gradients sent by malicious devices will correspond to lower correlation coefficients, and their weights will be close to or equal to zero.

The defense against poisoning attack capability of SPEFL comes directly from the aggregation protocol under "ciphertext". Secret sharing ensures that each server will not get the

client's private plaintext gradient, median gradient, and other intermediate information that can infer the client's privacy. And under the three-server architecture, the detection and aggregation functions can be completed safely and efficiently.

D. Constructions of SPEFL

Since the preprocessing and local training phases are relatively simple and have been clearly described above, we focus on describing the key aggregating phase in SPEFL. Besides, SPEFL transforms the floating-point operations into integer operations through fixed-point representation on the field $\mathbb{Z}_{2^L}$. Like the previous work, our method completes the conversion within a recognized preset parameter [39], [45], and the devices need to encode the gradient vector as integers as follows:

$$\boldsymbol{G_i} = \{\lfloor prec \cdot G_{ij} \rfloor\}_{j=1}^{m}, i \in [1, n] \tag{3}$$

where m represents the dimension of the gradient vector and $prec$ represents the preset conversion precision. For convenience, $\boldsymbol{G}_i$ is used to denote the integer gradient vector, and G_{ij} represents the jth dimension of the vector $\boldsymbol{G}_i$.

Gradients Median Calculating. First, SP_0 and SP_1 compute the sharing of the median gradient $\boldsymbol{G}_{med}$, which is the benchmark for judging whether gradients are malicious. We design the security protocol $SecMed$ for this process, and the details are given in Algorithm 2. The specific steps are as follows: for $x = 0, 1$, SP_x randomly generates a vector $\langle \boldsymbol{r} \rangle_x$, then for $i \in [1, n]$, SP_x computes $\langle \boldsymbol{R}_i \rangle_x = \langle \boldsymbol{G}_i \rangle_x + \langle \boldsymbol{r} \rangle_x$, and sends $\langle \boldsymbol{R}_i \rangle_x$ to CS. Then CS reconstructs $\boldsymbol{R}_i = \langle \boldsymbol{R}_i \rangle_0 + \langle \boldsymbol{R}_i \rangle_1$. Note that, $\boldsymbol{R}_i = \boldsymbol{G}_i + \boldsymbol{r}$, where $\boldsymbol{r} = \langle \boldsymbol{r} \rangle_0 + \langle \boldsymbol{r} \rangle_1$. Then CS calculates the median of each dimension for $\boldsymbol{R}_i$, i.e., for m-dimensional gradient vectors, m medians need to be calculated. The vector composed of the confused medians is denoted as $\boldsymbol{R}_{med} = \boldsymbol{G}_{med} + \boldsymbol{r}$, and CS sends the secret shares $\langle \boldsymbol{R}_{med} \rangle := (\langle \boldsymbol{R}_{med} \rangle_0, \langle \boldsymbol{R}_{med} \rangle_1)$ to SP_0, SP_1 respectively. After, SP_x sets $\langle \boldsymbol{G}_{med} \rangle_x = \langle \boldsymbol{R}_{med} \rangle_x - \langle \boldsymbol{r} \rangle_x$, thus obtaining the secret shares of the median gradients $\langle \boldsymbol{G}_{med} \rangle$.

Correlation coefficient Calculating. After SP_0 and SP_1 have obtained the secret shares of $\boldsymbol{G}_i$ and $\boldsymbol{G}_{med}$, the Pearson correlation coefficient ρ_i between $\boldsymbol{G}_i$ and $\boldsymbol{G}_{med}$ will be computed. We design a secure protocol $SecCorr$ for the calculation of ρ_i. Considering the vectors $\boldsymbol{G}_i$ and $\boldsymbol{G}_{med}$ as random variables, then the formula for the correlation coefficient ρ_i is as follows:

$$\begin{aligned} \rho_i &= \frac{\mathrm{Cov}\left(\boldsymbol{G}_i, \boldsymbol{G}_{\mathrm{med}}\right)}{\sigma\left(\boldsymbol{G}_i\right) \sigma\left(\boldsymbol{G}_{\mathrm{med}}\right)} \\ &= \frac{\sum_{j=1}^{m}\left(G_{ij} - \overline{G_i}\right)\left(G_{(med)j} - \overline{G_{med}}\right)}{\sqrt{\sum_{j=1}^{m}\left(G_{ij} - \overline{G_i}\right)^2}\sqrt{\sum_{j=1}^{m}\left(G_{(med)j} - \overline{G_{med}}\right)^2}} \end{aligned} \tag{4}$$

where $\mathrm{Cov}()$ denotes covariance, $\sigma()$ denotes standard deviation and $\overline{G_i}, \overline{G_{med}}$ represents the mean of vectors $\boldsymbol{G}_i, \boldsymbol{G}_{med}$. The communication overhead of Formula 4 can be reduced by formalizing it as a vector multiplication computation. Denote the vector $\left\{G_{ij} - \overline{G_i}\right\}_{j=1}^{m}$ as $\boldsymbol{C}_i$ and $\left\{G_{(med)j} - \overline{G_{med}}\right\}_{j=1}^{m}$ as $\boldsymbol{C}_{med}$, then Formula 4 can be expressed as

Algorithm 2 $SecMed$: Secure Median calculation.

Input: For $i \in [1, n]$, SP_0 has shares $\langle \boldsymbol{G}_i \rangle_0$ and SP_1 has shares $\langle \boldsymbol{G}_i \rangle_1$, where $\boldsymbol{G}_i$ is a m-dimensional vector.

Output: For $j \in [1, m]$, SP_0, SP_1 get share of the median G_{ij}, denoted as $\langle G_{(med)j} \rangle$.

1: **for** $x = 0, 1$ **do**
2: SP_x randomly generates a m-dimensional vector$\langle \boldsymbol{r} \rangle_x$.
3: **for** $i \in [1, n]$ **do**
4: SP_x computes $\langle \boldsymbol{R}_i \rangle_x = \langle \boldsymbol{G}_i \rangle_x + \langle \boldsymbol{r} \rangle_x$.
5: SP_x sends $\langle \boldsymbol{R}_i \rangle_x$ to CS.
6: **end for**
7: **end for**
8: CS sets $\boldsymbol{R}_i = \langle \boldsymbol{R}_i \rangle_0 + \langle \boldsymbol{R}_i \rangle_1$, $i \in [1, n]$.
9: **for** $j \in [1, m]$ **do**
10: CS computes the median as follows:

$$R_{(med)j} = median(R_{1j}, R_{2j}, \cdots, R_{nj})$$

11: **end for**
12: CS sets the vector $\boldsymbol{R}_{med} = \{R_{(med)j}\}_{j=1}^{m}$.
13: CS sends secret shares $\langle \boldsymbol{R}_{med} \rangle := (\langle \boldsymbol{R}_{med} \rangle_0, \langle \boldsymbol{R}_{med} \rangle_1)$ to SP_0, SP_1 respectively.
14: **for** $x = 0, 1$ **do**
15: SP_x outputs $\langle \boldsymbol{G}_{med} \rangle_x = \langle \boldsymbol{R}_{med} \rangle_x - \langle \boldsymbol{r} \rangle_x$.
16: **end for**

$$\rho_i = \frac{\boldsymbol{C}_i \cdot \boldsymbol{C}_{med}}{\sqrt{\left(\boldsymbol{C}_i\right)^2}\sqrt{\left(\boldsymbol{C}_{med}\right)^2}} \tag{5}$$

and the calculation can be completed by calling the vector multiplication protocol three times. Specifically, the SPs compute $\langle \boldsymbol{C}_i \cdot \boldsymbol{C}_{med} \rangle$, $\left\langle \left(\boldsymbol{C}_i\right)^2 \right\rangle$, $\left\langle \left(\boldsymbol{C}_{med}\right)^2 \right\rangle$ and send the shares to CS. After the shares are reconstructed, the CS can calculate the correlation coefficient from the intermediate data. Furthermore, $\left\langle \left(\boldsymbol{C}_i\right)^2 \right\rangle$ can be computed locally by the device P_i and sent directly to the CS. Since non-colluding device P_i does not know the relevant knowledge of the median gradient, malicious devices cannot control ρ_i by manipulating the value of $\left\langle \left(\boldsymbol{C}_i\right)^2 \right\rangle$. Algorithm 3 shows the overall process of the protocol.

Gradients aggregating. After CS calculates the correlation coefficient ρ_i, the update coefficient μ_i of the gradient $\boldsymbol{G}_i$ can be calculated as follows,

$$\mu_i = \max\left\{0, \ln\left(\frac{1 + \rho_i}{1 - \rho_i}\right) - 0.5\right\} \tag{6}$$

During this process, the poisoned gradient has a low correlation coefficient with the benchmark, and its μ_i will be close to or equal to zero. After that, the servers should aggregate the gradients according as follows,

$$\boldsymbol{G}^{global} = \sum_{i \in [1,n]} \frac{\mu_i}{\sum_{i \in [1,n]} \mu_i} \boldsymbol{G_i} \tag{7}$$

The process of gradient aggregation can be formalized as matrix-vector multiplication. Denote matrix $\boldsymbol{S} = (\boldsymbol{G}_1, \boldsymbol{G}_2, \cdots, \boldsymbol{G}_n)^T$, vector $\boldsymbol{\beta} = \{\mu_i / \sum_{i \in [1,n]} \mu_i\}_{i=1}^{n}$, then the gradient aggregation process can be expressed as

$$\boldsymbol{G}^{global} = \boldsymbol{S} \cdot \boldsymbol{\beta} \tag{8}$$

Algorithm 3 $SecCorr$: Secure Correlation coefficient calculation.

