Efficient and Secure Federated Learning Against Backdoor Attacks

Yinbin Miao, Rongpeng Xie, Xinghua Li, Zhiquan Liu, *Member, IEEE*, Kim-Kwang Raymond Choo, *Senior Member, IEEE*, and Robert H. Deng, *Fellow, IEEE*

***Abstract*—Due to the powerful representation ability and superior performance of Deep Neural Networks (DNN), Federated Learning (FL) based on DNN has attracted much attention from both academic and industrial fields. However, its transmitted plaintext data causes privacy disclosure. FL based on Local Differential Privacy (LDP) solutions can provide privacy protection to a certain extent, but these solutions still cannot achieve adaptive perturbation in DNN model. In addition, this kind of schemes cause high communication overheads due to the curse of dimensionality of DNN, and are naturally vulnerable to backdoor attacks due to the inherent distributed characteristic. To solve these issues, we propose an Efficient and Secure Federated Learning scheme (ESFL) against backdoor attacks by using adaptive LDP and compressive sensing. Formal security analysis proves that ESFL satisfies ϵ-LDP security. Extensive experiments using three datasets demonstrate that ESFL can solve the problems of traditional LDP-based FL schemes without a loss of model accuracy and efficiently resist the backdoor attacks.**

***Index Terms*—Adaptive local differential privacy, backdoor attacks, compressive sensing, federated learning.**

I. Introduction

As a distributed learning paradigm, Federated Learning (FL) [1] originally intends to prevent the direct disclosure of clients' raw data and solve the problem of data island. Different from the conventional centralized learning, FL delegates model training to clients who later upload local models or updates to the server. But the transmitted plaintext data still causes privacy leakage [2], [3]. In addition, due to the distributed nature, we cannot pledge that all clients are honest and their data is harmless. For example, a client may collect contaminated pictures from Internet as its local dataset for the training of image classifiers. Owing to less computation and communication overheads of Differential Privacy (DP), the DP-based FL solutions have been extensively explored [4], [5], [6], [7], [8], [9]. Among the above FL solutions, the Deep Neural Networks (DNN) is a commonly used machine learning model as it possesses powerful representation ability for complex nonlinear problems and superior performance. However, there are still some issues to be solved in previous DP-based FL schemes using DNN model structure.

Non-Adaptability of DP: The traditional DP mechanisms have a thorny problem that the sampled noise obeys a random distribution of noise source such as Gaussian noise, and we cannot control the noise level imposed on a single model parameter. There is a common situation that the excessive noise is added to the model parameter which is critical to the main task, resulting in serious damage to the model accuracy and making many schemes face a tradeoff between model accuracy and privacy protection. To solve the problem of excessive noise, an improved DP mechanism [9] achieves the so-called adaptive perturbation by exploring hierarchical characteristics of DNN to perturb the parameters of different layers to different degrees. Specifically, [9] perturbs the parameters layer by layer according to the range interval of parameters in each layer. However, the perturbation function used in the scheme just perturbs all parameters of the same layer to two fixed values, which does not achieve the real adaptability.

Dimension Disaster of DNN: Since the DNN model has the dilemma of high dimensions, the resource-limited clients such as mobile phones or wearable devices cannot afford huge communication overheads. A straightforward idea is to compress model, such as model pruning [10], tensor decomposition [11], weight quantization [12] and knowledge distillation [13]. Although those methods can be applied to reduce communication costs to some extent, they still either incur high burden on clients or cause loss of model accuracy. For example, model pruning consumes extra space to mark the location of pruned parameters, tensor decomposition involves decomposing a high-dimensional tensor into the sum of multiple low-rank tensors, which incur high storage and computation overheads respectively. The weight

Manuscript received 21 May 2023; revised 30 October 2023; accepted 12 January 2024. Date of publication 16 January 2024; date of current version 4 September 2024. This work was supported by the National Natural Science Foundation of China under Grant 62072361, in part by Key Research and Development Program of Shaanxi under Grant 2022GY-019, in part by Shaanxi Fundamental Science Research Project for Mathematics and Physics under Grant 22JSY019, in part by the National Natural Science Foundation of China under Grants 62125205, U23A20303, 61932010, and 62272195, in part by Fundamental Research Funds for the Central Universities under Grant QTZX23091, in part by Fellowship pf China Postdoctoral Science Foundation under Grant 2022T150507, in part by Opening Project of Intelligent Policing Key Laboratory of Sichuan Province under Grant ZNJW2023KFMS002. The work of Kim-Kwang Raymond Choo was supported only by the Cloud Technology Endowed Professorship. *(Corresponding author: Zhiquan Liu.)*

Yinbin Miao, Rongpeng Xie, and Xinghua Li are with the School of Cyber Engineering, Xidian University, Xian, Shaanxi 710071, China (e-mail: ybmiao@xidian.edu.cn; xierongpeng@stu.xidian.edu.cn; xhli1@mail.xidian.edu.cn).

Zhiquan Liu is with the College of Cyber Security, Jinan University, Guangzhou, Guangdong 510632, China, and also with Cyberdataforce (Beijing) Technology Ltd., Beijing 100020, China (e-mail: zqliu@vip.qq.com).

Kim-Kwang Raymond Choo is with the Department of Information Systems and Cyber Security, The University of Texas at San Antonio, San Antonio, TX 78249 USA (e-mail: raymond.choo@fulbrightmail.org).

Robert H. Deng is with the School of Information Systems, Singapore Management University, Singapore 178902 (e-mail: robertdeng@smu.edu.sg).

Digital Object Identifier 10.1109/TDSC.2024.3354736

quantization reduces the number of bits representing parameters, which leads to loss of model accuracy. And knowledge distillation trains a complicated teacher model with high time cost.

Backdoor Attacks Against FL: Due to the inherently distributed nature, FL faces various attacks caused by malicious clients or their adversarial manipulations, such as poisoning attacks where the adversary can arbitrarily manipulate the local models of malicious clients to corrupt the final global model. Among these attacks, we only focus on backdoor attacks, a subclass of poisoning attacks. The backdoor attacks inject almost negligible triggers into training data and alter corresponding labels to adversary-chosen ones, which make the model abnormal when it encounters data with the triggers. Most of existing defenses use the anomaly detection mechanisms to filter outliers [4], [14], [15], or design a variety of robust aggregation algorithms to alleviate the effect of the backdoor [16], [17], [18].

However, anomaly detection-based defenses require prior knowledge of adversary's attack strategy and distributions of clients' local datasets, which are strong assumptions and violate the fundamental premise of FL. For example, those solutions assume that the parameters of poisonous models are significantly greater than those of benign ones, which makes the poisonous ones to be easily detected and filtered. But the backdoored model outputs similar results with other benign models in most inputs, causing these methods invalid. Moreover, aggregation algorithms requiring trusted server have been proven ineffective in FL, in which the filtering steps fail to clean the backdoor [19]. Using the traditional DP can get a great defense effect, but it will lead to a sharp decline in the model utility.

Our Approach: To solve the above issues, we propose an Efficient and Secure Federated Learning scheme (ESFL) against backdoor attacks. In ESFL, we utilize the hierarchical characteristics of DNN [9], but more finely consider the value of each parameter and design the perturbation function to realize the real adaptability. Specifically, we first add Gaussian noise locally, and then use the adaptive perturbation function to regulate the added Gaussian noise, thereby alleviating the serious decline in model accuracy. In addition, ESFL exploits the Compressive Sensing (CS) [8], [20] to ease the uplink communication overheads of clients, which is a simple and fast method that includes both compression and decompression. Experiments have demonstrated that ESFL can indeed improve the model accuracy under a lower privacy budget, reduce the high communication costs and resist the backdoor attacks. The main contributions of this paper are summarized as follows:

- In the conference version of CAFL [21], we design a novel adaptive DP mechanism to implement the privacy-preserving FL, which improves the model utility under a lower privacy budget and adjusts the dilemma of excessive noise used in the traditional DP mechanisms, and apply CS to efficiently compress the local models to reduce high communication overheads, which accurately decompress the global model to guarantee the model utility.
- In this improved version of ESFL, we apply adaptive DP combined with Gaussian noise to resist backdoor attacks, which not only has the advantages of CAFL but also achieves a lower attack success rate.
- Our strict mathematical derivation proves that ESFL satisfies ϵ-LDP security. We also conduct extensive experiments to demonstrate that ESFL achieves the best model accuracy and lowest attack success rate using different datasets, when compared with previous state-of-the-art solutions.

The remainder of this paper is organized as follows: we begin with introducing some work related to this paper in Section II followed by preliminaries in Section III. Then we formalize the problems to be solved in this paper in Section IV. Our scheme's details will be described in Section V and we give the mathematical privacy analysis of our scheme in Section VI. We conduct extensive evaluations and show the results in Section VII. Finally, we conclude the paper in Section VIII.

II. Related Work

In this section, we will introduce some related work about privacy-preserving FL based on DP and defense measures against backdoor attack.

A. DP-Based Privacy-Preserving Federated Learning

As a lightweight privacy protection mechanism, DP can add noise to protect data without increasing the burden of resource-constrained clients, which has attracted much attention from both academic and industrial fields. For example, Geyer et al. countered differential attacks by adding Gaussian noise at the client level, hiding the contribution of the client to the global model [6]. But their goal is to deduce whether a client participates in training rather than to protect the training data itself. Seif et al. studied FL in wireless channel modeled by Gaussian multiple access channel, and realized private wireless gradient aggregation through random Gaussian noise [7]. Kerkouche et al. put forward FL-CS-DP by using compressive sensing and Gaussian noise. This scheme first compresses and clips the local model updates, then adds Gaussian noise, finally allows the server to aggregate the updates and restore the global model [8]. However, this scheme has a strong assumption that the model updates are sparse, which is difficult to meet in practice.

All the above schemes have achieved the privacy-preserving FL, but cause a loss of model accuracy due to a random distribution of noise source. To this end, Sun et al. proposed utilizing the hierarchy of DNN to design a semi-adaptive local DP mechanism, which can achieve privacy and high accuracy on complex models [9]. But this scheme perturbs the parameters of the same layer into two fixed values.

B. Backdoor Attack and its Defenses

The purpose of backdoor attack is to make model output the adversary-chosen target label to the input instance with triggers. Because of its strong concealment, many measures to resist backdoor attack have been put forward.

Regarding existing defenses against backdoor attack, they can be roughly divided into anomaly detection-based methods, methods to ensure the training integrity of participants, and robust aggregation methods. Cao et al. relied on the server training a root model to perform anomaly detection on the local models [14]. However, this assumption is too strong as the server is usually not trusted. Nguyen et al. proposed applying cosine similarity to calculate the distance between the clients, and used an expensive clustering algorithm to cluster the clients into malicious and non-malicious clusters [4]. Xu et al. utilized the concept of true discovery to dynamically calculate each client's trust score, and used this trust score as the weight assigned to each client during weighted aggregation [15]. In addition, when defending against backdoor attack, this scheme also applies the cosine similarity to filter the clients before calculating the trust score. Although the method of applying true discovery is novel, it will introduce high computation overheads.

Methods to ensure the training integrity of participants can reduce the anomalies caused by backdoor attacks by guaranteeing the full participation of local clients and the correctness of training process and results. Zhang et al. leveraged trusted execution environments and designed a commitment-based method with specific data selection to defend against various malicious attacks [22]. With the help of the idea of proof-of-work in blockchain, Jia et al. proposed a proof-of-learning method, which made malicious attackers spend a huge price to deceive the verifier [23]. However, this kind of methods have not been proved to guarantee security and robustness in ciphertext space, and they will have a huge overhead burden on the central server or local clients.

The robust aggregation algorithm is a defense from the perspective of the server, which implies the premise of a trusted server. Yin et al. proposed aggregating the local models at the dimension wise by taking the median and trimmed-mean [16], while Pillutla et al. took the median at the vector wise to aggregate the local models [17]. Blanchard et al. first calculated the euclidean distance between pairs of local models, then selected the nearest d distances and summed them as a score for each local model. It is worth noticing that the local model with the lowest score is selected as the updated global one [18]. This method of calculating the distance between two local models introduces the time complexity of $\mathcal{O}(n^2)$, and wastes the local resources as it only selects one local model to update the global one.

On the contrary, a fast DP mechanism can resist backdoor attack and protect privacy at the same time while saving computation overheads. Nguyen et al. added the defense mechanism of Gaussian noise apart from the filtering measure [4], which uses Centralized DP (CDP) deployed on server. As the server is not fully trusted, CDP will lead to an unrealistic assumption, and the traditional Local DP (LDP) seriously damages the model accuracy due to the random distribution of noise source. To this end, Naseri et al. tested the effectiveness of CDP and LDP against backdoor attack in FL framework [5]. A comparison between previous schemes and our scheme is summarized in Table I.

TABLE I
COMPARISON BETWEEN OUR SCHEME WITH PREVIOUS WORK

Schemes	Defense Mechanism	Backdoor Attack	Time Complexity	Adapt-ability	Hypothesis
[6]	CDP	$\sim$	$\mathcal{O}(n)$	✗	Trusted server
[7]	LDP	$\sim$	$\mathcal{O}(n)$	✗	Untrusted server
[8]	LDP	$\sim$	$\mathcal{O}(n)$	✗	Sparse updates
[9]	LDP	$\sim$	$\mathcal{O}(n)$	$*$	Untrusted server
[14]	Root model	✓	$\mathcal{O}(n^2)$	✗	Trusted server
[4]	Cosine similarity & CDP	✓	$\mathcal{O}(n^2)$ $\mathcal{O}(n)$	✗	Trusted server
[15]	True discovery	✓	$\mathcal{O}(n^2)$	✗	Trusted server
[16]	Dimension-wise robust aggregation	✗	$\mathcal{O}(n^2)$	✗	Trusted server
[17]	Vector-wise robust aggregation	✗	$\mathcal{O}(n^2)$	✗	Trusted server
[18]	Vector-wise robust aggregation	✗	$\mathcal{O}(n^2)$	✗	Trusted server
[5]	CDP & LDP	✓	$\mathcal{O}(n)$	✗	Trusted server Untrusted server
CAFL [21]	LDP	$*$	$\mathcal{O}(n)$	✓	Untrusted parties
ESFL	LDP	✓	$\mathcal{O}(n)$	✓	Untrusted parties

Notes: The symbol "✓" indicates that it owns this property; "✗" indicates that it does not own this property; "*" indicates that it has only part of this property; "$\sim$" indicates that it does not make comparisons.