Input: SP_0 has shares $\langle \boldsymbol{G}_i \rangle_0$, $\langle \boldsymbol{G}_{med} \rangle_0$; SP_1 has shares $\langle \boldsymbol{G}_i \rangle_1$, $\langle \boldsymbol{G}_{med} \rangle_1$; device P_i has $\boldsymbol{G}_i$.

Output: CS gets the correlation coefficient ρ_i.

1: **Preprocessing Phase**:
2: P_i computes the mean of $\boldsymbol{G}_i$, denoted as $\overline{G_i}$ and sets $\boldsymbol{C}_i = \left\{G_{ij} - \overline{G_i}\right\}_{j=1}^{m}$.
3: P_i computes $\sqrt{(\boldsymbol{C}_i)^2}$ and sends it to CS;
4: **Computing Phase**:
5: **for** $x = 0, 1$ **do**
6: SP_x computes $\left\langle \overline{G_i} \right\rangle_x = \frac{1}{m} \sum_{j \in [1,m]} \langle G_{ij} \rangle_x$ and $\left\langle \overline{G_{med}} \right\rangle_x = \frac{1}{m} \sum_{j \in [1,m]} \left\langle G_{(med)j} \right\rangle_x$.
7: SP_x sets $\langle \boldsymbol{C}_i \rangle_x = \left\{ \langle G_{ij} \rangle_x - \langle \overline{G_i} \rangle_x \right\}_{j=1}^{m}$ and $\langle \boldsymbol{C}_{med} \rangle_x = \left\{ \langle G_{(med)j} \rangle_x - \langle \overline{G_{med}} \rangle_x \right\}_{j=1}^{m}$.
8: **end for**
9: SP_0 and SP_1 get $\langle \boldsymbol{C}_i \cdot \boldsymbol{C}_{med} \rangle_0, \langle (\boldsymbol{C}_{med})^2 \rangle_0$ and $\langle \boldsymbol{C}_i \cdot \boldsymbol{C}_{med} \rangle_1, \langle (\boldsymbol{C}_{med})^2 \rangle_1$ respectively by executing the multiplication protocol based on secret sharing.
10: SP_0 and SP_1 send $\langle \boldsymbol{C}_i \cdot \boldsymbol{C}_{med} \rangle_0, \langle (\boldsymbol{C}_{med})^2 \rangle_0$ and $\langle \boldsymbol{C}_i \cdot \boldsymbol{C}_{med} \rangle_1, \langle (\boldsymbol{C}_{med})^2 \rangle_1$ to CS respectively.
11: CS reconstructs $\boldsymbol{C}_i \cdot \boldsymbol{C}_{med}$ and $(\boldsymbol{C}_{med})^2$.
12: CS computes the correlation coefficient ρ_i as follows:

$$\rho_i = \frac{\boldsymbol{C}_i \cdot \boldsymbol{C}_{med}}{\sqrt{(\boldsymbol{C}_i)^2} \sqrt{(\boldsymbol{C}_{med})^2}}$$

CS sends shares of $\langle \boldsymbol{\beta} \rangle$ to SPs, SP_x sets $\langle \boldsymbol{S} \rangle_x = (\langle \boldsymbol{G}_1 \rangle_x, \langle \boldsymbol{G}_2 \rangle_x, \cdots, \langle \boldsymbol{G}_n \rangle_x)^T$, then SPs calculate Formula 8 under secret sharing and SP_x obtains $\langle \boldsymbol{G}^{global} \rangle_x$. Finally, the SP_x set $\langle \boldsymbol{w} \rangle_x = \langle \boldsymbol{w} \rangle_x - \eta \cdot \langle \boldsymbol{G}^{global} \rangle_x$ to update the parameters, where $\langle \boldsymbol{w} \rangle_x$ represents the secret shares of global model held by SP_x. The corresponding secure protocol is $SecAgg$, the details are given in Algorithm 4.

Algorithm 4 $SecAgg$: Secure Gradients Aggregation.

Input: SP_0 and SP_1 have shares $\langle \boldsymbol{G}_i \rangle_0$ and $\langle \boldsymbol{G}_i \rangle_1$, respectively. CS has the correlation coefficient ρ_i, $i \in [n]$.

Output: SP_0 and SP_1 get global gradient shares $\langle \boldsymbol{G}^{global} \rangle_0$ and $\langle \boldsymbol{G}^{global} \rangle_1$, respectively.

1: CS computes $\mu_i = \max \left\{ 0, \ln \left(\frac{1+\rho_i}{1-\rho_i} \right) - 0.5 \right\}$, $i \in [n]$.
2: CS sets vector $\boldsymbol{\beta} = \{\mu_i / \sum_{i \in [1,n]} \mu_i\}_{i=1}^{n}$ and send shares $\langle \boldsymbol{\beta} \rangle_0$, $\langle \boldsymbol{\beta} \rangle_1$ to SP_0, SP_1, respectively.
3: **for** $x = 0, 1$ **do**
4: SP_x sets $\langle \boldsymbol{S} \rangle_x = (\langle \boldsymbol{G}_1 \rangle_x, \langle \boldsymbol{G}_2 \rangle_x, \cdots, \langle \boldsymbol{G}_n \rangle_x)^T$.
5: SP_x gets $\langle \boldsymbol{S} \cdot \boldsymbol{\beta} \rangle_x$ by executing the multiplication protocol based on secret sharing.
6: SP_x outputs $\langle \boldsymbol{G}^{global} \rangle_x = \langle \boldsymbol{S} \cdot \boldsymbol{\beta} \rangle_x$.
7: **end for**

IV. Theoretical Analysis

A. Security Analysis

We provide a proof for SPs by the ideal/real-world paradigm, and due to the universal composability framework [47], we need to prove the security of individual protocols. In the ideal-world, we construct a simulator Sim to simulate the view of SPs. And Sim calls ideal functions to perform computations. We assume that the ideal functions $\mathcal{F}_{\text{Med}}$, $\mathcal{F}_{\text{Weight}}$, and $\mathcal{F}_{\text{Model}}$ correspond to the protocols for calculating the median, the weight based on correlation coefficients and the global gradients, respectively.

Theorem 1. *SPEFL does not leak privacy data to service providers in the semi-honest setting.*

Proof. In the real-world, SPs complete the calculation by various protocols based on secret sharing. Sim performs the following operations: 1) Sim calls $\mathcal{F}_{\text{Med}}$ $(\left\langle \widetilde{\boldsymbol{G}_i} \right\rangle), i \in [n]$, and appends the output $\left\langle \widetilde{\boldsymbol{G}_{med}} \right\rangle$ to the general view for SPs. 2) Sim calls $\mathcal{F}_{\text{Weight}}$ $(\left\langle \widetilde{\boldsymbol{G}_i} \right\rangle, \left\langle \widetilde{\boldsymbol{G}_{med}} \right\rangle)$, followed by appending its output $\left\langle \widetilde{\beta_i} \right\rangle$ to the general view for SPs. 3) Sim calls $\mathcal{F}_{\text{Model}}$ $(\left\langle \widetilde{\boldsymbol{G}_i} \right\rangle, \left\langle \widetilde{\beta_i} \right\rangle, \langle \widetilde{\boldsymbol{w}} \rangle)$ to get the view for SPs, and gets the updated model. To demonstrate that the view of Sim is indistinguishable from the view of SPs in the real-world, we define a sequence of hybrid arguments, $H_1, \cdots, H_3$, where $H_i, i \in \{1, 2, 3\}$ is the simulated view.

- First, we let H_1 be the ideal view of $\mathcal{F}_{\text{Med}}$. Since the shares $\langle \boldsymbol{G}_i \rangle, \langle \boldsymbol{G}_{med} \rangle$ of SPs in real execution is randomly distributed in the protocol $SecMed$, so it is indistinguishable from H_1.
- Then, we let H_2 be the view of $\mathcal{F}_{\text{Weight}}$ in ideal execution. In real execution, SPs use the multiplication protocol based on secret sharing to complete weight calculation. The view in real-world are shares of intermediate values and the Beaver triples in $SecCorr$ and the final output $\langle \beta \rangle$ in $SecAgg$ which are randomly distributed in the ring $\mathbb{Z}_{2^L}$. So the real view is indistinguishable from H_2.
- Finally, we update the model locally based on secret sharing with multiplication protocol. Let H_3 be the ideal view of $\mathcal{F}_{\text{Model}}$. The output of $SecAgg$ of SPs is random share which is indistinguishable from H_3.