III. PRELIMINARIES

In this section, we will introduce preliminary knowledge of related technologies, including federated learning, backdoor attacks, differential privacy and compressive sensing.

A. *Federated Learning*

As a distributed machine learning paradigm, FL links N clients $\{C_1, C_2, \ldots, C_N\}$ with a server to cooperatively train a model, in which each client $C_i (i \in [1, N])$ collects data as its private local dataset D_i. FL differs from traditional centralized machine learning in that the former assumes that D_i and $D_j (i \neq j)$ are Non-Independent and Identically Distributed (Non-IID). Without losing generality, a general FL problem can be formally described as the following optimization problem by (1),

$$\begin{cases} \arg\min\limits_{\mathbf{w}\in\mathbb{R}^n} \mathcal{L}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}_i(\mathbf{w}, X); \\ \mathcal{L}_i(\mathbf{w}, X) = \mathbb{E}_{x\sim\mathcal{P}_i}[\mathcal{L}_i(\mathbf{w}, x)], \end{cases} \tag{1}$$

where $\mathcal{L}_i(\mathbf{w}, X)$ is the local loss function of client C_i, $\mathbf{w}$ is the model, X represents the sample variable, x is a sample data in D_i, $\mathcal{P}_i$ denotes the data distribution of D_i such that $\mathcal{P}_i \neq \mathcal{P}_j$ for $i \neq j$, and the symbol $\mathbb{E}$ represents the mathematical expectation.

Suppose FL needs a total T of communication rounds, the following steps are iteratively performed between the clients and the server in each communication round $t \in [1, T]$.

- **Step-1**: Before starting round t, the server distributes the latest global model $\mathbf{w}^{t-1}$ to N_t random clients, where $N_t \leq N$ and the initial global model $\mathbf{w}^0$ is randomly initialized by the server.
- **Step-2**: Based on the local dataset D_i, each client C_i trains its local model $\mathbf{w}_i^t$ by $\mathbf{w}_i^t = \mathbf{w}_i^t - \eta \cdot \triangledown\mathcal{L}_i(\mathbf{w}_i^t, X)$, where $i \in [1, N_t]$, η is the local learning rate and $\triangledown\mathcal{L}_i(\mathbf{w}_i^t, X)$ is the gradient of its loss function $\mathcal{L}_i$.
- **Step-3**: After the local training, the local model updates $\{\Delta\mathbf{w}_i^t = \mathbf{w}_i^t - \mathbf{w}^{t-1}, i \in [1, N_t]\}$ are uploaded to the server which is responsible for aggregating and updating the global model $\mathbf{w}^t = \mathbf{w}^{t-1} + \frac{\zeta}{N} \cdot \sum_{i=1}^{N_t} \Delta\mathbf{w}_i^t$.[1]

[1]The global learning rate ζ controls the proportion of the joint model in each update. Especially, when $\zeta = \frac{N}{N_t}$, the aggregation rule is equivalent to FedAvg [1].

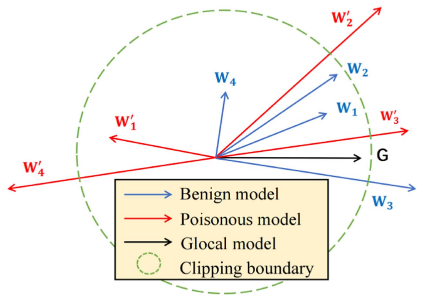


Fig. 1. Comparison of weight vectors between benign and poisonous models.

B. Backdoor Attacks

The poisoning attacks include data poisoning attacks [24], [25], [26], [27], model poisoning attacks [27], [28], [29], [30], [31], untargeted attacks [28], [29], [30], targeted attacks [24], [32] and backdoor attacks [25], [26], [33], [34]. The last one is a special case of data poisoning attacks and target attacks, which inject adversary-chosen (whether artificial or semantic) triggers into the training data and behaves abnormally when the model encounters data with the trigger. Moreover, the backdoor attacks can also be combined with model poisoning attacks to improve the attack accuracy. For example, the adversary can enlarge the weights of its poisonous model by (2),

$$\mathbf{w}_{adv}^t = \gamma \cdot \mathbf{w}_{adv}^t, \tag{2}$$

where the subscript adv and γ indicates the adversary and scaling factor, respectively.

Since we can regard a DNN model as a vector, the direction and magnitude between vectors can be distinct. Therefore, there are four differences between benign models and poisonous ones. First, the direction between the two models is different but the magnitude is analogous, such as the benign model $\mathbf{w}_1$ and the poisonous model $\mathbf{w}'_1$ in Fig. 1. Second, they are similar in direction but diverse in magnitude, such as the benign model $\mathbf{w}_2$ and the poisonous model $\mathbf{w}'_2$ in Fig. 1. The last two differences are that the direction and magnitude are similar, such as the benign model $\mathbf{w}_3$ and the poisonous model $\mathbf{w}'_3$ in Fig. 1, or poles apart, such as the benign model $\mathbf{w}_4$ and the poisonous model $\mathbf{w}'_4$. In addition, the backdoor attacks only misclassifies inputs with triggers and behaves as benign models for other inputs, so the adversary $\mathcal{A}$ will clip its poisonous model(s) to evade the anomaly detection and maximize the memory of its poisonous model(s) to the triggers.

C. Differential Privacy

Differential Privacy (DP) [35] is a de facto paradigm to protect data privacy by perturbing original data so that any adversary cannot infer the original private information from the obtained data. DP does not require special attack assumptions and is indifferent to the background knowledge of the attacker. Therefore, it can resist attacks in a more realistic scenario. The formal definition of local DP is as follows:

Definition 1 (Local differential privacy (LDP)): Let $\mathcal{M}$ be a randomized perturbation mechanism, for any pair input x and x' in D and any output Y of $\mathcal{M}$, $\mathcal{M}$ satisfies ϵ-LDP such that

$$Pr[\mathcal{M}(x) = Y] \leq e^{\epsilon} \cdot Pr[\mathcal{M}(x') = Y],$$

where ϵ is the privacy budget of $\mathcal{M}$. A smaller value for ϵ indicates a stronger privacy guarantee, but also means lower data availability [9].

D. Compressive Sensing

Compressive Sensing (CS) aims to compress a signal during sampling and the original information is recovered from the sampled fewer measurements [8].

Given a signal $\mathbf{x} \in \mathbb{R}^n$, we want to sample it as a compressed signal $\mathbf{y} \in \mathbb{R}^m$, where $m < n$. This can be easily achieved by matrix multiplication as $\mathbf{y} = \mathbf{\Phi x}$, where $\mathbf{\Phi} \in \mathbb{R}^{m \times n}$ is the measurement matrix. And the compression ratio is denoted as $CR = \frac{m}{n}$.

However, we cannot directly solve $\mathbf{x}$ from $\mathbf{y}$ and $\mathbf{\Phi}$ unless that the original signal $\mathbf{x}$ itself is sparse. That is, $\mathbf{x}$ only has a few non-zero values. So we need to sparse $\mathbf{x}$ first and get its sparse representation $\mathbf{s} \in \mathbb{R}^n$ as $\mathbf{x} = \mathbf{\Psi s}$, where $\mathbf{\Psi} \in \mathbb{R}^{n \times n}$ is the sparsity orthonormal basis matrix. Note that if $\mathbf{\Psi}$ is an identity matrix, then $\mathbf{s}$ is equal to $\mathbf{x}$, which means that $\mathbf{x}$ itself is sparse. Therefore, we can get

$$\mathcal{C}(\mathbf{x}, m) = \mathbf{y} = \mathbf{\Phi x} = \mathbf{\Phi \Psi s} = \mathbf{\Theta s}, \tag{3}$$

where $\mathcal{C}$ indicates compression operation symbol and $\mathbf{\Theta}$ is the sparsity sensing matrix.

Note that the compression operator $\mathcal{C}$ in (3) is linear, which means that

$$\begin{cases} \sum_{k=1}^{K} \mathcal{C}(\mathbf{x}_k, m) = \mathcal{C}(\sum_{k=1}^{K} \mathbf{x}_k, m); \\ \mathcal{D}(\sum_{k=1}^{K} \mathcal{C}(\mathbf{x}_k, m)) \approx \sum_{k=1}^{K} \mathbf{x}_k, \end{cases} \tag{4}$$

where $\mathcal{D}$ indicates decompression operation symbol. This linearity guarantees that we can first aggregate the compressed local models to obtain a compressed global model easily, then decompress at one time to obtain a complete global model, instead of decompressing multiple incomplete local models for multiple times before aggregation, which saves the calculation time.

IV. System and Problem Setting

In this section, we will formally describe the problem in this paper from the aspects of system model, threat model, problem definition and design goals.

A. System Model

Our system model is shown in Fig. 2, and consists of a server and a total of N clients, among which the $P_{\mathcal{A}} \in [0, 0.5]$ proportion of N clients are compromised by the adversary $\mathcal{A}$ and become malicious.

- The server acts as an aggregator, which is responsible for collecting N_t local model updates and aggregating them into a global model update to optimize the global model. In our scheme, it also needs to perform decompression operation.

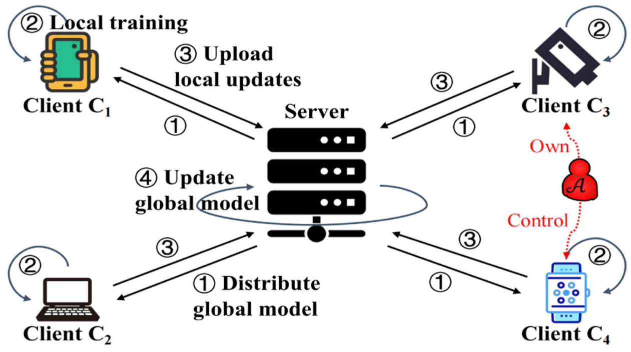


Fig. 2. System model of our scheme.

– The non-malicious clients are isolated devices, which are in charge of performing local model training on clean datasets, compression and noise addition.
– The malicious clients will conduct local model training on poisonous datasets with their chosen triggers, then compress and amplify model updates before uploading. Note that they do not add noise.

B. Threat Model

In our scheme, we assume that the server is semi-honest which faithfully aggregates all uploaded local model updates without deliberately dropping anyone into a global model update, but it can invade privacy only by tracking the local and global model updates in every communication round to infer additional information about any client. The clients are divided into two categories: non-malicious and malicious. The former honestly adheres to algorithms, but infers the private information of their chosen victims from the global model. The latter injects or alters instances with crafted triggers into their datasets, and trains poisonous models with backdoor. Therefore, the threats of our scheme can be summarized as follows:

- *Privacy leakage of clients data:* If the model updates are transmitted in plaintext form, semi-honest server and non-malicious clients can infer private information about the training data.
- *Backdoor attacks:* For malicious clients, they will launch backdoor attack on the model training process to make the model misclassify the input with trigger.
- *Model utility reduction:* The backdoor attacks can cause a portion of test samples to be misclassified, which impairs the model utility.

C. Problem Definition

1) Local Differential Privacy and Compression: In the traditional DP-based FL schemes resisting backdoor attack, the weight w of local model is perturbed as follows,

$$w^* = w + \delta \quad \text{s.t.} \quad \delta \sim \mathcal{N}(0, \sigma^2), \tag{5}$$

where w^* is the perturbed weight and δ is the random noise sampled from the Gaussian distribution $\mathcal{N}(0, \sigma^2)$. Then, the server aggregates the noisy local model updates into a latest noisy global model update by (6),

$$\sum_{i=1}^{N_t} \Delta \mathbf{w}_i^{t*}. \tag{6}$$

However, the noise δ is randomly sampled from the Gaussian distribution $\mathcal{N}(0, \sigma^2)$, which makes its magnitude not be strictly controlled and non-adaptive.

For adaptive perturbation, Sun et al. perturbed local model weights hierarchically [9]. Specifically, their scheme first calculates the range interval $[c_l - r_l, c_l + r_l]$ of the weights of the l-th layer, where c_l represents the range center and r_l represents the range radius. Then it perturbs the weight according to the privacy budget ϵ, range center c_l and range radius r_l as follows,

$$\mathcal{M}(w) = w^*$$
$$= \begin{cases} c_l + r_l \cdot \frac{e^\epsilon+1}{e^\epsilon-1}, \text{w.p.} \frac{(w-c_l)(e^\epsilon-1)+r_l(e^\epsilon+1)}{2r_l(e^\epsilon+1)}; \\ c_l - r_l \cdot \frac{e^\epsilon+1}{e^\epsilon-1}, \text{w.p.} \frac{-(w-c_l)(e^\epsilon-1)+r_l(e^\epsilon+1)}{2r_l(e^\epsilon+1)}, \end{cases} \tag{7}$$

where $\mathcal{M}$ the perturbation function and "w.p." stands for "with probability". However, the range center c_l and range radius r_l are uniform for all weights in the same layer l. In addition, the product factor $\frac{e^\epsilon+1}{e^\epsilon-1}$ is also identical in (7). Thus, this scheme results in the weights of the same layer being perturbed to only two values, which cannot facilitate the real adaptability.

Besides, the high dimension of DNN model not only causes high communication overheads but also increases the computation overheads of clients. However, the existing compression methods either are inefficient or lead to a loss of accuracy. Thus, a method of integrating efficient compression and accurate decompression needs to be considered.

2) Defense Against Backdoor Attacks: Following the principle of backdoor attacks, the adversary $\mathcal{A}$ first injects triggers into the private datasets of the malicious clients, and then loyally executes the training algorithm as the normal clients to obtain its poisonous local models. To enhance the memory of poisonous models for triggers and to resist the weakening of the aggregation on the server, the adversary $\mathcal{A}$ will amplify the weights of its poisonous models by (2).

Then, to avoid anomaly detection and keep the memory of triggers to the maximum extent, the adversary $\mathcal{A}$ will clip the weights of its poisonous models as follows,

$$\mathbf{w}_i^t = \min\left(1, \frac{S}{||\mathbf{w}_i^t||_2}\right) \cdot \mathbf{w}_i^t, \tag{8}$$

where S is the clipping boundary and the symbol $||\cdot||_2$ represents the l_2 norm.

In our scheme, we use the local differential privacy mechanism to encrypt local models, which makes the anomaly detection-based defense methods ineffective. Formally, we use δ_G to represent the traditional Gaussian noise and δ_{ada} to represent the adaptive noise, so we have the following defense formula:

$$\mathbf{w}_i^t = \mathbf{w}_i^t + \delta_G + \delta_{ada}. \tag{9}$$

Note that we proceed with δ_G first and then with δ_{ada}.

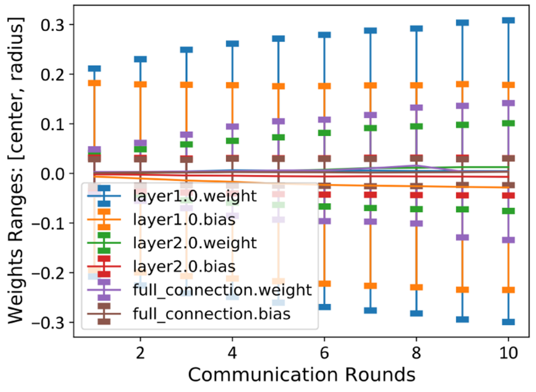
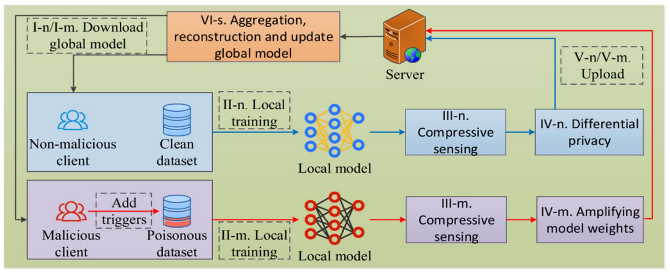


Fig. 3. Weights changes in different layers of the DNN model.



I-n/I-m. Download global model
VI-s. Aggregation, reconstruction and update global model
Server
V-n/V-m. Upload
Non-malicious client
Clean dataset
II-n. Local training
Local model
III-n. Compressive sensing
IV-n. Differential privacy
Malicious client
Add triggers
Poisonous dataset
II-m. Local training
Local model
III-m. Compressive sensing
IV-m. Amplifying model weights

Fig. 4. Overview of our ESFL.

D. Design Goals

Inspired by previous work [5], [8], [9], we aim to design a backdoor-resistant and privacy-preserving FL with high model accuracy and low communication overheads. Based on these, our scheme should achieve the following goals:

1) *High model precision:* Our scheme should maintain high model accuracy after successfully implementing privacy protection and resisting backdoor attacks by using DP.
2) *Mitigate the curse of dimensionality in DNN:* Since our scheme is a model designed under the DNN structure, we should alleviate the large amount of traffic caused by high dimensions.
3) *Reduction of backdoor attacks success rate:* Our scheme should not output the adversary-expected target label when the final global model faces a testing input instance with an adversary-chosen trigger.

V. ESFL

In this section, we first introduce the technical overview of our scheme, then formally and completely describe the workflow of our scheme.

A. Technical Overview

In our ESFL, we will compress the local models before perturbation. This is because our adaptive perturbation function is done sequentially for individual weights, which saves computation overheads for clients. Specifically, we first use the Discrete Cosine Transform (DCT) to change the form of original weights, then employ the compression algorithm of CS to compress local models from the original total n weights to m via a simple and fast matrix multiplication, which mitigates the uplink communication overheads, where $m < n$.

After compression, we start to perturb the model weights. Our intuition is that since the different layers of DNN process different information, the weights in different layers should also have different range intervals. We empirically demonstrate the intuition through experiments, referring to Fig. 3, where the range intervals of different layers are different and variational. Similar to scheme [9], we assume that the DNN model has a total of L layers, in which each layer $l \in [1, L]$ will be in a range interval $[c_l - r_l, c_l + r_l]$. The perturbation function $\mathcal{M}$ of ESFL involves the offset μ of each weight from its range center c_l rather than the range radius r_l, where $\mu = w - c_l$. Thus, ESFL achieves finer-grained noise addition, thereby minimizing noise, improving model accuracy and realizing the real adaptability. Referring to (7), the independent variables involved in the perturbation function $\mathcal{M}$ of scheme [9] are range center c_l, range radius r_l and privacy budget ϵ, while ours are range center c_l, offset μ and privacy budget ϵ.

Considering that the adaptive perturbation can control the scale of the noise, we intend to utilize this method to adjust the traditional Gaussian noise whose effectiveness has been empirically proven. Specifically, we must first add Gaussian noise and then adaptive noise, not in reverse order. Throughout experiments, we know that if the noise is added in the opposite order, it will seriously damage the final model utility even though it can well defend against the backdoor attacks.

Finally, we consider the synchronous scenario where the server will wait for enough N_t local model updates to be uploaded before starting the aggregation. Since CS is linear, we can aggregate first and then decompress at one time. Specifically, after obtaining the aggregation result of the t-th round, the server performs inverse discrete cosine transform on the incomplete global model update containing m weights to recover the original data form, then restores the complete global model update containing n weights through the decompression algorithm of CS, and finally updates the global model.

B. Full Flow of ESFL

Referring to Fig. 4, each selected client $C_i (i \in [1, N_t])$ first downloads the latest complete global model $\mathbf{w}^{t-1}$ as its local model $\mathbf{w}_i^t$ (step I). For non-malicious clients, they faithfully perform local training (step II-n) on their clean datasets, while for malicious clients, they launch backdoor attacks (step II-m) by adding triggers into datasets. Then, for all N_t clients, they must perform the compression (step III-n/III-m). After that, the non-malicious clients will perform an additional phase of adding noise to resist attacks (step IV-n) before uploading the model updates (step V-n), while malicious clients will amplifying model updates for strengthening the attacks effect (step IV-m) before uploading (step V-m). By exploiting the linear advantage of compressive sensing, the server aggregates local updates first,

Algorithm 1: Local Training with SGD.

```
Input: Latest complete global model $\mathbf{w}^{t-1}$; total local epochs $E$; total batches $B$; learning rate $\eta$
Output: Trained local model $\mathbf{w}_i^t$
Client $C_i$ ($i \in [1, N_t]$):
1: $\mathbf{w}_i^t \leftarrow \mathbf{w}^{t-1}$;
2: for each local epoch $e = 1, 2, \ldots, E$ do
3:    for each mini batch $b \in B$ do
4:       $\mathbf{w}_i^t = \mathbf{w}_i^t - \eta \bigtriangledown \mathcal{L}_i(\mathbf{w}_i^t, b)$;
5:    end for
6: end for
7: if $C_i$ is malicious then
8:    $\mathbf{w}_i^t = \gamma \cdot \mathbf{w}_i^t$;
9:    $\mathbf{w}_i^t = \min(1, \frac{S}{||\mathbf{w}_i^t||_2}) \cdot \mathbf{w}_i^t$;
10: end if
11: Return $\Delta \mathbf{w}_i^t = \mathbf{w}_i^t - \mathbf{w}^{t-1}$.
```

then reconstructs global update, and finally updates the global model (step VI-s).

1) Server-Side Initialization: Before formally starting the model training, we require the server to initialize a global model which is then distributed to each client. In this paper, since we use DNN as model structure and exploit its hierarchical characteristics to design the adaptive perturbation function $\mathcal{M}$, we can make the server negotiate with all N clients relying on their prior knowledge or rich experience. Specifically, for each layer $l \in [1, L]$, each client participates in the negotiation of an initial range interval $[c_l - r_l, c_l + r_l]$ and a set pair (C, R), where C represents a set containing L range centers and R represents a set containing L corresponding range radii. Finally, the server initializes the global model $\mathbf{w}^0$ according to (C, R) and issues to random N_1 clients participating in the first round of communication.

2) Local Training: At this stage, both malicious (if they are selected in N_t) and non-malicious clients will honestly implement the optimization algorithm of the model.

In Algorithm 1, the client C_i first replaces its local model $\mathbf{w}_i^{t-1}$ of the previous round with the latest complete global model $\mathbf{w}^{t-1}$ (line 1), and then performs local iterations multiple epochs to adequately train its local model $\mathbf{w}_i^t$ (lines 2-6). In each epoch $e \in [1, E]$, the client C_i samples a mini batch b from its dataset D_i and puts it into the model to speed up the local training, where b is a user-defined super-parameter and B is the total batches whose number is equal to $|D_i|/b$. Note that $\bigtriangledown \mathcal{L}_i(\mathbf{w}_i^t, b)$ represents the gradient of the loss function $\mathcal{L}_i$ with respect to the mini batch b.

If there are malicious clients in this round, they will amplify their local models to enhance the ability of classifying poisonous instances (line 8). To further reduce the difference between malicious clients and other non-malicious ones, they also need to clip their enlarged models (line 9).

3) Local Compression: In the preliminary knowledge of CS in Section III-D, we define the notion of compression ratio CR, which is conducive to achieve our compression purpose by

Algorithm 2: Local Compression.

```
Input: Trained local model $\mathbf{w}_i^t$; total layers $L$; compression ratio $CR$
Output: Trained and compressed local model update $\mathbf{c}_i^t$
Client $C_i$ ($i \in [1, N_t]$):
1: $\mathbf{c}_i^t = \Delta \mathbf{w}_i^t$;
2: for each layer $l \in [1, L]$ in $\mathbf{c}_i^t$ do
3:    $n_l = \text{len}(\mathbf{c}_{i,l}^t)$; // the number of weights for this layer
4:    $m_l = n_l \times CR$;
5:    $\mathbf{c}_{i,l}^t = \mathcal{C}(\mathbf{c}_{i,l}^t, m_l)$; // including DCT and compression
6: end for
7: Return $\mathbf{c}_i^t$.
```

setting any $CR \in (0, 1)$. Since the subsequent adaptive perturbation stage is performed hierarchically, we will also compress $\Delta \mathbf{w}_i^t$ hierarchically, instead of flattening total n weights of $\Delta \mathbf{w}_i^t$ into an n-dimensional vector $\mathbf{x}^n$ and then compressing it into only m-dimensional vector $\mathbf{y}^m (m < n)$. Specifically, we first acquire the initial number n_l of weights of the l-th layer by using a function len $(\cdot)$ (line 3), then calculate the compressed number m_l of weights of this layer based on CR (line 4), finally perform compression after the transformation by DCT (line 5).

In Algorithm 2, we represent the trained and compressed local model update with the symbol $\mathbf{c}_i^t$ (except for line 1 for obvious reasons) and the symbol $\mathbf{c}_{i,l}^t$ for the l-th layer.

4) Gaussian Noise Addition and Adaptive Perturbation: After the non-malicious client C_i obtains its $\mathbf{c}_i^t$, it will add noise into the model update to resist backdoor attack. In Algorithm 3, let $\mathbf{p}_i^t$ be the compressed perturbed local model update and $\mathbf{p}_{i,l}^t$ be weights in the l-th layer.

For Gaussian noise δ_G, the distribution of noise source is $\mathcal{N}(0, \sigma^2)$, where the sampled δ_G from it is random. Note that when adding noise to the weights, we should do it one by one, but we describe Gaussian noise addition layer by layer in Algorithm 3 for simplicity (line 3), which is different from the adaptive noise addition (lines 7-11).

For adaptive noise δ_{ada}, the non-malicious client C_i first needs to calculate the range interval $[c_l - r_l, c_l + r_l]$ of each layer $l \in [1, L]$. Specifically, the non-malicious client C_i acquires the maximum weight $w_l^{\max}$ and the minimum weight $w_l^{\min}$ for the l-th layer by using functions $\max(\cdot)$ and $\min(\cdot)$ respectively (lines 4-5), then obtains the range center and radius of this layer by (10) (line 6),

$$\begin{cases} c_l = \frac{(w_l^{\max} + w_l^{\min})}{2}; \\ r_l = w_l^{\max} - c_l. \end{cases} \tag{10}$$

Next, the non-malicious client C_i calculates the offset μ of each weight w from the range center c_l as follows (line 8),

$$\mu = w - c_l. \tag{11}$$

Note that the calculation method of offset μ does not take absolute value, but that of range radius r_l ensures that it is always non-negative. The reason why we do this will be explained in detail after giving the specific form of the adaptive perturbation function $\mathcal{M}$.

Let's rewrite (11) to get the expression for the weight w, i.e., $w = c_l + \mu$. For all weights located at a certain layer l, the range center c_l is the same and fixed. Thus, the difference between the weights is only reflected in the offset μ, which deduces that the perturbation of weight w is essentially the perturbation of offset μ.