Next, we prove the security of CS. In the protocol $SecMed$, CS can obtain $\boldsymbol{G}_i + \boldsymbol{r}$. Since $\boldsymbol{r}$ is a random vector, $\boldsymbol{G}_i + \boldsymbol{r}$ is also randomly distributed and does not contain valuable information. In the protocol $SecCorr$, CS computes the correlation coefficient ρ based on the shares of the gradient vectors and further computes μ. According to Equation 5, inferring the private information of the gradient vector is equivalent to inferring the vector information based on the inner product result. It has been proved in many works that under the semi-honest threat model, the attacker cannot recover any vector information from the dot product of vectors when the number of unknown variables is much greater than the number of linear equations [48], [49]. In PEFL [25], the author also states that the servers cannot invert the magnitude and sign of gradient vectors even if the coefficient is public. According to the correlation coefficient formula, there are $2m$ unknown variables and 3 linear equations, where m is the dimension of the gradient vector. In practice, m is a high-dimensional vector with tens of thousands or millions of dimensions, so the protocol is secure under the semi-honest model.

In summary, the ideal-world and real-world views of SPs are indistinguishable. And CS can only compute statistical characteristics such as variance that cannot be used to infer clients' privacy.

B. Correctness Analysis

To demonstrate that the privacy-preserving computation techniques in SPEFL do not compromise the effectiveness of defending against poisoning attacks, we show that the results obtained from ciphertext are equivalent to those computed on the plaintext.

In the protocol $SecMed$, the SPs calculate the median of the gradients $\boldsymbol{G}_{med} = \boldsymbol{R}_{med} - \boldsymbol{r}$, where $\boldsymbol{R}_{med} = median(\boldsymbol{R}_0, \boldsymbol{R}_1, \cdots, \boldsymbol{R}_n), \boldsymbol{R}_i = \boldsymbol{G}_i + \boldsymbol{r}$. Since the confusion vector $\boldsymbol{r}$ is eliminated, the protocol $SecMed$ can correctly calculate the median of gradients.

In the protocol $SecCorr$, CS computes the correlation coefficient ρ_i between the gradient $\boldsymbol{G}_i$ and $\boldsymbol{G}_{med}$ according to Equations 4 and 5, from which we can see that $SecCorr$ can correctly calculate the correlation coefficient between the gradient and the benchmark.

The correctness of the additive secret-sharing cryptographic primitives used in $SecMed$, $SecCorr$ and $SecAgg$ can be found in [43]. Therefore, our protocol can correctly complete the calculation on the ciphertext and defend against poisoning attacks.

C. Complexity Analysis

In SPEFL, the devices only need to perform the addition on the finite field to calculate secret shares after the local training is completed, and the main overhead comes from the calculation based on secret sharing between two SPs and the communication between CS and SP. In the protocol $SecMed$, two SPs need to confuse the m-length gradient secret sharing of n devices and send them to CS, so the communication cost is $O(Tmn)$, where T is the number of iterations. For each device, in order to calculate the correlation coefficient, SPs need to call the multiplication protocol based on secret sharing to calculate the inner product of the vectors and send the results to CS. In addition, the covariance information of the median needs to be calculated separately. Therefore, the computational complexity of calculating the correlation coefficients is $O(Tm(2n+1))$, and the communication cost is $O(T(2n+1))$. When aggregating gradients, CS needs to send the n-length secret shares of the weight vector to the SPs, which incurs the communication cost of $O(Tn)$. SPs need to call a matrix multiplication to compute the product of the gradient matrix and the weight vector with computational complexity $O(Tmn)$.

The computational overhead of SPEFL is lower than the works adopted by Krum and Bulyan as the defense schemes, such as [31], [32] which all have $O(Tmn^2)$ computational complexity. Furthermore, many operations in the customized protocol we designed are based on secret sharing over the ring $\mathbb{Z}_{2^L}$, and native implementation of matrix multiplication over long (or int) automatically implements the modulo operation over $\mathbb{Z}_{2^{64}}$ (or $\mathbb{Z}_{2^{32}}$), which is secure and efficient enough [43]. In comparison, the schemes based on homomorphic encryption require encryption and decryption, which imposes a heavy burden on both servers and devices. For example, PEFL [25], we have adopted the same defense method as it, but its communication overhead of a single iteration reaches $O(\sigma Tmn)$. PEFL uses homomorphic encryption ciphertext packaging technology to encrypt multiple plaintexts into one ciphertext, which will increase the overhead of σ, where σ is linearly related to the number of plaintexts [50]. Moreover, it requires complex homomorphic calculations on the ciphertext, so the overhead of a single atomic operation (such as multiplication) in the HE-based scheme is higher than that of SPEFL. Moreover, in order to realize m-length sharing between n users, the communication round of the schemes based on Shamir's secret sharing [32] reach $O(Tmn^2)$. Shamir's secret sharing also requires clients to perform frequent polynomial reconstructions, while SPEFL only requires IoT devices to perform addition over finite fields. The servers in SPEFL use additive secret sharing to preserve privacy, and we design computationally efficient protocols that do not need to perform expensive computations such as garbled circuits, unlike [39]. Therefore, SPEFL can efficiently perform secure and private FL tasks.

V. EVALUATION

In this section, we conduct experiments to evaluate the performance of SPEFL. By comparison with previous work, we demonstrate the accuracy and efficiency of our framework.

A. Experimental Setup

We simulate multiple servers and IoT devices by Python, following previous work [51]. Three server processes are running on one high-performance machine with an Intel Xeon Gold 5118 CPU (2.40GHz). A desktop computer with an Intel Core i7 CPU (1.80GHz) is used to simulate multiple resource-limited IoT device nodes.

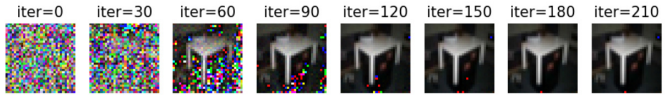
Datasets and Model Architectures: We evaluate the performance of SPEFL on commonly used HAR [52], MNIST [53], and CIFAR10 [54] datasets. For the HAR dataset, we use a two-layer fully-connected network, and the shape of each layer is 1152×100 and 100×6 sequentially. The convolutional neural network LeNet [55] and the residual neural network ResNet20 [56] are used as global models for the MNIST dataset and the CIFAR10 dataset respectively. In the experiment, the data set is evenly divided into each device, and the data set held by each device is independent and identically distributed (IID). For MNIST and CIFAR-10, the default total number of devices is 41. For the HAR dataset, data is collected from 30 users, and each device holds data for one user. Therefore, the total number of devices is 30. The default percentage of malicious devices in different scenarios is 20%. The experimental setup refers to existing work [24], [25], [39].

Poisoning Attacks: Referring to related work, we conduct experiments on the three representative poisoning attacks, label-flipping attack, backdoor attack, and local model attack. In the label-flipping attack, malicious devices reverse the labels

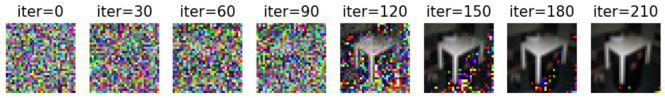
of normal features to the target class. For example, malicious devices modify the sample label from 1 to 4. In the backdoor attack [11], attackers induce the model to identify data with a specific trigger as the target class. For example, malicious devices randomly select a certain amount of local data, cover the lower right 5×5 sub-matrix in the data matrix with the maximum value, and mark it as the target class. In the local model attack, we use the poisoning attack method **Attack Median** in [57], which is a model poisoning attack against robust aggregation methods.

B. Privacy and Robustness Evaluation of SPEFL

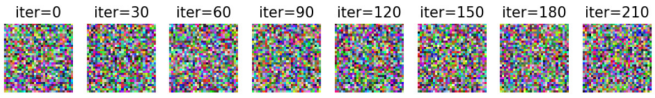
1) Privacy Evaluation: In addition to the above security analysis, we demonstrate that SPEFL can protect privacy sufficiently through the DLG attack experiments. In SPEFL, the CS knows $\boldsymbol{G}_i + \boldsymbol{r}$ when calculating the median, where $\boldsymbol{r}$ is a random vector. While in PEFL [25], CS not only learns $\boldsymbol{G}_i + \boldsymbol{r}$ but also knows $r_1 \cdot \boldsymbol{G}_i$ and $r_2 \cdot \boldsymbol{G}_{med}$ in calculating the correlation coefficient, where r_1, r_2 are random numbers. We found that client's private training data can still be inferred from $r_i \cdot G_i$ using DLG [14], as shown in Fig. 2. Unlike the Laplacian or Gaussian distributed noise added in differential privacy, the noise added in SPEFL is randomly distributed, and the obfuscated data looks like meaningless random numbers. Therefore, the noise added in SPEFL can effectively protect private data.



(a) Infer data from original gradients $\boldsymbol{G}$.