Similar to scheme [9], we design the perturbation function $\mathcal{M}$ as a piecewise function which perturbs different weights in various ways according to different conditions. Concretely, we first calculate the distribution probability Pr of Bernoulli variable b based on the privacy budget ϵ by (12),

$$\begin{cases} Pr[b=1] = \frac{e^{\epsilon}-1}{2e^{\epsilon}}; \\ Pr[b=0] = \frac{e^{\epsilon}+1}{2e^{\epsilon}}. \end{cases} \tag{12}$$

Then, the non-malicious client C_i randomly samples from the Bernoulli distribution satisfying the above probability, and perturbs the weight w as follows,

$$\mathcal{M}(w) = w^* = \begin{cases} c_l + \mu \cdot \frac{e^{\epsilon}+1}{e^{\epsilon}-1}, & \text{if } b=1; \\ c_l + \mu \cdot \frac{e^{\epsilon}-1}{e^{\epsilon}+1}, & \text{if } b=0. \end{cases} \tag{13}$$

Note that the product factors of the offsets are reciprocal, which helps to further reduce the degree of perturbation as $\frac{e^{\epsilon}-1}{e^{\epsilon}+1} < \frac{e^{\epsilon}+1}{e^{\epsilon}-1}$. As can be seen from (13), our adaptive perturbation is indeed perturbing the offset μ in essence. Below we give the detailed improvements of our scheme:

1) As mentioned earlier, the offset μ does not take the absolute value, which naturally implies that the sign does not need the explicit explanation as in the scheme [9].
2) Comparing (10) with (11), we can find that the range radius r_l is always not less than any offset μ. Therefore, our perturbation on μ is limited to a smaller scope, which reduces the amount of noise and thus improves the model utility.
3) We flip the product factor so that the perturbation function $\mathcal{M}$ cannot always change the weight with an increasing tendency, which weakens the weights amplified by the adversary $\mathcal{A}$.

5) Aggregation and Decompression: After the non-malicious client C_i performs local training, compression and noise adding, it uploads the compressed perturbed local model update $\mathbf{p}_i^t$ to the server. Note that the malicious client will not add noise to its $\mathbf{c}_i^t$. Thus, for ease of expression, we still $\mathbf{p}_i^t$ to represent the compressed non-perturbed local model update of the malicious client.

Because we are considering FL in the synchronous scenario, the server will wait until a total of N_t local model updates from different clients are collected before starting the aggregation. In Section III-D, we mention that CS is linear, which facilitates the server to aggregate first and then decompress once. Therefore, the aggregated but compressed global model update is $\mathbf{y}^t = \sum_{i=1}^{N_t} \mathbf{p}_i^t$.

Following the basic principles of local compression and noise addition, the server decompresses $\mathbf{y}^t$ hierarchically. In Algorithm 4, we use $\mathbf{y}_{*,l}^t$ to represent weights of the l-th layer for $\mathbf{y}^t$, $\Delta\mathbf{w}_{*,l}^t$ to represent updates of the l-th layer for the final complete global model update $\Delta\mathbf{w}^t$. Specifically, the server first calculates the compressed number m_l of weights in the l-th layer (line 3), then gets the original number n_l according to the compression ratio CR (line 4), finally decompresses $\mathbf{y}_{*,l}^t$ after inverse discrete cosine transform (line 5). After getting L reconstructed $\Delta\mathbf{w}_{*,l}^t$, the server assembles them into a complete global model update $\Delta\mathbf{w}^t$ (line 7) and updates the global model $\mathbf{w}^t = \mathbf{w}^{t-1} + \frac{\zeta}{N} \cdot \Delta\mathbf{w}^t$ (line 8).

After the server obtains the latest complete global model $\mathbf{w}^t$, it will repeat all the processes except the above-mentioned initialization phase.There are generally two conditions for stopping communication. (1) After the server aggregates the global model $\mathbf{w}^t$, we can test its accuracy on the main task. If the accuracy is kept within a certain accuracy range for several consecutive rounds, it is considered that the model training has reached convergence. (2) The communication will continue until the number of communication rounds increases to the preset

Algorithm 3: Gaussian Noise Addition and Adaptive Perturbation.

```
Input: Trained and compressed local model update $\mathbf{c}_i^t$
Output: Trained, compressed and perturbed local model
  update $\mathbf{p}_i^t$
Non − malicious Client $C_i$:
1: $\mathbf{p}_i^t = \mathbf{c}_i^t$
2: for each layer $l \in [1, L]$ in $\mathbf{p}_i^t$ do
3:    $\mathbf{p}_{i,l}^t = \mathbf{p}_{i,l}^t + \mathcal{N}(0, \sigma^2)$;
4:    $w_l^{max} = \max(\mathbf{p}_{i,l}^t)$;
5:    $w_l^{min} = \min(\mathbf{p}_{i,l}^t)$;
6:    Calculate the range center $c_l$ by (10);
7:    for each weight $w \in \mathbf{p}_{i,l}^t$ do
8:       Calculate the offset $\mu$ by (11);
9:       Randomly sample a Bernoulli variable $b$ according
         to the Bernoulli sampling probability in (12);
10:      Perturb the weight $w$ by (13);
11:   end for
12: end for
13: Return $\mathbf{p}_i^t$.
```

Algorithm 4: Aggregation and Decompression.

```
Input: $N_t$ local model updates $\{\mathbf{p}_1^t, \mathbf{p}_2^t, \ldots, \mathbf{p}_{N_t}^t\}$;
  compression ratio $CR$
Output: Complete global model $\mathbf{w}^t$
Server:
1: Aggregate $N_t$ local model updates into a compressed
   global model update $\mathbf{y}^t$;
2: for each $l \in (i \in [1, L])$ in $\mathbf{y}^t$ do
3:    $m_l = len(\mathbf{y}_{*,l}^t)$;
4:    $n_l = \frac{m_l}{CR}$;
5:    $\Delta\mathbf{w}_{*,l}^t = \mathcal{D}(\mathbf{y}_{*,l}^t, n_l)$;
6: end for
7: $\Delta\mathbf{w}^t = assemble(\Delta\mathbf{w}_{*,1}^t, \Delta\mathbf{w}_{*,2}^t, \ldots, \Delta\mathbf{w}_{*,L}^t)$;
8: update the global model $\mathbf{w}^t = \mathbf{w}^{t-1} + \frac{\zeta}{N} \cdot \Delta\mathbf{w}^t$
9: Return $\mathbf{w}^t$.
```

Algorithm 5: The Complete Algorithm Process of ESFL.

```
Server:
1: Initialize a global model $\mathbf{w}^0 \in \mathbb{R}^n$ based on $(C, R)$,
   where $C = \{c_1, c_2, \dots, c_L\}$ and $R = \{r_1, r_2, \dots, r_L\}$;
2: for each round $t = 1$ to $T$ do
3:     Select $N_t$ clients randomly in total $N$ clients;
4:     for each client $C_i (i \in [1, N_t])$ in parallel do
5:         $\mathbf{p}_i^t$ = Client $C_i$ $(\mathbf{w}^{t-1})$;
6:     end for
7:     //Aggregation and decompression
8:     Call Algorithm 4 to generate $\mathbf{w}^t$;
9: end for
10: Output $\mathbf{w}^T$
Client $C_i$ $(\mathbf{w}^{t-1})$:
1: //Local training
2: Call Algorithm 1 to generate $\mathbf{w}_i^t$;
3: //Local compression
4: Call Algorithm 2 to generate $\mathbf{c}_i^t$;
5: //Gaussian noise addition and adaptive perturbation
6: if $C_i$ is non-malicious then
7:     Call Algorithm 3 to generate $\mathbf{p}_i^t$;
8: end if
9: Return $\mathbf{p}_i^t$.
```

maximum value T. For simplicity, we adopt the second method in the complete Algorithm 5.

VI. Privacy Analysis

In this section, we will formally analyze the privacy guarantee and utility cost of our adaptive perturbation mechanism.

First, according to Theorem 1, our scheme satisfies the definition of LDP under the given conditions.

Theorem 1: Given an arbitrary weight $w \in [c_l - r_l, c_l + r_l]$ and the privacy budget $\epsilon \geq \ln(1 + \sqrt{2})$, the perturbation mechanism $\mathcal{M}$ proposed in Eq.13 satisfies ϵ-LDP.

Proof: Suppose the perturbed weight $w^* = c_l + \mu \cdot \frac{e^\epsilon + 1}{e^\epsilon - 1}$, the probability of a Bernoulli variable b taking 1 is equal to $\frac{e^\epsilon - 1}{2e^\epsilon} < \frac{1}{2}$. For any $w, w' \in [c_l - r_l, c_l + r_l]$, we get

$$\frac{Pr[\mathcal{M}(w) = w^*]}{Pr[\mathcal{M}(w') = w^*]} \leq \frac{\max_w Pr[\mathcal{M}(w) = w^*]}{\min_{w'} Pr[\mathcal{M}(w') = w^*]} = \frac{1 - \frac{e^\epsilon - 1}{2e^\epsilon}}{\frac{e^\epsilon - 1}{2e^\epsilon}} = \frac{e^\epsilon + 1}{e^\epsilon - 1}. \tag{14}$$

According to the Definition 1 of LDP, we expect the above expression to satisfy the constraint: $\frac{e^\epsilon + 1}{e^\epsilon - 1} \leq e^\epsilon$. Therefore, we can get a threshold $\epsilon_0 = \ln(1 + \sqrt{2})$. This allows the Definition 1 to be satisfied only if the privacy budget satisfies $\epsilon \geq \epsilon_0$. □

Second, we will prove that the adaptive perturbation mechanism in Algorithm 3 estimates the average weights with zero bias.

Theorem 2: The adaptive perturbation mechanism introduces zero bias to estimating average weights, i.e., $\mathbb{E}[\overline{\mathcal{M}(w)}] = \overline{w}$.

Proof: For any weight w from any client C_i and its corresponding c_l and μ,

$$\mathbb{E}[\mathcal{M}(w)] = \left(c_l + \mu \cdot \frac{e^\epsilon + 1}{e^\epsilon - 1}\right) \cdot \frac{e^\epsilon - 1}{2e^\epsilon} + \left(c_l + \mu \cdot \frac{e^\epsilon - 1}{e^\epsilon + 1}\right) \cdot \left(1 - \frac{e^\epsilon - 1}{2e^\epsilon}\right) = c_l + \mu = w; \tag{15}$$

$$\mathbb{E}[\overline{\mathcal{M}(w)}] = \mathbb{E}\left[\frac{1}{N_t}\sum_{i=1}^{N_t} \mathcal{M}(w)\right] = \frac{1}{N_t}\sum_{i=1}^{N_t} \mathbb{E}[\mathcal{M}(w)] = \frac{1}{N_t}\sum_{i=1}^{N_t} w = \overline{w}. \tag{16}$$

□

Theorem 3: Algorithm 3 can achieve the same security performance as the traditional DP mechanism based on Gaussian noise.

Proof: Algorithm 3 stipulates that Gaussian noise must be added first and then adaptive noise be added. Theorem 2 indicates that the adaptive perturbation mechanism introduces zero bias to evaluating the average weights, which shows that the security achieved by introducing Gaussian noise will not be destroyed by the adaptive perturbation mechanism. □

In Algorithm 3, since the center c_l has no product factor associated with it, we can equivalently regard it as a perturbation to the offset μ which is perturbed into μ^*. Therefore, we have the following Theorem 4.

Theorem 4: Given any weight w, the variance of the perturbation function is $Var[\mathcal{M}(w)] = \frac{4r^2}{e^{2\epsilon} - 1}$.

Proof: Given any parameter w, the variance of the perturbed model parameter can be derived as follows:

$$Var[\mathcal{M}(w)] = Var[w^*] = Var[c + \mu^*] = Var[\mu^*] = \mathbb{E}[\mu^{*2}] - \mathbb{E}^2[\mu^*] = \mathbb{E}[\mu^{*2}] - \mu^2; \tag{17}$$

$$\begin{aligned}\mathbb{E}[\mu^{*2}] &= \left(\mu \cdot \frac{e^\epsilon + 1}{e^\epsilon - 1}\right)^2 \cdot \frac{e^\epsilon - 1}{2e^\epsilon} + \left(\mu \cdot \frac{e^\epsilon - 1}{e^\epsilon + 1}\right)^2 \cdot \left(1 - \frac{e^\epsilon - 1}{2e^\epsilon}\right) \\ &= \frac{\mu^2}{2e^\epsilon} \cdot \left[\frac{(e^\epsilon + 1)^2}{e^\epsilon - 1} + \frac{(e^\epsilon - 1)^2}{e^\epsilon + 1}\right] \\ &\leq \frac{r^2}{2e^\epsilon} \cdot \left[\frac{(e^\epsilon + 1)^2}{e^\epsilon - 1} + \frac{(e^\epsilon - 1)^2}{e^\epsilon + 1}\right];\end{aligned} \tag{18}$$

$$\begin{aligned}Var[\mathcal{M}(w)] &\leq \frac{r^2}{2e^\epsilon} \cdot \left[\frac{(e^\epsilon + 1)^2}{e^\epsilon - 1} + \frac{(e^\epsilon - 1)^2}{e^\epsilon + 1}\right] - \mu^2 \\ &\leq \frac{r^2}{2e^\epsilon} \cdot \left[\frac{(e^\epsilon + 1)^2}{e^\epsilon - 1} + \frac{(e^\epsilon - 1)^2}{e^\epsilon + 1}\right] - r^2 \\ &= \frac{4r^2}{e^{2\epsilon} - 1}.\end{aligned} \tag{19}$$

□

Then, for the variance of the estimated average perturbed weight $\overline{\mathcal{M}(w)}$, we analyze its lower bound and upper bound as below.

Theorem 5: Let $\overline{\mathcal{M}(w)}$ be the estimated average weight. The lower bound and upper bound of the estimated average weight is: $0 \le Var[\overline{\mathcal{M}(w)}] \le \frac{4r^2}{N \cdot (e^{2\epsilon}-1)}$.

Proof: The variance of the estimated average weight is

$$\begin{aligned} Var[\overline{\mathcal{M}(w)}] &= Var\left[\frac{1}{N_t}\sum_{i=1}^{N_t}\mathcal{M}(w)\right] \\ &= \frac{1}{N_t}\left\{\frac{\mu^2}{2e^\epsilon}\cdot\left[\frac{(e^\epsilon+1)^2}{e^\epsilon-1}+\frac{(e^\epsilon-1)^2}{e^\epsilon+1}\right]-\mu^2\right\} \\ &= \frac{4\mu^2}{N_t\cdot(e^{2\epsilon}-1)}. \end{aligned} \tag{20}$$

Thus, the bound of variance is

$$0 \le Var[\overline{\mathcal{M}(w)}] \le \frac{4r^2}{N_t\cdot(e^{2\epsilon}-1)}. \tag{21}$$

□

From Theorem 5 above, we can see that the bounds of variance are determined by the radius r, the number N_t of clients participating in the round t and the privacy budget ϵ. Thus, if there are enough participating clients, we can get a low variance on the estimated average perturbed weight.

Finally, since the model accuracy is directly affected by the average perturbation weight $\overline{\mathcal{M}(w)}$, we need to ensure that the deviation from the true average weight $\overline{w}$ is within an acceptable range. Therefore, we will analyze the accuracy guarantee for calculating average weights based on the above theorems.

Theorem 6: Given any weight w, with at least $1-\beta$ probability, $|\overline{\mathcal{M}(w)}-\overline{w}| < O(\frac{r\sqrt{-\log\beta}}{\epsilon\sqrt{N_t}})$, where β is a small number which is close to 0.

Proof: For each client C_i, $|\mathcal{M}(w)-w| \le c + r\cdot\frac{e^\epsilon+1}{e^\epsilon-1} - (c-r) = \frac{2re^\epsilon}{e^\epsilon-1}$ and $Var[\mathcal{M}(w)] = Var[\mathcal{M}(w)-w] = \mathbb{E}[(\mathcal{M}(w)-w)^2] - \mathbb{E}^2[\mathcal{M}(w)-w]$. According to Theorem 2, $\mathbb{E}[\mathcal{M}(w)-w]=0$. Thus, $Var[\mathcal{M}(w)] = \mathbb{E}[(\mathcal{M}(w)-w)^2]$ holds according to Bernstein inequality,

$$\begin{aligned} &Pr[|\overline{\mathcal{M}(w)}-\overline{w}| \ge \lambda] \\ &= Pr\left[\left|\sum_{i=1}^{N_t}\mathcal{M}(w)-\sum_{i=1}^{}N_t w_k\right| \ge \lambda N_t\right] \\ &\le 2\cdot\exp\left\{-\frac{\frac{1}{2}\lambda^2N_t^2}{\sum_{i=1}^{N_t}\mathbb{E}[(\mathcal{M}(w)-w)^2]+\frac{2\lambda N_t r e^\epsilon}{3(e^\epsilon-1)}}\right\} \\ &\le 2\cdot\exp\left\{-\frac{\lambda^2N_t^2}{2N_t\frac{4r^2}{e^{2\epsilon}-1}+\frac{4\lambda N_t r e^\epsilon}{3(e^\epsilon-1)}}\right\} \\ &= 2\cdot\exp\left\{-\frac{\lambda^2N_t}{\frac{8r^2}{e^{2\epsilon}-1}+\frac{4\lambda r e^\epsilon}{3(e^\epsilon-1)}}\right\} \\ &= 2\cdot\exp\left\{-\frac{\lambda^2N_t}{r^2\cdot O(\epsilon^{-2})+\lambda r\cdot O(\epsilon^{-1})}\right\}. \end{aligned} \tag{22}$$

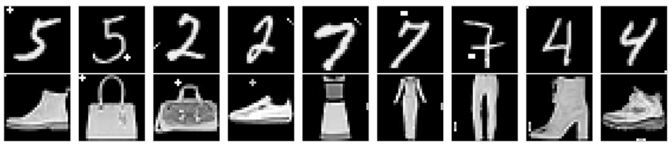


Fig. 5. Visual pictures for dynamic backdoor attacks.

In other words, there exists $\lambda = O(\frac{r\sqrt{-\log\beta}}{\epsilon\sqrt{N_t}})$ such that $|\overline{\mathcal{M}(w)} - \overline{w}| < \lambda$ holds with at least $1-\beta$ probability. □

VII. Experiments

In this section, we will show the performance of our scheme on different DNN models and datasets.

A. Experimental Setup

1) Datasets and Backdoor Attack Schemes: We test the backdoor attack and our defense against it using three common open source datasets, MNIST, Fashion-MNIST, and CIFAR10. To simulate the realistic Non-IID scenarios, we first scramble the entire dataset, and then divide it into multiple slices evenly, where each slice contains an equal number of samples but the labels are not distributed evenly. Each client has an unequal number of slices. Take MNIST dataset as an example, a total of 60,000 training samples are divided into 300 slices, that is, each slice contains 200 samples, in which slice 1 only contains samples of digits 1, 4 and 5, and slice 2 only contains samples of digits 3 and 7, etc. Moreover, in slice 1, the numbers of samples of digits 1, 4 and 5 are also different.

In the experiments, we test the performance of our defense scheme in three state-of-the-art backdoor attacks schemes. We will briefly describe their core attack methods as follows:

1) *Model Replacement Attacks (MRA) [36]:* Assume that the client C_1 is malicious. $\mathcal{A}$ attempts to replace the global model $\mathbf{w}^t$ with its malicious model $\mathbf{w}^t_{adv}$, i.e., $\mathbf{w}^t_{adv} = \mathbf{w}^{t-1} + \frac{\zeta}{N}\cdot\sum_{i=1}^{N_t}\Delta\mathbf{w}^t_i$. When the model training approaches convergence, the local model updates $\Delta\mathbf{w}^t_i$ uploaded by the honest clients ($i \in [2, N_t]$) are almost equal to 0. Thus, $\mathbf{w}^t_{adv} \approx \mathbf{w}^{t-1} + \frac{\zeta}{N}\cdot\Delta\mathbf{w}^t_1 = \mathbf{w}^{t-1} + \frac{\zeta}{N}\cdot(\mathbf{w}^t_1 - \mathbf{w}^{t-1})$. Then the local model $\mathbf{w}^t_1$ of client C_1 is scaled up to $\frac{N}{\zeta}\cdot(\mathbf{w}^t_{adv}-\mathbf{w}^{t-1})+\mathbf{w}^{t-1}$, where $\frac{N}{\zeta}$ is the scaling factor.
2) *Dynamic Backdoor Attacks (DBA-I) [37]:* The authors in [37] consider the influence of different trigger patterns and different adding positions on backdoor attacks. In our experiments, the attacker generates different patterns of triggers and adds them to any position in the picture. For example, Fig. 5 shows the visual pictures in which triggers of same or different patterns are added at different positions.
3) *Distributed Backdoor Attacks (DBA-II) [38]:* The authors in [37] exploit the distributed learning concept of FL to decompose the complete global trigger into multiple local triggers, and make malicious clients add only one of

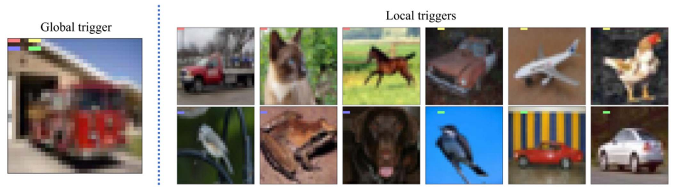


Fig. 6. Visual pictures for distributed backdoor attacks.

TABLE II
MODEL STRUCTURE

Dataset	Model structure
MNIST	conv1 (1, 32, (5 × 5), 0) - MaxPool (2) - conv2 (32, 64, (5 × 5), 0) - MaxPool (2) - full_connection (1024, 10) - ReLU - softmax
Fashion-MNIST	conv1 (1, 64, (1 × 1), 1) - conv2 (64, 64, (3 × 3), 1) - MaxPool (2) - BatchNorm(64) - ReLU - conv3 (64, 128, (3 × 3), 1) - conv4 (128, 128, (3 × 3), 1) - MaxPool (2) - BatchNorm(128) - ReLU - full_connection1 (128 * 8 * 8, 256) - ReLU - full_connection2 (256, 10)
CIFAR10	ResNet18 [39]

Notes: $conv$ is the convolution layer. $full\_connection$ is the full connection layer. $MaxPool$ is the max pool layer. $ReLu$ and $softmax$ are activation functions. $BatchNorm$ means batch normalization of results.

the local triggers to their datasets. For example, the left part of Fig. 6 shows the complete global trigger of the previous schemes, while the right part of Fig. 6 shows the decomposed local triggers of DBA-II.

2) Model Structure: The model structures we adopted are shown in Table II.

3) Metric Criteria and Key Parameters: To simulate the actual attack scene as much as possible, we define the key parameters for $\mathcal{A}$'s attack strategies and ability, and the real training process of FL, which need to be considered for any attack or defense scheme.

Note that $\mathcal{A}$ will not pollute all instances, which has two advantages: 1) it prevents the poisonous models from being detected due to abnormal performance in any instance; 2) it improves the distinguishability of the trained model between the backdoor and the non-backdoor, so as to promote the attack success rate. For simplicity, we define the notion of local poisoning ratio as follows,

Definition 2 (Local Poisoning Ratio (LPR)): For a malicious client C_i, its local dataset D_i contains a total of d_i instances, in which includes $\widehat{d_i}$ instances with triggers. Therefore, its local poisoning ratio $LPR = \widehat{d_i}/d_i$.

In the community, a lot of previous work has assumed that FL is in a honesty-majority situation where the total number $P_{\mathcal{A}}$ of malicious clients is less than 50% out of a total of N clients, and tests the attack success rate or effect of defense mechanism with the increase of $P_{\mathcal{A}}$. However, in the general scenario of FL, the clients and their number N_t in each round are randomly selected from N. Therefore, we cannot guarantee that malicious clients can participate in every round. Consequently, in this paper, we define the notion of participation rate PR as follows,

Definition 3 (Participation Rate (PR)): Among the total T of communication rounds, the number of rounds that the malicious clients are really selected to participate in model training is $T_{\mathcal{A}}$. Thus, the participation rate $PR = T_{\mathcal{A}}/T$.

In our experiments, we test the attack success rate and the effect of defense mechanism with the change of PR, which is not widely considered in prior work. Moreover, the proportion $P_{\mathcal{A}}$ of malicious clients is actually a very important metric criteria, which we will slightly modify its definition. Since the number of clients participating in the t-th round is N_t, it may include the existence of malicious clients (assuming the number is $N_{\mathcal{A}_t}$) or not. Therefore, we have the following definition,

Definition 4 (Malicious proportion): In the t-th round, there are total N_t clients randomly selected, including $N_{\mathcal{A}_t}$ malicious clients. Thus, the malicious proportion $\widetilde{P_{\mathcal{A}}} = N_{\mathcal{A}_t}/N_t$, which is slightly different from $P_{\mathcal{A}}$ in concept. We also test how the attack success rate and the effectiveness of defense mechanism vary as the $\widetilde{P_{\mathcal{A}}}$ changes.

In a word, LPR and $\widetilde{P_{\mathcal{A}}}$ simulate $\mathcal{A}$'s attack strategies in one communication round, while PR simulates $\mathcal{A}$'s attack ability in a total of T communication rounds.

For non-malicious clients, they desire the model to correctly classify as many data instances as possible. Therefore, we use accuracy as their metric criterion. As for malicious clients, they desire the model to remember as many triggers as possible and classify them into adversary-chosen labels. Therefore, we use the attack success rate (ASR) defined below as the adversary's metric criterion.

Definition 5 (Attack Success Rate (ASR)): The adversary $\mathcal{A}$ gets the number $N_{\mathcal{A}_t}$ of malicious clients according to the malicious proportion $\widetilde{P_{\mathcal{A}}}$. Let the dataset size of malicious client $C_i (i \in [1, N_{\mathcal{A}_t}])$ be d_i. C_i calculates the number $\widehat{d_i}$ of data instances to be poisoned according to the local poisoning ratio LPR, and counts the number $\widehat{d_i^*}$ of data instances that are successfully classified as the adversary-chosen labels by the model. Then the attack success rate ASR$_i$ of C_i equals to $\widehat{d_i^*}/\widehat{d_i}$. Finally the adversary $\mathcal{A}$ calculates the average attack success rate ASR $= \frac{1}{N_{\mathcal{A}_t}} \cdot \sum_{i=1}^{N_{\mathcal{A}_t}}$ ASR$_i$.

Table IV summarizes the values of the key parameters measured in our experiments. Note that the bolded values represent the default values of other parameters when measuring the impact of changes in one parameter on the scheme, unless explicitly indicated.

4) Runtime Environment: Our experiments are implemented by Pytorch 1.10.0 and Numpy 1.21.5 with a single CPU @ 3.30 GHz, 16.0 GB RAM.