(b) Infer data from obfuscated data $r \cdot \boldsymbol{G}$.



(c) Infer data from obfuscated data $\boldsymbol{G} + \boldsymbol{r}$.

Fig. 2. Recover Client's Private Data Using DLG.

2) Robustness Evaluation: We implemented SPEFL as well as some other defense methods, such as PEFL [25], Krum [8], and FLTrust [24]. Since SPEFL needs to convert floating-point numbers to integers for calculation, this will cause some loss of precision. We reproduce other frameworks on the plaintext without privacy protection. This is to demonstrate that the loss of precision in SPEFL does not reduce the defense against poisoning attacks even compared to the plaintext methods. Table II shows the test errors of SPEFL and other FL schemes with different datasets and poisoning attacks. It can be observed that SPEFL can achieve close accuracy to FedAvg with no malicious device. However, in the presence of poisoning attackers, the test errors of FedAvg rise significantly while SPEFL still maintains high accuracy. In all cases, SPEFL is more accurate than Krum. One of the reasons is that Krum only selects a set of gradients to update, while SPEFL aggregates multiple benign gradients. Although SPEFL is computed based on ciphertext, it has similar test errors to the secure aggregation scheme on the plaintext, indicating that our privacy-preserving scheme can maintain model accuracy and robustness simultaneously.

In Table III and Table IV, we give the performance under different proportions of malicious devices on the MNIST dataset. As the proportion of attackers increases, SPEFL can still resist poisoning attacks and maintain high accuracy as PEFL and FLTrust, while fewer attackers can successfully attack FedAvg, and the accuracy of Krum also drops significantly when there are 50% attackers. There are similar conclusions in other datasets and model structures, which are not shown due to limited space.

3) Effectiveness of the Defense Method: In SPEFL, the correlation coefficient between the poisoned gradient and the median benchmark will be significantly lower than that of the benign gradient. According to Equation 6 in $SecAgg$, this will make the weights of poisoned gradients close to or equal to zero in aggregation. Fig. 3 shows the correlation coefficients between the gradient of each device and the benchmark under different attacks in three datasets. It can be observed that the correlation coefficients of the poisoned gradients are significantly lower than those of the benign gradients. This difference allows benign gradients to participate in aggregation while poisoned gradients have minimal weight during the execution of $SecAgg$.

Due to the finer operation of the gradient by Attack Median, the difference between poisoned gradients and benign gradients under Attack Median is smaller than that of label-flipping attack and backdoor attack. But this difference is still significant enough for SPEFL to defend against poisoning attacks. Because SPEFL combines median and linear correlation, it not only analyzes the gradient vector in a single dimension but also considers the overall structure of the gradients.

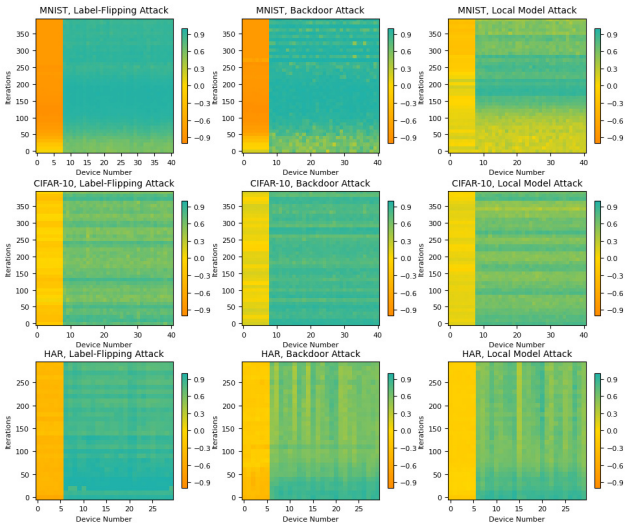


Fig. 3. Correlation coefficients between the benchmark and gradients of different devices. The first 20% devices are malicious.

TABLE II
TEST ERRORS OF DIFFERENT AGGREGATION SCHEMES ON DIFFERENT DATASETS.

	No Attack			Label-flipping attack			Backdoor attack			Local Model Attack		
	HAR	MNIST	CIFAR	HAR	MNIST	CIFAR	HAR	MNIST	CIFAR	HAR	MNIST	CIFAR
FedAvg	0.18	0.04	0.16	0.38	0.18	0.27	0.41	0.13	0.38	0.28	0.10	0.24
Krum	0.20	0.09	0.38	0.21	0.10	0.39	0.19	0.09	0.42	0.31	0.12	0.42
FLTrust	0.17	0.04	0.18	**0.17**	0.04	0.18	0.17	0.06	**0.18**	**0.17**	0.04	0.18
PEFL	**0.17**	**0.04**	**0.17**	0.18	**0.04**	**0.18**	**0.17**	**0.05**	0.19	0.18	**0.04**	**0.18**
SPEFL	0.18	0.04	0.18	0.19	0.04	0.19	0.19	0.05	0.19	0.18	0.05	0.18

TABLE III
MODEL ACCURACY UNDER LABEL-FLIPPING ATTACK ON MNIST.

		No Poisoner	10% Poisoner	20% Poisoner	30% Poisoner	40% Poisoner	50% Poisoner
FedAvg	Pred(S')=T[1]	0	0.95	0.98	0.98	1	1
	Pred(S')=S[2]	0.97	0.03	0.01	0.001	0	0
	Pred(R')=R[3]	0.95	0.93	0.92	0.88	0.81	0.67
Krum	Pred(S')=T	0	0	0.002	0.001	0.001	0.96
	Pred(S')=S	0.95	0.94	0.93	0.93	0.93	0.03
	Pred(R')=R	0.90	0.91	0.89	0.90	0.90	0.77
FLTrust	Pred(S')=T	0	0	0	0	0	0
	Pred(S')=S	0.98	0.98	0.98	0.98	0.97	0.97
	Pred(R')=R	0.96	0.96	0.96	0.95	0.95	0.95
PEFL	Pred(S')=T	0	0	0	0.01	0.01	0.04
	Pred(S')=S	0.98	0.98	0.97	0.98	0.97	0.93
	Pred(R')=R	0.95	0.95	0.95	0.95	0.93	0.90
SPEFL	Pred(S')=T	0	0	0.001	0.01	0.02	0.07
	Pred(S')=S	0.98	0.98	0.98	0.98	0.97	0.91
	Pred(R')=R	0.95	0.95	0.93	0.94	0.93	0.88

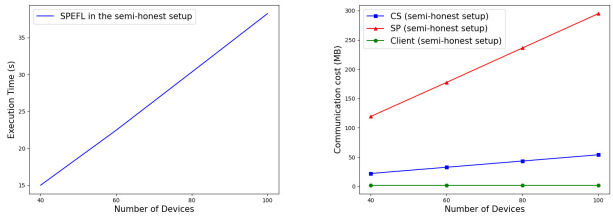
[1] Pred(S)=T means the proportion of identifying the source class as the target class, that is, the attack success rate.
[2] Pred(S)=S means the proportion of correctly identifying the source class.
[3] R represents other classes except the source class, and Pred(R)=R means the model accuracy rate for non-source class.

C. Execution Efficiency of SPEFL

In this section, we implement several other privacy-preserving and robust FL methods and compare our execution efficiency with them.

Fig. 4 shows the execution time and communication cost of SPEFL with the increasing number of devices. From Fig. 4a and Fig. 4b we can observe that the execution time of SPEFL and the communication cost of CS and SP all increase linearly with the number of devices. Theoretically, SPEFL has $O(n)$ overhead for the same model, where n is the number of devices, which is consistent with the results in Fig. 4. The protocol of SPEFL is mainly carried out between two SPs that hold secret sharing, so the communication volume of CS responsible for auxiliary calculation is smaller than that of SP. Furthermore, the communication overhead of IoT devices in SPEFL is much lower than that of CS and SP, and the increase in the total number of devices has no impact on the communication cost of a single device, which is reflected in Fig. 4b.

We compare execution time and communication efficiency with previous work PEFL (HE-based) [25], privacy-preserving Krum (MPC-based) [31], and privacy-preserving FLTrust (MPC-based) [58] on different model architectures. The privacy-preserving Krum and FLTrust are based on secret sharing, the same as SPEFL. Both PEFL and SPEFL adopt the same robust aggregation scheme, but SPEFL achieves privacy protection based on secret sharing, while PEFL is



(a) Execution time of SPEFL. (b) Communication cost.