B. Effect of Privacy Budget and Compression Ratio

The two most critical parameters in our scheme are the privacy budget ϵ and the compression ratio CR. Therefore, we first show the influence of these two parameters on the performance of the schemes under different attack methods. Note that for

TABLE III
EFFECT OF PRIVACY BUDGET ϵ AND COMPRESSION RATIO CR ON MRA

Scheme	ϵ	MNIST		FMNIST		CIFAR10		CR	MNIST		FMNIST		CIFAR10	
		Acc	ASR	Acc	ASR	Acc	ASR		Acc	ASR	Acc	ASR	Acc	ASR
FedAvg(no attack)	$+\infty$	98.04	╲	90.31	╲	88.76	╲	1	98.04	╲	90.31	╲	88.76	╲
FedAvg+MRA	$+\infty$	97.92	100	90.87	100	88.57	100	1	97.92	100	90.87	100	88.57	100
Clipping+Gauss [5]	1	82.45	8.26	75.84	12.33	70.23	10.78	0.05	88.76	10.83	78.50	12.79	72.78	13.66
	2	88.76	10.83	78.50	12.79	72.78	13.66	0.1	90.92	10.63	82.61	12.06	73.12	13.53
	3	90.33	12.30	81.46	16.89	73.45	14.74	0.5	92.64	12.84	85.11	11.73	72.94	14.21
	4	92.74	13.59	86.45	15.74	76.82	18.33	1	95.38	13.74	85.04	12.18	73.89	14.56
CAFL [21]	1	94.45	12.09	82.30	14.34	82.47	25.36	0.05	95.16	14.58	85.46	15.88	83.69	29.66
	2	95.16	14.58	85.46	15.88	83.69	29.66	0.1	95.33	13.40	85.94	15.82	82.89	27.45
	3	95.41	15.00	85.43	20.96	84.57	35.19	0.5	96.38	16.75	86.71	16.21	83.46	30.29
	4	96.78	17.36	86.74	25.41	86.74	49.23	1	96.98	16.23	88.96	14.08	83.99	32.75
ESFL	1	93.94	7.25	82.64	12.32	80.33	13.68	0.05	95.35	9.63	85.06	14.36	81.68	17.54
	2	95.35	9.63	85.06	14.36	81.68	17.54	0.1	95.46	9.47	86.24	14.01	82.04	18.66
	3	96.91	8.60	85.22	18.74	81.59	18.23	0.5	96.33	10.05	86.53	13.35	82.74	17.78
	4	97.56	10.58	87.61	19.86	82.41	20.19	1	96.45	11.52	88.97	14.03	83.46	19.45

TABLE IV
SUMMARY OF KEY PARAMETERS

Symbol	Connotation	Values
T	communication rounds	100 / 200
η	local learning rate	0.05 / 0.01
ζ	global learning rate	$\frac{N_t}{N}$
LPR	Local Poisoning Ratio	0 / 0.1 / 0.2 / 0.3 / 0.4 / **0.5**
PR	Participation Rate	0 / **0.2** / 0.4 / 0.6 / 0.8 / 1
$\widetilde{P_A}$	Malicious proportion	0 / **0.1** / 0.2 / 0.3 / 0.4 / 0.5
CR	compression ratio	**0.05** / 0.1 / 0.5 / 1
ϵ	privacy budget	1 / **2** / 3 / 4 / $+\infty$

traditional differential privacy defense methods, they do not have the parameter CR. But for the sake of equal comparison, we added this step to other defense schemes that did not use compressive sensing in our experiments. Tables III, V and VI show the effect of privacy budget and compression ratio of defense scheme on resisting MRA, DBA-I and DBA-II attacks.

Note that Tables III, V and VI show the results of the ablation experiments. When we change the ϵ (or CR), the CR (or ϵ) is fixed. Although we fix CR as 0.05 instead of 1 when changing ϵ alone, CS's main contribution is to reduce the overhead of uplink communication rather than resist backdoor attacks because of its decompression ability. Similarly, when we change CR separately, we fix ϵ as 2 instead of $+\infty$ because adding noise will not cause additional burden on communication overhead. The contribution of CS is shown in Section VII-F.

For the privacy budget ϵ, the results show that the smaller ϵ is, the smaller the model accuracy on the main task is, and the lower the attack success rate on the backdoor task is. Besides, we can see that, compared with the traditional defense method [5] of clipping and Gaussian noise, CAFL [21] has improved the accuracy of the main task (whether it is a simple dataset or a complex dataset), but it has shown different performances in reducing the attack success rate of backdoor tasks for datasets with different difficulties. For example, CAFL is nearly as effective against backdoor attacks on the simple dataset MNIST as traditional methods, but less effective against complex datasets FMNIST and CIFAR10. However, the accuracy of ESFL on the main task is almost consistent with that of CAFL, and the attack success rate on the backdoor task is almost consistent with the strong defense ability of traditional scheme (whether on simple dataset MNIST or complex datasets FMNIST and CIFAR10).

For compression ratio CR, its main function is to reduce the overhead of uplink communication from clients to server. Experimental results show that compressive sensing (CS) reduces the communication cost by 95%. In addition, we also find that CS is helpful to improve the defense capability of the schemes. For example, with the reduction of CR, the degree of compression is also strengthened, and the attack success rate of backdoor attacks is also reduced by several percentage points. This is because CS is a kind of lossy compression technique, and the server cannot perfectly recover the real aggregation results, which leads to the introduction of noise in another form and achieves the effect of defending against backdoor attacks.

Note that ESFL is improved on CAFL, and CAFL is improved on [9]. We have tested the effectiveness of CAFL and [9] in resisting backdoor attacks, and the results show that they are almost consistent. Therefore, we only show the experimental results of CAFL in this paper. From Tables III, V and VI, we can find that ESFL is more robust than CAFL, which indirectly shows that ESFL is also more resistant to backdoor attacks than [9].

C. *Effect of Local Poisoning Ratio*

When an attacker launches the backdoor attacks, it generally only pollutes part of the local dataset in order for the model to be able to distinguish between backdoor data and clean data, and in order to avoid being detected by the server due to its excessively abnormal behavior. Therefore, we test the effect of the local poisoning ratio (LPR) on attack capability and defense mechanisms in experiments. Note that LPR $= 0$ indicates that the attacker does not launch the backdoor attacks. The test results on the MNIST, Fashion-MNIST and CIFAR10 datasets are shown in Fig. 7.

It can be seen from the solid broken line that the traditional gaussian noise will seriously reduce the model accuracy on the main task. For example, the accuracy of the classical FedAvg algorithm is 97.92% when it is attacked by MRA on the simple dataset MNIST and there is no defense, while the accuracy is

TABLE V
Effect of Privacy Budget ϵ and Compression Ratio CR on DBA-I

Scheme	ϵ	**MNIST**		**FMNIST**		**CIFAR10**		CR	**MNIST**		**FMNIST**		**CIFAR10**	
		Acc	ASR	Acc	ASR	Acc	ASR		Acc	ASR	Acc	ASR	Acc	ASR
FedAvg(no attack)	$+\infty$	98.04	\	90.31	\	88.76	\	1	98.04	\	90.31	\	88.76	\
FedAvg+DBA-I	$+\infty$	98.32	100	89.56	100	88.23	100	1	98.32	100	89.56	100	88.23	100
Clipping+Gauss [5]	1	84.21	9.53	75.22	10.43	71.86	12.37	0.05	87.61	12.74	75.76	13.47	72.62	14.28
	2	87.61	12.74	75.76	13.47	72.62	14.28	0.1	91.76	13.15	80.05	12.77	74.25	15.22
	3	88.86	16.78	77.70	16.24	75.52	15.71	0.5	93.25	15.98	84.65	15.27	75.89	17.42
	4	90.38	22.37	78.64	21.86	76.44	18.59	1	94.08	14.36	85.88	14.37	75.21	16.64
CAFL [21]	1	94.78	15.23	83.47	15.36	83.46	22.41	0.05	95.98	16.74	86.72	23.56	83.10	27.12
	2	95.98	16.74	86.72	23.56	83.10	27.12	0.1	96.41	20.15	86.30	27.16	84.22	26.58
	3	96.37	21.41	88.65	29.79	83.69	33.74	0.5	96.22	22.35	85.36	25.68	84.51	26.74
	4	96.66	25.96	88.97	35.77	84.85	45.78	1	96.21	18.31	88.22	25.36	84.36	26.55
ESFL	1	90.63	8.66	82.30	10.28	78.12	15.62	0.05	93.26	13.77	82.39	15.21	80.11	17.16
	2	93.26	13.77	82.39	15.21	80.11	17.16	0.1	94.78	14.56	85.54	13.46	81.91	17.55
	3	95.75	15.27	85.73	19.35	81.65	19.36	0.5	95.77	14.88	85.97	16.21	81.47	18.12
	4	96.99	19.32	87.45	24.64	82.84	21.26	1	95.36	15.25	87.56	20.88	82.03	18.76

TABLE VI
Effect of Privacy Budget ϵ and Compression Ratio CR on DBA-II

Scheme	ϵ	**MNIST**		**FMNIST**		**CIFAR10**		CR	**MNIST**		**FMNIST**		**CIFAR10**	
		Acc	ASR	Acc	ASR	Acc	ASR		Acc	ASR	Acc	ASR	Acc	ASR
FedAvg(no attack)	$+\infty$	98.04	\	90.31	\	88.76	\	1	98.04	\	90.31	\	88.76	\
FedAvg+DBA-II	$+\infty$	97.69	100	90.07	100	87.85	100	1	97.69	100	90.07	100	87.85	100
Clipping+Gauss [5]	1	82.74	7.66	73.47	11.19	72.14	8.46	0.05	83.56	10.21	75.34	14.11	74.74	10.14
	2	83.56	10.21	75.34	14.11	74.74	10.14	0.1	85.43	12.77	76.71	13.46	75.66	12.75
	3	85.26	13.43	76.04	15.32	76.11	14.76	0.5	86.46	12.46	79.12	15.45	74.26	12.96
	4	86.14	15.74	78.57	18.71	78.86	17.27	1	88.58	13.13	81.33	15.18	75.87	13.42
CAFL [21]	1	93.56	19.75	83.16	14.47	82.71	20.88	0.05	93.77	21.64	84.57	19.55	83.42	26.32
	2	93.77	21.64	84.57	19.55	83.42	26.32	0.1	93.24	20.36	85.14	20.37	83.11	25.74
	3	94.24	25.78	85.04	24.86	83.85	31.74	0.5	94.02	22.45	84.87	22.14	85.52	24.66
	4	94.33	30.65	85.52	31.78	84.11	42.96	1	94.25	20.53	86.41	21.96	85.36	24.45
ESFL	1	83.78	9.37	76.88	13.74	75.16	11.11	0.05	84.19	12.55	77.14	15.55	76.47	14.69
	2	84.19	12.55	77.14	15.55	76.47	14.69	0.1	85.34	14.74	78.33	16.74	76.99	16.43
	3	87.12	15.86	79.28	18.61	77.31	18.37	0.5	85.97	15.23	78.22	17.81	77.22	16.74
	4	87.96	19.44	80.12	22.74	80.74	22.94	1	86.95	14.26	79.65	17.95	78.31	15.77

88.76% after adding Gaussian noise for defense, a decrease of 9.16%, which is unacceptable for such a simple dataset. In contrast, the accuracy of CAFL and ESFL in the main task only decreased by 2.76% and 2.57%, respectively. This indicates that CAFL and ESFL can guarantee the availability of the model.

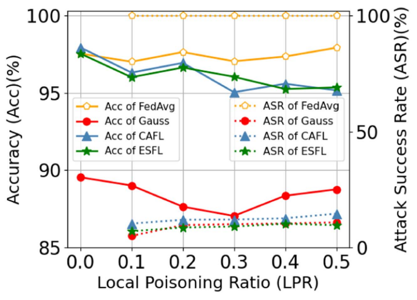
With the increase of local poisoning rate (LPR), the attack success rate (ASR) of backdoor attacks should increase both intuitively and theoretically. The experimental results also confirmed this, but the increase is very small. For example, when the LPR increases from 10% to 50%, the maximum gap between ASR is only 7.67%, which indicates the robustness of the differential privacy defense mechanism against backdoor attacks. In addition, the ASR of CAFL is very close to that of traditional Gaussian noise and ESFL on simple dataset MNIST, and the difference is very significant on complex datasets Fashion-MNIST and CIFAR10. For example, the maximum difference between ASR of CAFL on MNIST and traditional Gaussian noise and ESFL is about 5%, while their maximum difference on Fashion-MNIST is about 25.6% and about 30% on CIFAR10. Furthermore, the ASR of ESFL is located between CAFL and traditional Gaussian noise, and is closer to the latter, which indicates that ESFL can achieve strong protection capability similar to Gaussian noise.

D. *Effect of Participation Rate*

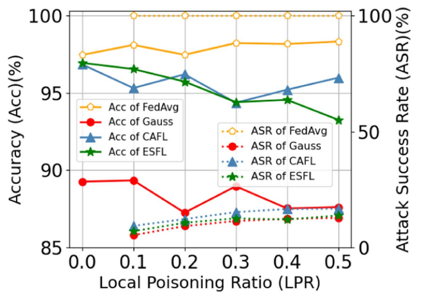
Since the server randomly selects N_t out of N clients to participate in t-th round communication, the attacker cannot control the number of rounds of attacks by malicious clients. But we intuitively know that the more attack rounds the attacker participates in, the better the attack effect will be (i.e., the larger ASR). Therefore, in order to simulate the robustness of each scheme against the backdoor attacks under this scenario, we define the total number of participation rounds T_A of the malicious clients in the total number of T rounds of communication in Definition 3, i.e., the participation rate $PR = T_A/T$. Note that the extreme cases of $PR = 0$ and $PR = 1$ indicate that there is no attacker and there are malicious clients participating in the model training for each round of communication, respectively. The test results on the MNIST, Fashion-MNIST and CIFAR10 datasets are shown in Fig. 8.

From the experimental results, with the increase of the participation rate PR of malicious clients, the model accuracy on the main task generally presents a downward trend, while the ASR on the backdoor task generally presents an upward trend, which verifies our intuition.