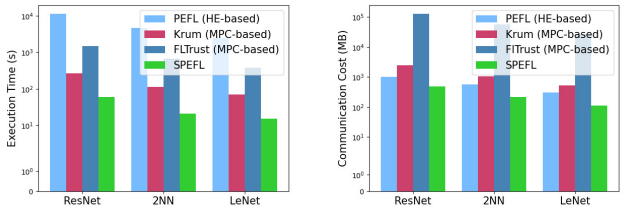
Fig. 4. Execution Time and Communication cost of SPEFL with increasing number of devices on LeNet.

based on homomorphic encryption for privacy protection. Fig. 5 shows the difference in execution time and communication overhead for the three methods with 41 devices. Due to the different system structures, Fig. 5b does not show the communication difference between different parties but counts the traffic of the entire system. Both the execution time and communication overhead of SPEFL are lower than privacy-preserving Krum, FLtrust, and PEFL. Specifically, SPEFL achieves privacy-preserving robust aggregation in about **141**$\times$ less time than HE-based frameworks, about **25.5**$\times$ less than privacy-preserving FLTrust and about **4.2**$\times$ less than privacy-preserving Krum for the total time. And the communication cost of SPEFL is about **half** of PEFL, **250**$\times$ less than FLTrust and **4.3**$\times$ less than Krum. This is because Krum has $O(n^2)$

TABLE IV
MODEL ACCURACY UNDER BACKDOOR ATTACK ON MNIST.

		No Poisoner	10% Poisoner	20% Poisoner	30% Poisoner	40% Poisoner	50% Poisoner
FedAvg	Pred(S')=T[1]	0	0.98	0.99	1	1	1
	Pred(S')=S[2]	0.97	0.01	0.01	0.009	0	0
	Pred(R')=R[3]	0.95	0.93	0.92	0.9	0.81	0.71
Krum	Pred(S')=T	0	0.02	0.08	0.11	0.16	0.98
	Pred(S')=S	0.91	0.90	0.92	0.87	0.81	0.01
	Pred(R')=R	0.91	0.91	0.90	0.89	0.88	0.81
FLTrust	Pred(S')=T	0	0	0	0	0	0
	Pred(S')=S	0.98	0.98	0.98	0.98	0.97	0.96
	Pred(R')=R	0.96	0.96	0.96	0.95	0.95	0.95
PEFL	Pred(S')=T	0	0	0	0.01	0.02	0.04
	Pred(S')=S	0.98	0.97	0.96	0.96	0.94	0.88
	Pred(R')=R	0.95	0.95	0.95	0.95	0.92	0.76
SPEFL	Pred(S')=T	0	0.001	0.001	0.01	0.04	0.08
	Pred(S')=S	0.98	0.96	0.96	0.94	0.93	0.90
	Pred(R')=R	0.95	0.94	0.94	0.93	0.92	0.87

[1] S' represents data with the trigger, Pred(S')=T means the proportion of identifying data with the trigger as the target class.
[2] Pred(S')=S means the proportion of correctly identifying the source class.
[3] R represents other data without the trigger, and Pred(R)=R means the model accuracy rate for data without the trigger.



(a) Execution time. (b) Communication cost.

Fig. 5. Execution Time and Communication cost over different model architectures.

complexity, while SPEFL has only $O(n)$ complexity. PEFL also has $O(n)$ complexity, which makes its communication overhead lower than Krum, but the privacy protection method based on homomorphic encryption increases the execution time. In addition, the FLTrust implemented based on the general MPC protocol needs huge computation and communication overhead to complete the nonlinear calculation. And SPEFL reduces the overhead through the three-server architecture and customized computation protocols.

Table V presents the detailed execution times of different frameworks on LeNet in the LAN scenario. Both SPEFL and PEFL use the defense method based on the median and correlation coefficient, so the server-side calculation time of each step is listed. As for Krum and FLTrust, due to different aggregation methods, only their total execution time is given. And the total execution time in SPEFL is about **141**× faster than PEFL. SPEFL's servers operation time is about **258**× times faster than PEFL and **42**× times faster than MPC-based FLTrust, this is because SPEFL adopts a three-server architecture and designs customized calculation protocols and much of the computation is actually done based on secret sharing over the ring $\mathbb{Z}_{2^L}$, which is efficient enough as plaintexts computation. HE-based PEFL not only complicates the calculation between servers but also requires each device to perform expensive encryption and decryption. The resource-limited devices need to execute the homomorphic encryption algorithm at each iteration, while devices in SPEFL only need to perform addition computations over finite fields to generate and reconstruct secret shares. Therefore, a single device of SPEFL obfuscates the gradient and updates the model about **217**× and **86**× faster than PEFL, respectively. Moreover, the high complexity of Krum makes the calculation overhead of servers too large. Therefore, SPEFL is more friendly to resource-limited IoT devices.

Table VI shows the execution time of different frameworks for LAN/WAN scenarios. The network bandwidth and latency are simulated using Linux Traffic Tools(tc) command, as in existing work [43]. We consider the WAN setting with 12MB/s and 80 ms RTT (round-trip time). It can be observed that due to a large amount of communication, the execution time of MPC-based Krum and FLTrust increases significantly in the low-bandwidth WAN scenario. The main overhead of PEFL comes from HE computation rather than communication, so its execution time increases in a small proportion in the WAN scenario. The communication overhead of SPEFL is lower than other frameworks, so the execution time in the WAN scenario does not increase significantly, which is better than others.

VI. SECURITY-ENHANCED SPEFL

In the previous discussion, we assumed that all three servers were semi-honest, but if there is a malicious server, it may tamper with the global model and corrupt the training results [59]–[62]. Therefore, we design additional security-enhancing verifiable protocols for SPEFL. The function is to verify that the server updates the model correctly according to the negotiated method.

A. Threat Model

In our security-enhanced protocol, the behavior of CS and devices is the same as described in Section III-B. We consider

TABLE V
DETAILED EXECUTION TIME OF EACH FRAMEWORK

	Computation time on the device side		Computation time on the servers side				Total time
	Gradient Upload	Model Update	Correlation Coefficient Calculation	Gradient Aggregation	Median Calculation	Total	
SPEFL	72 ms	70 ms	2914 ms	2892 ms	2930 ms	8.736 s	**16.792 s**
PEFL (HE-based)	15618 ms	6049 ms	720884 ms	823706 ms	711842 ms	2256.432 s	2368.554 s
Krum (MPC-based)	70 ms	70 ms	61.544 s				70.571 s
FLTrust (MPC-based)	75 ms	65 ms	373.884 s				382.412s

TABLE VI
EXECUTION TIME OF DIFFERENT FRAMEWORKS IN LAN/WAN SCENARIOS.

	LAN		WAN	
	Device side	Server side	Device side	Server side
SPEFL	142ms	8.736s	617ms	19.079s
PEFL(HE-based)	21667ms	2256.432s	22209ms	2321.016s
Krum(MPC-based)	140ms	61.544s	582ms	465.651s
FLTrust(MPC-based)	140ms	373.884s	688ms	7060.162s

that there may be a malicious SP among the two SPs, who will deviate from the protocol rules.

A malicious SP can tamper with the secret shares when delivering the global model, causing victim devices to reconstruct the corrupted model imperceptibly. This means that a malicious SP has the ability to launch untargeted model degradation attacks and compromise the integrity of the global model [63], [64]. Furthermore, the SP may collude with malicious devices to launch a poisoning attack targeting specific samples. As shown in Fig. 6, suppose a malicious SP_x is corrupted by a malicious adversary P_m and obtains abnormal gradients $\boldsymbol{G}_m$ from P_m. Then the malicious SP_x can set $\langle \boldsymbol{w} \rangle'_x = \langle \boldsymbol{w} \rangle_x + \boldsymbol{G}_m$. In this way, the global model $\boldsymbol{w}$ has tampered with $\boldsymbol{w}' = \boldsymbol{w} + \boldsymbol{G}_m$, which is equivalent to the poisoned gradient participating in the aggregation with high weight. Table VII shows the change in attack success rate after a malicious SP tampered with the secret shares and increased the weight of poisoned gradients on LeNet. When there is malicious SP colluding with a malicious device, the success rate of poisoning attack increases from 0.1% to 97%. Fortunately, our improved scheme can detect malicious behaviors of malicious SPs and avoid such situations.

TABLE VII
INFLUENCE OF MALICIOUS SP ON LABEL-FLIPPING ATTACK.

	SPEFL with no malicious SP	**SPEFL with a malicious SP**
Pred(S)=T	0.001	0.97
Pred(S)=S	0.98	0.02
Pred(R)=R	0.93	0.91

It's worth noting that SPEFL isn't the only one having this problem. Many privacy-preserving FL frameworks are designed in semi-honest settings without considering malicious behavior [25], [31], [39], but SPEFL considers this situation and gives an effective method.

B. Verifiable Protocol in Malicious Model

To avoid the above problem, we use a verifiable secret sharing consisting of secret shares and verification shares

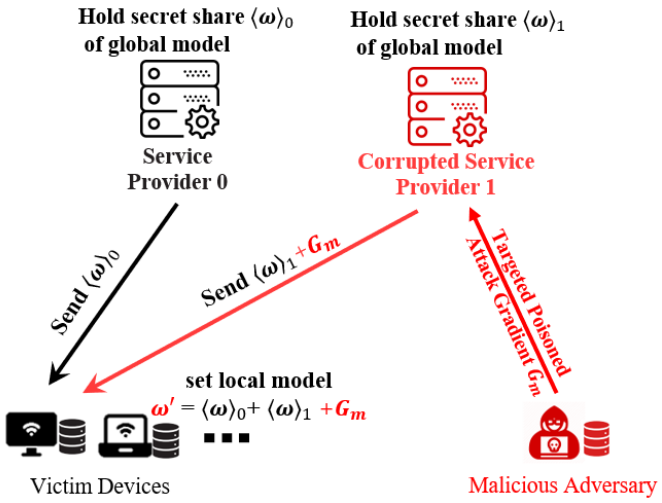


Fig. 6. The corrupted service provider modifies the secret share when broadcasting the global model.

based on SPDZ [65]. The secret shares can be reconstructed into a secret value, and the verification shares can verify the integrity of the secret value.