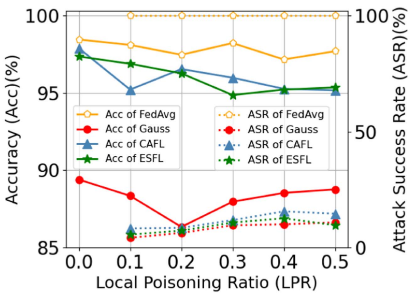
In terms of model accuracy, CAFL and ESFL decrease by about 5% and Gaussian noise by about 10% as compared with



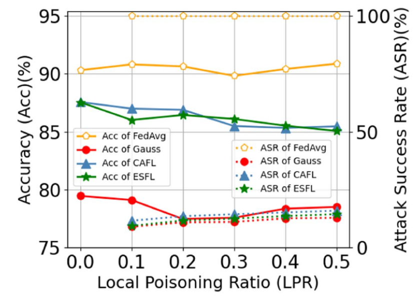
(a) MRA on MNIST



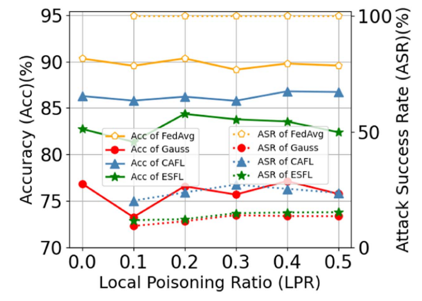
(b) DBA-I on MNIST



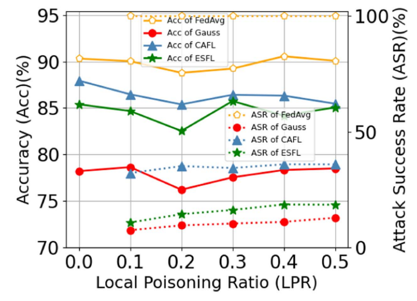
(c) DBA-II on MNIST



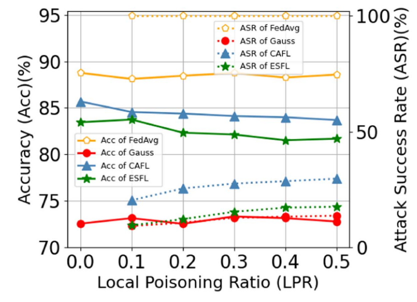
(d) MRA on Fashion-MNIST



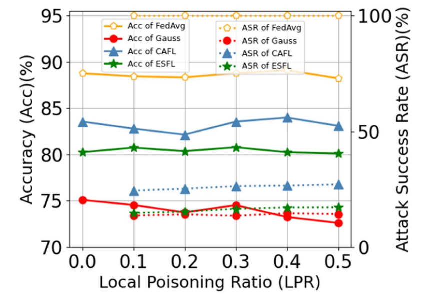
(e) DBA-I on Fashion-MNIST



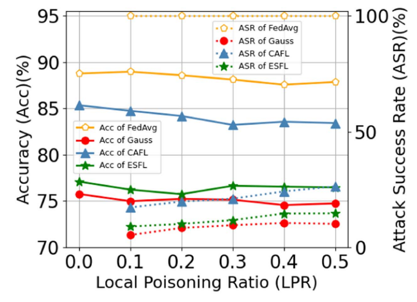
(f) DBA-II on Fashion-MNIST



(g) MRA on CIFAR10



(h) DBA-I on CIFAR10



(i) DBA-II on CIFAR10

Fig. 7. Impact of local poisoning ratio.

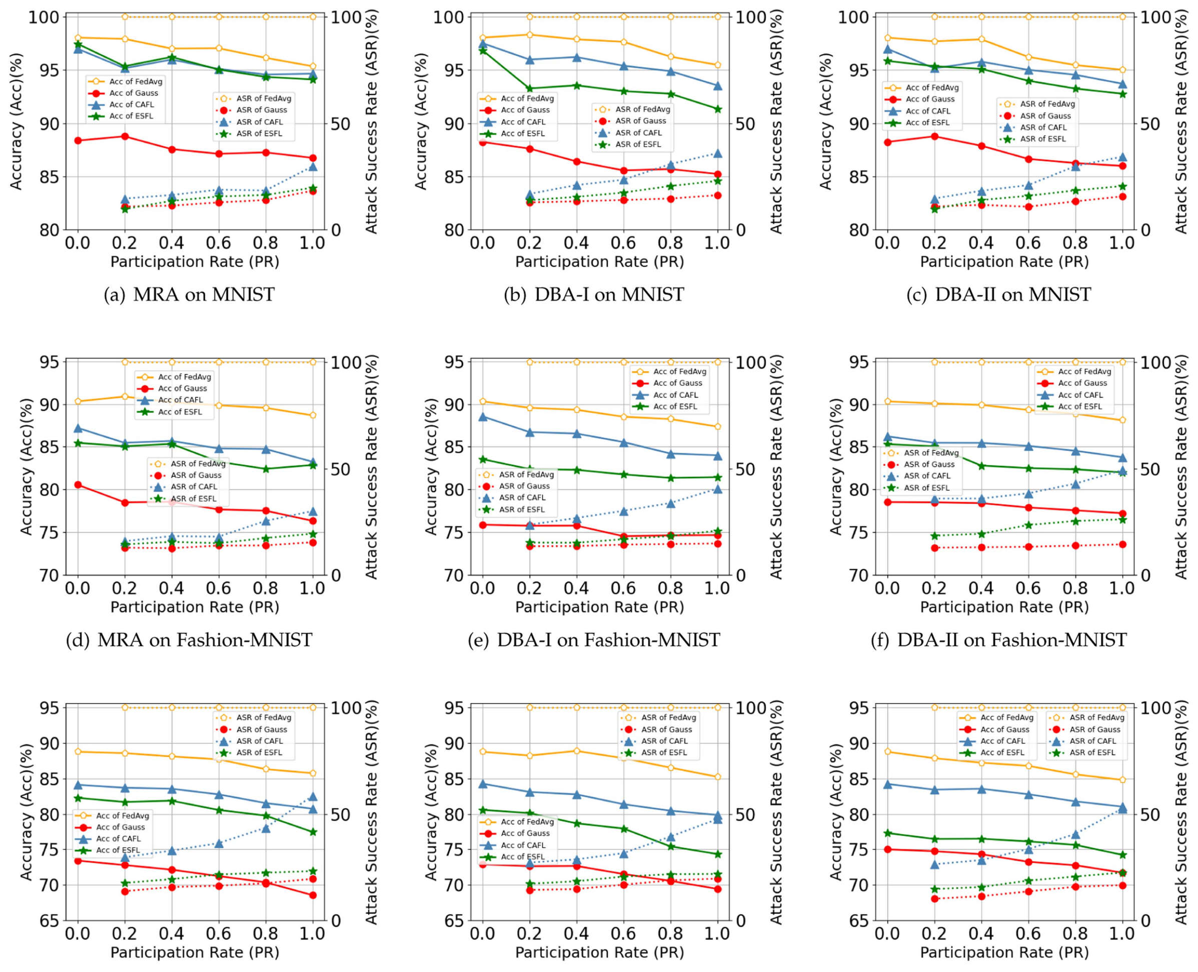
plain FedAvg, which indicates that ESFL can guarantee the availability of the model under different PR of malicious clients. In addition, CAFL contains only an adaptive noise, while ESFL introduces two kinds of noises and one of them is also the noise that impairs the model utility. Therefore, in most cases, the model accuracy ESFL is lower than that of CAFL, but the difference is only about 5% at most. For example, as shown in Fig. 8(e), when $PR = 0$, the model accuracy of ESFL is 5.01% lower than that of CAFL; when $PR = 0.2$, the model accuracy of ESFL is 4.33% lower than that of CAFL.

As far as ASR of backdoor task is concerned, the results of ESFL are between CAFL and Gaussian noise, and the virtual polyline of ASR of ESFL is closer to Gaussian noise than CAFL, which indicates that ESFL has strong defense capability against backdoor attacks. Moreover, we also found that CAFL has different defense effects for the backdoor attacks schemes with different degrees of complexity. For example, CAFL's defense effect on MRA, DBA-I and DBA-II gradually decreases. When $PR = 1$, the ASR of CAFL is about 25% for MRA, about 40% for DBA-I, and nearly 50% for DBA-II.

E. Effect of Malicious Proportion $\widetilde{P_{\mathcal{A}}}$

Many of the current schemes focus on the proportion of malicious clients in all clients because this is a critical factor in the effectiveness of an attack and an important factor in assessing defense capabilities. However, under the general framework of federated learning, not all clients will participate in every round of communication, but some random clients. Then the proportion of malicious clients in all clients loses its original meaning. For example, suppose that the percentage of malicious clients is 50%, but due to the random scheduling principle, the percentage of malicious clients that actually launch attacks will be significantly lower than 50%. Therefore, in the experiments, we turn to control the proportion of malicious clients among the N_t clients in t-th communication round, i.e., Malicious Proportion $\widetilde{P_{\mathcal{A}}}$. The test results on the MNIST, Fashion-MNIST and CIFAR10 datasets are shown in Fig. 9.

The test results show that the broken lines (whether the accuracy of the main task or the ASR of the backdoor tasks) obviously show a slight fluctuation. This indicates that even if



(a) MRA on MNIST (b) DBA-I on MNIST (c) DBA-II on MNIST

(d) MRA on Fashion-MNIST (e) DBA-I on Fashion-MNIST (f) DBA-II on Fashion-MNIST

(g) MRA on CIFAR10 (h) DBA-I on CIFAR10 (i) DBA-II on CIFAR10

Fig. 8. Impact of participation rate.

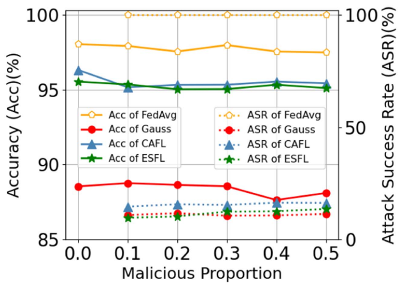
the percentage of malicious clients in a certain round of communication reaches 50%, the local differential privacy-based defense method can still effectively weaken the role of backdoor attacks.

Similarly, under different $\widetilde{P_{\mathcal{A}}}$, the model trained by the defense method based on Gaussian noise has much lower accuracy on the main task than those of CAFL and ESFL, and the curves of CAFL and ESFL are very close in most cases. Generally speaking, the model accuracy of ESFL is lower than that of CAFL, but only about 2.5% lower at most, as shown in Fig. 9(b). The model accuracy of ESFL is higher than that of gaussian noise, about 6% on average, which indicates that ESFL can resist attacks and ensure the availability of the model.

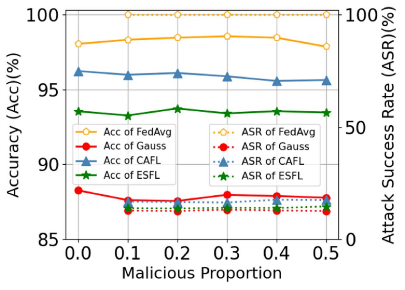
1) Discussion: From the existing work on attacks, as the percentage of malicious clients increases, the attack success rate should also increase. But our experimental results do not show this phenomenon. We suspect this is because the rounds that malicious clients are selected by the server to participate in are almost always the first part of the total rounds T. In the framework of FL, the server needs to perform aggregation operation, and the role of aggregation can be seen as either cohesion or weakening individual contributions. If the rounds selected by the server for the malicious client are located in the front part of the total number of rounds T, for example, in the first $\frac{1}{3}T$ rounds, then the effect of the attack is gradually weakened as the communication round t increases. Thus, we cannot determine whether the aggregation operation or the differential privacy mechanism works against the backdoor attacks.

To solve the above doubts and guesses, in another set of tests, we fix the rounds t that malicious clients participate to the front and latter part of the total communication rounds T, e.g., $t \in [0, \frac{1}{3}T]$ and $t \in [\frac{2}{3}T, T]$ respectively. To show the impact of a uniform distribution of t, we test the scenario where malicious clients would be selected every I rounds and launch the backdoor attacks. For illustration purposes, the above three situations correspond to front-fixing, tail-fixing and even-fixing, respectively. The test results are shown in Fig. 10 and we only test MRA attacks.

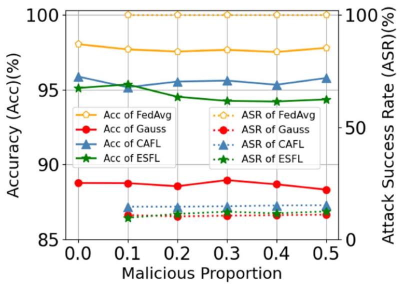
The broken lines in Fig. 10(a) are more horizontal than those in Fig. 10(b) and (c), which indicates that the timing of launching the attacks is critical. If it is front fixing, the attacks effect will be mainly weakened by aggregation operation on the server. If it is tail-fixing or even-fixing, the attacks effect is mainly weakened by the local differential privacy mechanism.