For a secret value a, its normal secret sharing can be expressed as $\langle a \rangle := (\langle a \rangle_0, \langle a \rangle_1)$, where $a = \langle a \rangle_0 + \langle a \rangle_1$. The verifiable secret sharing can be expressed as

$$[a] := (\delta_a, ([a]_0, [a]_1), (\gamma(a)_0, \gamma(a)_1))$$

where α is a random number called global key and

$$a = [a]_0 + [a]_1 \tag{9}$$

$$\gamma(a)_0 + \gamma(a)_1 = \alpha(\delta_a + a) \tag{10}$$

The verifiable secret sharing requires new computational protocols. For secret sharing $[a], [b]$ and constant c, the associated computation can be expressed as

$$\delta_{a+b} = \delta_a + \delta_b, [a+b]_x = [a]_x + [b]_x, \gamma(a+b)_x = \gamma(a)_x + \gamma(b)_x$$

$$\delta_{c \cdot a} = c \cdot \delta_a, [c \cdot a]_x = c \cdot [a]_x, \gamma(c \cdot a)_x = c \cdot \gamma(a)_x, x = 0, 1$$

According to Algorithm 1, the basic operations of the secret sharing multiplication protocol include constant number multiplication and sharing addition calculations.

In our verifiable protocol, SP_x has $[a]_x, \gamma(a)_x$ and δ_a. α is generated by the CS, unknown to the two SPs, they need to verify that Equation 10 holds. If a malicious SP_x tries to tamper with its own secret share $[a]_x$ to $[a]_x + \triangle$, then it must modify $\gamma(a)_x$ or δ_a at the same time, otherwise the equation $\gamma(a)_0 + \gamma(a)_1 = \alpha(\delta_a + a + \triangle)$ does not hold. However, δ_a is known by both SPs and cannot be tampered with by a single SP and both two SPs do not know the value of α to modify $\gamma(a)$. Therefore, it can be judged whether the secret share has been modified by verifying **whether Equation 10 holds**. Therefore, by using verifiable secret sharing $[\cdot]$ instead of normal secret sharing $\langle\cdot\rangle$, we can detect whether the secret share has been tampered with.

In order to enable participants to use verifiable secret sharing to complete the calculation, this paper designs a share-converting protocol and a share-sending protocol.

Share-Converting Protocol. For a normal secret share $\langle a\rangle$ held by SPs, we design a share-converting protocol so that $\langle a\rangle$ can be converted into a verifiable share $[a]$ with the assistance of CS. Specifically, two SPs negotiate δ_a, and SP_x sends $\langle a\rangle_x + x \cdot \delta_a$ to CS. Then CS generates $\gamma(a)_0, \gamma(a)_1$ and send to SP_0, SP_1 respectively, where $\gamma(a)_0 + \gamma(a)_1 = \alpha(\delta_a + a)$. Since δ_a is not known to CS, it cannot learn the secret a.

Share-Sending Protocol. In poisoning detection and gradients aggregation, devices and CS also need to send verifiable secret shares of gradients and weight vectors to SPs, which requires a new sending protocol. For CS, since it knows the global key α, it can generate a verifiable secret share of a secret by itself and send it to the SPs. For devices, we design a secret-sharing sending protocol so that devices can send verifiable secret shares to SPs. The process of device P_i sending a verifiable share of secret a requires the assistance of CS. First, CS randomly generates a seed share $[s]$, sends s and δ_s to P_i, and sends $\gamma(s)_x$ to SP_x. Then P_i sets $\delta_a = \delta_s - a + s$ and sends $[a]_x$ to SP_x while SP_x sets $\gamma(a)_x = \gamma(s)_x$. In this way, the secret sharing $[a]$ is completed. CS can generate multiple seed shares and send corresponding data in the preprocessing phase, the process of the share-sending protocol is shown in Algorithm 5.

By using the share-converting protocol and share-sending protocol, participants can complete calculations based on verifiable secret sharing. Before training starts, SPs convert the locally generated initial model share $\langle \boldsymbol{w}\rangle$ into the verifiable secret share $[\boldsymbol{w}]$ with the assistance of CS. After the local training of device P_i is completed, the verifiable secret sharing of the gradient is sent to the SPs through the share-sending protocol. In the protocol $SecMed$, SPs convert $\langle \boldsymbol{r}\rangle$ into $[\boldsymbol{r}]$, CS also sends verifiable secret share $[\boldsymbol{R}_{med}]$ instead of $\langle \boldsymbol{R}_{med}\rangle$ to SPs, and they complete related calculations based on the verifiable secret sharing. In protocols $SecCorr$ and $SecAgg$ similarly, CS sends verifiable secret shares, and SPs also perform calculations based on verifiable secret shares.

Share-Verifying Protocol. At the end of each iteration, devices have parameters $\boldsymbol{w}$, SP_0, SP_1 know $\delta_{\boldsymbol{w}}$ and have verification shares $\gamma(\boldsymbol{w})_0, \gamma(\boldsymbol{w})_1$ respectively, and CS has the global key α. It is needed to verify:

$$\gamma(\boldsymbol{w})_0 + \gamma(\boldsymbol{w})_1 = \alpha(\delta_{\boldsymbol{w}} + \boldsymbol{w}) \tag{11}$$

CS only helps to complete the calculation task during training without knowing more specific information about the model or gradients. So this paper uses a secure verifying protocol as follows:

In the preprocessing phase, a Pseudorandom Generator (PRG) is jointly negotiated and each device P_i negotiates a public key k_i with SP_0 and SP_1. If the integrity of $\boldsymbol{w}$ needs to be verified, P_i sends $\boldsymbol{w} - PRG(k_i)$ to CS. SP_0 and SP_1 send $\delta_{\boldsymbol{w}} + PRG(k_i), \gamma(\boldsymbol{w})_0$ and $\delta_{\boldsymbol{w}} + PRG(k_i), \gamma(\boldsymbol{w})_1$ to CS respectively. Then CS verifies the following equation:

$$\gamma(\boldsymbol{w})_0 + \gamma(\boldsymbol{w})_1 = \alpha(\delta_{\boldsymbol{w}} + PRG(k_i) + \boldsymbol{w} - PRG(k_i)) \tag{12}$$

Obviously, Equation 11 and Equation 12 are equivalent. In addition, CS can select a part of devices to participate and randomly select several numbers of parameters for verification in each iteration to reduce overhead. We give the details in Algorithm 6.

In summary, by using our designed verifiable secret sharing, if a malicious SP_x tries to modify the secret share $[w]_x$, it needs to modify $\gamma(w)_x$ at the same time to make the verification pass. However, due to lack of knowledge, it cannot accurately modify $\gamma(w)_x$.

Algorithm 5 Verifiable Share-Sending Protocol.

Input: The device P_i has secret a; CS has the global key α.
Output: SP_0, SP_1 get secret share

$$[a] := (\delta_a, ([a]_0, [a]_1), (\gamma(a)_0, \gamma(a)_1))$$

where SP_x has $\delta_a, [a]_x, \gamma(a)_x$ and $a = [a]_0 + [a]_1, \gamma(a)_0 + \gamma(a)_1 = \alpha(\delta_a + a)$.

Preprocessing Phase:
CS randomly generates k secret share

$$[s_j] := (\delta_{s_j}, ([s_j]_0, [s_j]_1), (\gamma(s_j)_0, \gamma(s_j)_1)), j \in [k]$$

for $j \in [k]$, CS sends s_j, δ_{s_j} to P_i and sends $\gamma(s_j)_0, \gamma(s_j)_1$ to SP_0, SP_1 respectively.
Computing Phase:
P_i randomly chooses $t \in [k]$, sends t to SP_0, SP_1.
P_i computes $\delta_a = \delta_{s_t} - a + s_t$ and randomly generates $[a]_0, [a]_1$, where $[a]_0 + [a]_1 = a$.
for $x = 0, 1$ **do**
    P_i sends $\delta_a, [a]_x$ to SP_x.
    SP_x outputs $\delta_a, [a]_x, \gamma(a)_x = \gamma(s_t)_x$.
end for

C. Theoretical Analysis of the Verifiable Protocol

In this section, we give the following theorem and related proof.

Theorem 2. *Our verifiable protocol achieves statistical security with abort against the active adversarial SP of any one.*

Algorithm 6 Secret Share Verifying Protocol.