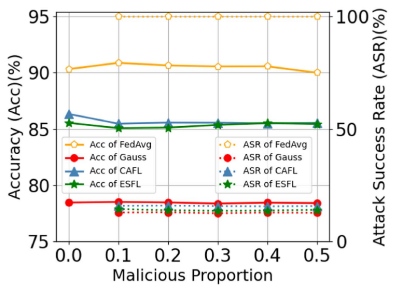
(a) MRA on MNIST



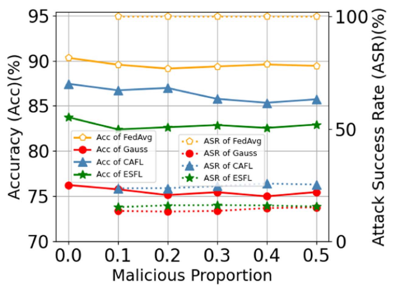
(b) DBA-I on MNIST



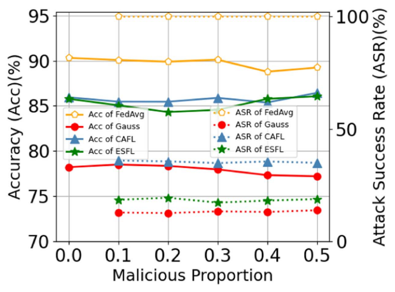
(c) DBA-II on MNIST



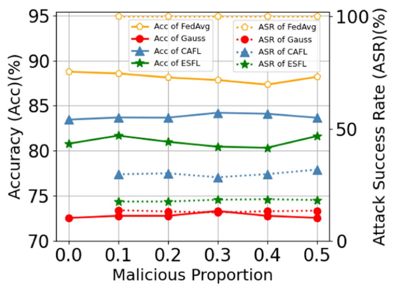
(d) MRA on Fashion-MNIST



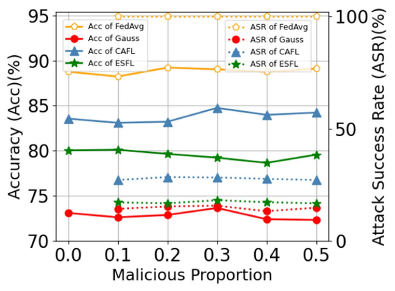
(e) DBA-I on Fashion-MNIST



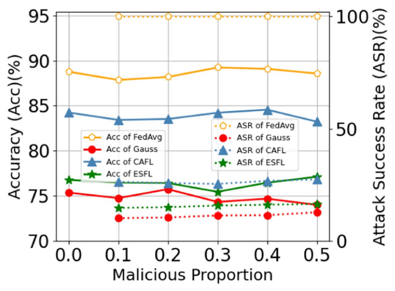
(f) DBA-II on Fashion-MNIST



(g) MRA on CIFAR10

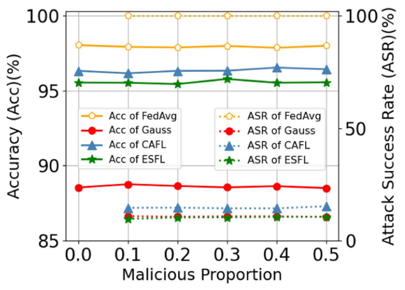


(h) DBA-I on CIFAR10

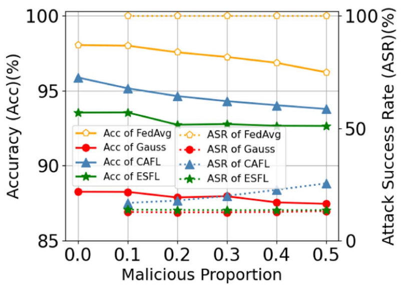


(i) DBA-II on CIFAR10

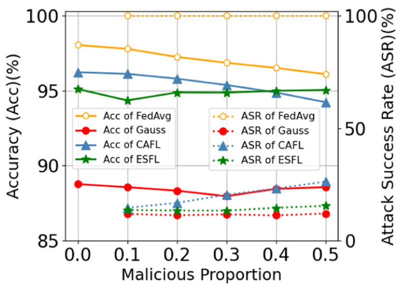
Fig. 9. Impact of malicious proportion $\widetilde{P_{\mathcal{A}}}$.



(a) front fixing



(b) tail fixing



(c) even fixing

Fig. 10. Impact of malicious proportion $\widetilde{P_{\mathcal{A}}}$ on MNIST for different round t.

F. Communication Efficiency of ESFL

Similar to CAFL [21], we adopt CS to compress and upload models, which reduces the uplink communication overhead on the premise of ensuring the model accuracy. It is worth noting that the extended function of ESFL compared with CAFL and the noise adding process of LDP mechanism will not cause additional burden on communication efficiency. Therefore, the communication efficiency of ESFL is as superior as that of CAFL. Read CAFL for detailed experimental results.

VIII. Conclusion

In this paper, to solve the issues that the traditional DP-based FL cannot realize adaptive perturbation and the high communication cost under the DNN model structure, we first propose a basic scheme that uses CS to alleviate uplink traffic and adaptively perturbs local model by exploiting the hierarchy of DNN. On this basis, we further adopt the adaptive perturbation mechanism to adjust the traditional gaussian noise, which resists backdoor attack. Extensive experiments prove that we can improve the model accuracy and defense performance under a lower privacy budget, and reduce the communication cost. In the future work, we will test the effectiveness of our scheme on larger DNN models and datasets, and further reduce the communication overheads of uploading and downloading models.

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 *Proc. 20th Int. Conf. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

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

[3] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Proc. 32nd Annu. Conf. Neural Inf. Process. Syst.*, 2019, pp. 14 747–14 756.

[4] T. D. Nguyen et al., "FLAME: Taming backdoors in federated learning," in *Proc. 31st USENIX Secur. Symp.*, 2022, pp. 1415–1432.

[5] M. Naseri, J. Hayes, and E. De Cristofaro, "Local and central differential privacy for robustness and privacy in federated learning," in *Proc. 29th Netw. Distrib. Syst. Secur. Symp.*, 2022, pp. 1–18.

[6] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," 2017, *arXiv: 1712.07557*.

[7] M. Seif, R. Tandon, and M. Li, "Wireless federated learning with local differential privacy," in *Proc. IEEE Int. Symp. Inf. Theory*, 2020, pp. 2604–2609.

[8] R. Kerkouche, G. Ács, C. Castelluccia, and P. Genevès, "Compression boosts differentially private federated learning," in *Proc. IEEE Eur. Symp. Secur. Privacy*, 2021, pp. 304–318.

[9] L. Sun, J. Qian, and X. Chen, "LDP-FL: Practical private aggregation in federated learning with local differential privacy," in *Proc. 30th Int. Joint Conf. Artif. Intell.*, 2021, pp. 1571–1578.

[10] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," in *Proc. 5th Int. Conf. Learn. Representations*, 2017, pp. 1–17.

[11] C. Tai, T. Xiao, X. Wang, and W. E, "Convolutional neural networks with low-rank regularization," in *Proc. 4th Int. Conf. Learn. Representations*, 2016, pp. 1–16.

[12] S. Zhou, Z. Ni, X. Zhou, H. Wen, Y. Wu, and Y. Zou, "DoReFa-Net: Training low bitwidth convolutional neural networks with low bitwidth gradients," 2016, *arXiv:1606.06160*.

[13] G. Hinton et al., "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.

[14] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "FLTrust: Byzantine-robust federated learning via trust bootstrapping," in *Proc. 28th Annu. Netw. Distrib. Syst. Secur. Symp.*, 2021, pp. 1–18.

[15] C. Xu, Y. Jia, L. Zhu, C. Zhang, G. Jin, and K. Sharif, "TDFL: Truth discovery based byzantine robust federated learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 12, pp. 4835–4848, Dec. 2022.

[16] D. Yin, Y. Chen, K. Ramchandran, and P. L. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *Proc. 35th Int. Conf. Mach. Learn.*, PMLR, 2018, pp. 5636–5645.

[17] V. K. Pillutla, S. M. Kakade, and Z. Harchaoui, "Robust aggregation for federated learning," 2019, *arXiv: 1912.13445*.

[18] P. Blanchard, E. M. E. Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Proc. 31st Annu. Conf. Neural Inf. Process. Syst.*, 2017, pp. 119–129.

[19] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. B. Calo, "Analyzing federated learning through an adversarial lens," in *Proc. 36th Int. Conf. Mach. Learn.*, PMLR, 2019, pp. 634–643.

[20] R. G. Baraniuk, "Compressive sensing [lecture notes]," *IEEE Signal Process. Mag.*, vol. 24, no. 4, pp. 118–121, Jul. 2007.

[21] Y. Miao, R. Xie, X. Li, X. Liu, Z. Ma, and R. H. Deng, "Compressed federated learning based on adaptive local differential privacy," in *Proc. 38th Annu. Comput. Secur. Appl. Conf.*, 2022, pp. 159–170.

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

[23] H. Jia et al., "Proof-of-learning: Definitions and practice," in *Proc. IEEE Symp. Secur. Privacy*, 2021, pp. 1039–1056.

[24] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," in *Proc. 25th Eur. Symp. Res. Comput. Secur.*, 2020, pp. 480–501.

[25] H. Wang et al., "Attack of the tails: Yes, you really can backdoor federated learning," *Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 16 070–16 084, 2020.

[26] E. Rosenfeld, E. Winston, P. Ravikumar, and J. Z. Kolter, "Certified robustness to label-flipping attacks via randomized smoothing," in *Proc. 37th Int. Conf. Mach. Learn.*, 2020, pp. 8230–8241.

[27] T. Zheng and B. Li, "Poisoning attacks on deep learning based wireless traffic prediction," in *Proc. IEEE Int. Conf. Comput. Commun.*, 2022, pp. 660–669.

[28] V. Shejwalkar, A. Houmansadr, P. Kairouz, and D. Ramage, "Back to the drawing board: A critical evaluation of poisoning attacks on production federated learning," in *Proc. IEEE 43rd Symp. Secur. Privacy*, 2022, pp. 1354–1371.

[29] V. Shejwalkar and A. Houmansadr, "Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning," in *Proc. 28th Annu. Netw. Distrib. Syst. Secur. Symp.*, 2021, pp. 1–18.

[30] M. Fang, X. Cao, J. Jia, and N. Z. Gong, "Local model poisoning attacks to byzantine-robust federated learning," in *Proc. 29th USENIX Secur. Symp.*, 2020, pp. 1605–1622.

[31] D. Rong, S. Ye, R. Zhao, H. N. Yuen, J. Chen, and Q. He, "FedRecAttack: Model poisoning attack to federated recommendation," in *Proc. IEEE 38th Int. Conf. Data Eng.*, 2022, pp. 2643–2655.

[32] H. Liu, J. Jia, and N. Z. Gong, "PoisonEdencoder: Poisoning the unlabeled pre-training data in contrastive learning," in *Proc. 31st USENIX Secur. Symp.*, 2022, pp. 3629–3645.

[33] J. Jia, Y. Liu, and N. Z. Gong, "BadEncoder: Backdoor attacks to pre-trained encoders in self-supervised learning," in *Proc. IEEE 43rd Symp. Secur. Privacy*, 2022, pp. 2043–2059.

[34] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proc. 23rd Int. Conf. Artif. Intell. Statist.*, 2020, pp. 2938–2948.

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

[36] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proc. Int. Conf. Artif. Intell. Statist.*, PMLR, 2020, pp. 2938–2948.

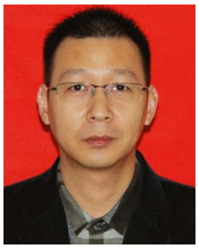
[37] A. Salem, R. Wen, M. Backes, S. Ma, and Y. Zhang, "Dynamic backdoor attacks against machine learning models," in *Proc. IEEE 7th Eur. Symp. Secur. Privacy*, 2022, pp. 703–718.

[38] C. Xie, K. Huang, P.-Y. Chen, and B. Li, "DBA: Distributed backdoor attacks against federated learning," in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 1–15.

[39] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

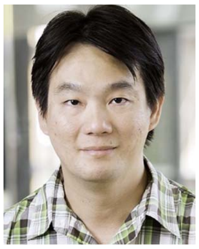
Yinbin Miao received the BE degree with the Department of Telecommunication Engineering from Jilin University, Changchun, China, in 2011, and the PhD degree with the Department of Telecommunication Engineering from Xidian University, Xian, China, in 2016. He is also a postdoctor in Nanyang Technological University from 2018 to 2019, and a postdoctor in City University of Hong Kong from 2019 to 2021. He is currently a professor with the Department of Cyber Engineering in Xidian University, Xian, China. His research interests include information security and applied cryptography.

Rongpeng Xie received the BE degree from the Department of Earth Sciences, Chengdu University of Technology, Chengdu, China, in 2021. He is currently working toward the ME degree with the Department of Cyber Engineering, Xidian University, Xian, China. His research interests include information security and applied cryptography.



Xinghua Li received the MS and PhD degrees in computer science from Xidian University, China, in 2004 and 2007, respectively. He is currently a professor with the School of Cyber Engineering, Xidian University. His research interests include wireless networks security, privacy protection, cloud computing, and security protocol formal methodology.

Zhiquan Liu (Member, IEEE) received the BS degree from the School of Science, Xidian University, Xi'an, China, in 2012, and the PhD degree from the School of Computer Science and Technology, Xidian University, Xi'an, China, in 2017. He is currently an associate professor with the College of Cyber Security, Jinan University, Guangzhou, China, and his current research focuses on trust management and privacy preservation in vehicular networks and UAV networks.



Kim-Kwang Raymond Choo (Senior Member, IEEE) received the PhD degree in Information Security from Queensland University of Technology, Australia, in 2006. He currently holds the Cloud Technology Endowed Professorship with The University of Texas at San Antonio. He is the founding co-editor-in-chief of *ACM Distributed Ledger Technologies: Research & Practice*, and the founding Chair of IEEE Technology and Engineering Management Society Technical Committee (TC) on Blockchain and *Distributed Ledger Technologies*. He is the recipient of the 2022 IEEE Hyper-Intelligence TC Award for Excellence in Hyper-Intelligence Systems (Technical Achievement award), the 2022 IEEE TC on Homeland Security Research and Innovation Award, the 2022 IEEE TC on Secure and Dependable Measurement Mid-Career Award, and the 2019 IEEE TC on Scalable Computing Award for Excellence in Scalable Computing (Middle Career Researcher).

Robert H. Deng (Fellow, IEEE) has been an AXA chair professor in cybersecurity and a professor in information systems with the School of Information Systems, Singapore Management University, since 2004. His research interests include data security and privacy, multimedia security, networks, and system security. He served/is serving on the editorial boards of many international journals, including *IEEE Transactions on Information Forensics and Security* and *IEEE Transactions on Dependable and Secure Computing*. He has received the Distinguished Paper Award (NDSS 2012), Best Paper Award (CMS 2012), and Best Journal Paper Award (IEEE Communications Society 2017).