Input: CS has the global key α; The device P_i has the secret m-dimensional vector $\boldsymbol{w}$ and key k_i; SP_x has $\delta_{\boldsymbol{w}}, \gamma(\boldsymbol{w}_x)$ and key k_i, $x = 0, 1$.

Output: CS gets 1 if the equation $\gamma(\boldsymbol{w})_0 + \gamma(\boldsymbol{w})_1 = \alpha(\delta_{\boldsymbol{w}} + \boldsymbol{w})$ holds, 0 otherwise.

CS randomly generates l different numbers $s \in [1, m]$, denoted as selected devices set S.
CS sends the set S to P_t.
for $s \in S$ **do**
    P_i sends $w_{si} - PRG(k_i)$ to CS.
end for
for $x = 0, 1$ **do**
    CS sends the set S to SP_x.
    for $s \in S$ **do**
        SP_x computes $\delta_{w_s} + PRG(k_i)$ and sends it to CS.
    end for
end for
for $s \in S$ **do**
    CS computes and verifies

$$(\delta_{w_s} + PRG(k_i))^0 = (\delta_{w_s} + PRG(k_i))^1$$

$$\gamma(w_s)_0 + \gamma(w_s)_1 = \alpha(\delta_{w_s} + PRG(k_i) + w_s - PRG(k_i))$$

    where $(\delta_{w_s} + PRG(k_i))^x$ denotes the data sent by SP_x.
end for
CS output 1 if the equations are all established, 0 otherwise.

1) Security Analysis: We construct a simulator Sim such that the environment $\mathcal{Z}$ cannot distinguish real protocol systems from ideal protocol systems to prove our theorem. The simulator runs a copy of the verifiable protocol, relaying messages between parties and $\mathcal{Z}$, which makes $\mathcal{Z}$ see the same functional interface as when interacting with a real protocol.

We first prove that the views up to output reconstruction have exactly the same distribution in the real and simulated cases. First, the verification share γ_0, γ_1 held by SP are randomly distributed. Second, the values generated by CS in the Share-Converting Protocol and Share-Sending Protocol are uniformly random. Third, the secret is subtracted by a random number before being reconstructed in a secure multiplication, so the values obtained by participants are independent and uniformly distributed.

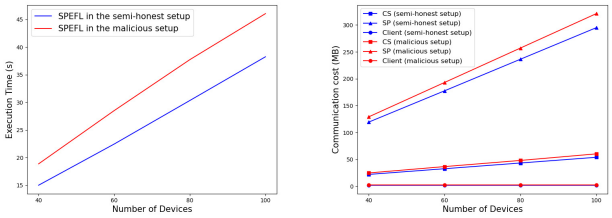
The last step in the real or simulated protocol is to perform the reconstruction of the output value y. In the simulation, y is the correct evaluation of the user input. We will prove that in the real world, the protocol can also correctly output the result y, otherwise the protocol will be aborted.

2) Correctness Analysis: Verifiable secret sharing $[a]$ is an extension of secret sharing $\langle a \rangle$. The operation is the same for secret shares $([a]_0, [a]_1)$ and $(\langle a \rangle_0, \langle a \rangle_1)$. Therefore, as long as the protocol is implemented correctly, the expected value will be correctly output. To complete the proof, we will show that the real protocol aborts when outputting an incorrect value with overwhelming probability.

Incorrect outputs are due to corrupted SPs that successfully cheated in the protocol, for which we verify the correctness of the outputs. In the verifying protocol, all parties will verify the correctness of the results according to Equations 9 and 10. Being able to cheat in rebuilding share $[y]$ means that an uncolluded malicious SP can correctly guess the global key α. Assuming α is chosen randomly in $\mathbb{Z}_{2^L}$, the probability is at most $1/|\mathbb{Z}_{2^L}|$. Since the protocol terminates as soon as a check fails, the probability that it outputs an error is the maximum probability $1/|\mathbb{Z}_{2^L}|$ that the check can be cheated. This is negligible since we assume that 2^L is exponential.

D. Experiments of the Verifiable Protocol

Fig. 7 shows the runtime and communication overhead of the SPEFL protocol under different settings. It can be observed that SPEFL enhances security at a relatively low cost. Specifically, the privacy-enhancing protocol increases the total running time by about **23%** and the communication overhead for **CS, SP, and clients by 12%, 8%, and 28%**, respectively. The main reason is that the additional operations of our security-enhanced protocol can be computed locally by the SP and the verification protocol is executed only once per iteration. Therefore, the additional overhead introduced by the security-enhanced protocol is limited.



(a) Execution time with Different Setup.

(b) Communication cost of CS, SP, and device with increasing number of devices.

Fig. 7. Execution Time and Communication cost of SPEFL on LeNet.

VII. CONCLUSION

In this paper, we propose SPEFL, a private, lightweight, and robust FL framework that preserves device privacy while maintaining robustness against poisoning attacks. SPEFL designs efficient cryptographic protocols to detect malicious devices using the three-server architecture. Most of the overhead introduced by privacy calculation and poisoning defense in SPEFL is borne by the servers, which is friendly to the resource-limited IoT devices. Moreover, the computation on the server does not involve high-overhead MPC primitives and can be executed efficiently. Furthermore, we design a verifiable FL protocol to guarantee the correctness of the aggregated results from the malicious server with low additional overhead. Experiments on different datasets and models demonstrate the advantages of our framework in performance. In future work, we will focus on the combination of more efficient privacy techniques and robust aggregation and seek privacy-robust FL in non-IID scenarios.

REFERENCES

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

[2] C. G. C. Index, "Forecast and methodology, 2016–2021 white paper," *Updated: February*, vol. 1, 2018.

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

[4] T. Yu, T. Li, Y. Sun, S. Nanda, V. Smith, V. Sekar, and S. Seshan, "Learning context-aware policies from multiple smart homes via federated multi-task learning," in *2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI)*. IEEE, 2020, pp. 104–115.

[5] M. Song, Z. Wang, Z. Zhang, Y. Song, Q. Wang, J. Ren, and H. Qi, "Analyzing user-level privacy attack against federated learning," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 10, pp. 2430–2444, 2020.

[6] A. Alwarafy, K. A. Al-Thelaya, M. Abdallah, J. Schneider, and M. Hamdi, "A survey on security and privacy issues in edge-computing-assisted internet of things," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4004–4022, 2020.

[7] J. Domingo-Ferrer, A. Blanco-Justicia, J. Manjón, and D. Sánchez, "Secure and privacy-preserving federated learning via co-utility," *IEEE Internet of Things Journal*, vol. 9, no. 5, pp. 3988–4000, 2021.

[8] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[9] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5650–5659.

[10] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *International Conference on Machine Learning*. PMLR, 2019, pp. 634–643.

[11] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 2938–2948.

[12] G. Sun, Y. Cong, J. Dong, Q. Wang, L. Lyu, and J. Liu, "Data poisoning attacks on federated machine learning," *IEEE Internet of Things Journal*, vol. 9, no. 13, pp. 11 365–11 375, 2021.

[13] W. Jiang, H. Li, S. Liu, X. Luo, and R. Lu, "Poisoning and evasion attacks against deep learning algorithms in autonomous vehicles," *IEEE transactions on vehicular technology*, vol. 69, no. 4, pp. 4439–4449, 2020.

[14] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[15] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients-how easy is it to break privacy in federated learning?" *Advances in Neural Information Processing Systems*, vol. 33, pp. 16 937–16 947, 2020.

[16] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the gan: information leakage from collaborative deep learning," in *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, 2017, pp. 603–618.

[17] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, "Beyond inferring class representatives: User-level privacy leakage from federated learning," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 2512–2520.

[18] X. Shen, Y. Liu, and Z. Zhang, "Performance-enhanced federated learning with differential privacy for internet of things," *IEEE Internet of Things Journal*, vol. 9, no. 23, pp. 24 079–24 094, 2022.

[19] W. Liu, X. Xu, D. Li, L. Qi, F. Dai, W. Dou, and Q. Ni, "Privacy preservation for federated learning with robust aggregation in edge computing," *IEEE Internet of Things Journal*, 2022.

[20] R. Guerraoui, S. Rouault *et al.*, "The hidden vulnerability of distributed learning in byzantium," in *International Conference on Machine Learning*. PMLR, 2018, pp. 3521–3530.

[21] L. Muñoz-González, K. T. Co, and E. C. Lupu, "Byzantine-robust federated machine learning through adaptive model averaging," *arXiv preprint arXiv:1909.05125*, 2019.

[22] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," in *Computer Security–ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14–18, 2020, Proceedings, Part I 25*. Springer, 2020, pp. 480–501.

[23] C. Fung, C. J. Yoon, and I. Beschastnikh, "The limitations of federated learning in sybil settings." in *RAID*, 2020, pp. 301–316.

[24] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "Fltrust: Byzantine-robust federated learning via trust bootstrapping," in *ISOC Network and Distributed System Security Symposium (NDSS)*, 2021.

[25] X. Liu, H. Li, G. Xu, Z. Chen, X. Huang, and R. Lu, "Privacy-enhanced federated learning against poisoning adversaries," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 4574–4588, 2021.

[26] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečnỳ, S. Mazzocchi, B. McMahan *et al.*, "Towards federated learning at scale: System design," *Proceedings of Machine Learning and Systems*, vol. 1, pp. 374–388, 2019.

[27] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[28] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[29] J. H. Bell, K. A. Bonawitz, A. Gascón, T. Lepoint, and M. Raykova, "Secure single-server aggregation with (poly) logarithmic overhead," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 1253–1269.

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

[31] L. He, S. P. Karimireddy, and M. Jaggi, "Secure byzantine-robust machine learning," *arXiv preprint arXiv:2006.04747*, 2020.

[32] J. So, B. Güler, and A. S. Avestimehr, "Byzantine-resilient secure federated learning," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 7, pp. 2168–2181, 2020.

[33] A. Imteaj, U. Thakker, S. Wang, J. Li, and M. H. Amini, "A survey on federated learning for resource-constrained iot devices," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 1–24, 2021.

[34] Y. Mao, X. Yuan, X. Zhao, and S. Zhong, "Romoa: Ro bust mo del a ggregation for the resistance of federated learning to model poisoning attacks," in *Computer Security–ESORICS 2021: 26th European Symposium on Research in Computer Security, Darmstadt, Germany, October 4–8, 2021, Proceedings, Part I 26*. Springer, 2021, pp. 476–496.

[35] Z. Zhang, X. Cao, J. Jia, and N. Z. Gong, "Fldetector: Defending federated learning against model poisoning attacks via detecting malicious clients," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 2545–2555.

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

[37] H. Lycklama, L. Burkhalter, A. Viand, N. Küchler, and A. Hithnawi, "Rofl: Robustness of secure federated learning," in *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2023, pp. 453–476.

[38] Y. Khazbak, T. Tan, and G. Cao, "Mlguard: Mitigating poisoning attacks in privacy preserving distributed collaborative learning," in *2020 29th International Conference on Computer Communications and Networks (ICCCN)*, 2020.

[39] M. Hao, H. Li, G. Xu, H. Chen, and T. Zhang, "Efficient, private and robust federated learning," in *Annual Computer Security Applications Conference*, 2021, pp. 45–60.

[40] M. Chase and S. S. Chow, "Improving privacy and security in multi-authority attribute-based encryption," in *Proceedings of the 16th ACM conference on Computer and communications security*, 2009, pp. 121–130.

[41] D. Demmler, T. Schneider, and M. Zohner, "{Ad-Hoc} secure {Two-Party} computation on mobile devices using hardware tokens," in *23rd USENIX Security Symposium (USENIX Security 14)*, 2014, pp. 893–908.

[42] R. Bahmani, M. Barbosa, F. Brasser, B. Portela, A.-R. Sadeghi, G. Scerri, and B. Warinschi, "Secure multiparty computation from sgx," in *International Conference on Financial Cryptography and Data Security*. Springer, 2017, pp. 477–497.

[43] S. Wagh, D. Gupta, and N. Chandran, "Securenn: 3-party secure computation for neural network training." *Proc. Priv. Enhancing Technol.*, vol. 2019, no. 3, pp. 26–49, 2019.

[44] L. Shen, X. Chen, J. Shi, Y. Dong, and B. Fang, "An efficient 3-party framework for privacy-preserving neural network inference," in *European Symposium on Research in Computer Security*. Springer, 2020, pp. 419–439.

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

[46] P. Mohassel and P. Rindal, "Aby3: A mixed protocol framework for machine learning," in *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, 2018, pp. 35–52.

[47] R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols," in *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*. IEEE, 2001, pp. 136–145.

[48] G. Sheng, T. Wen, Q. Guo, and Y. Yin, "Privacy preserving inner product of vectors in cloud computing," *International Journal of Distributed Sensor Networks*, vol. 10, no. 5, p. 537252, 2014.

[49] J. Zhang, X. Wang, S.-M. Yiu, Z. L. Jiang, and J. Li, "Secure dot product of outsourced encrypted vectors and its application to svm," in *Proceedings of the Fifth ACM International Workshop on Security in Cloud Computing*, 2017, pp. 75–82.

[50] Y. Aono, T. Hayashi, L. Wang, S. Moriai *et al.*, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1333–1345, 2017.

[51] C. Zhou, A. Fu, S. Yu, W. Yang, H. Wang, and Y. Zhang, "Privacy-preserving federated learning in fog computing," *IEEE Internet of Things Journal*, vol. 7, no. 11, pp. 10 782–10 793, 2020.

[52] D. Anguita, A. Ghio, L. Oneto, X. Parra, J. L. Reyes-Ortiz *et al.*, "A public domain dataset for human activity recognition using smartphones." in *Esann*, vol. 3, 2013, p. 3.

[53] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[54] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[55] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

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

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

[58] T. Gehlhar, F. Marx, T. Schneider, A. Suresh, T. Wehrle, and H. Yalame, "Safefl: Mpc-friendly framework for private and robust federated learning," in *2023 IEEE Security and Privacy Workshops (SPW)*, 2023, pp. 69–76.

[59] Z. Ghodsi, T. Gu, and S. Garg, "Safetynets: Verifiable execution of deep neural networks on an untrusted cloud," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[60] F. Tramer and D. Boneh, "Slalom: Fast, verifiable and private execution of neural networks in trusted hardware," *arXiv preprint arXiv:1806.03287*, 2018.

[61] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, "Verifynet: Secure and verifiable federated learning," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 911–926, 2019.

[62] Y. Ren, Y. Li, G. Feng, and X. Zhang, "Privacy-enhanced and verification-traceable aggregation for federated learning," *IEEE Internet of Things Journal*, vol. 9, no. 24, pp. 24 933–24 948, 2022.

[63] S. Qiu, Q. Liu, S. Zhou, and C. Wu, "Review of artificial intelligence adversarial attack and defense technologies," *Applied Sciences*, vol. 9, no. 5, p. 909, 2019.

[64] L. Lyu, H. Yu, X. Ma, C. Chen, L. Sun, J. Zhao, Q. Yang, and S. Y. Philip, "Privacy and robustness in federated learning: Attacks and defenses," *IEEE transactions on neural networks and learning systems*, 2022.

[65] I. Damgård, V. Pastro, N. Smart, and S. Zakarias, "Multiparty computation from somewhat homomorphic encryption," in *Annual Cryptology Conference*. Springer, 2012, pp. 643–662.

Liyan Shen received the Ph.D. degree from the University of Chinese Academy of Sciences, Beijing, China, in 2021.

She is currently a postdoctoral researcher at Beijing University of Posts and Telecommunications. Her research interests include security of machine learning and privacy-preserving computation.

Zhenhan Ke received the B.E. degree from Beijing University of Posts and Telecommunications, Beijing, China, in 2021. He is currently pursuing the M.Sc. degree with the School of Cyberspace Security, Beijing University of Posts and Telecommunications.

His research interests include federated learning and privacy-preserving computation.

Jinqiao Shi was born in 1978. He received his Ph.D. degree from the Harbin Institute of Technology, Harbin, China, in 2007.

He is a Professor and a Ph.D. Supervisor with the Beijing University of Posts and Telecommunications, Beijing, China. His current research interest includes network and information security, especially privacy-enhancing techniques and data leakage detection.

Dr. Shi is a member of the China Computer Federation.

Xi Zhang (Member, lEEE) received the Ph.D. degree in computer science from Tsinghua University, Beijing, China, in 2012.

He is a professor with the Beijing University of Posts and Telecommunications, and is also the vice Director of the Key Laboratory of Trustworthy Distributed Computing and Service, Ministry of Education, China. He was a visiting scholar at the University of Illinois. Chicago. His research interests include data mining and computer architecture.

Yanwei Sun received the Ph.D. degree in information security from the University of Chinese Academy of Sciences, Beijing, China, in 2018.

He is currently a lecturer of information security with the Beijing University of Posts and Telecommunications, Beijing, China. His current research interests include cyber situational awareness, cyber-attack attribution and threat intelligence operation.

Jiaopeng Zhao received the Ph.D. degree in information security from the University of Chinese Academy of Sciences, Beijing, China, in 2022.

He is currently a postdoctor of information security with Beijing University of Posts and Telecommunications. His current research interests include the dark web, network traffic analysis, cyberspace security and nature language processing.

Xuebin Wang received the Ph.D.degree in information security from the University of Chinese Academy of Sciences, Beijing, China, in 2022.

He is also with the Institute of Information Engineering, Chinese Academy of Sciences. His current research interests include information security and cryptocurrencies privacy.

Dr. Wang is a member of the China Computer Federation.

Xiaojie Zhao was born in China in 1997 and received her B.E. degree in Mathematics and Applied Mathematics from Shandong Technology and Business University, Shandong, China in 2020.

She is currently pursuing the Ph.D. degree in Cyberspace security professional, from the Beijing University of Posts and Telecommunications. Her research interests mainly include attack detection and federated learning.