Information Sciences 630 (2023) 158–172

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Model poisoning attack in differential privacy-based federated learning

Ming Yang [a], Hang Cheng [b,c,*], Fei Chen [a], Ximeng Liu [a,**], Meiqing Wang [b], Xibin Li [a]

[a] *College of Computer Science and Big Data, Fuzhou University, Fuzhou, Fujian 350108, China*
[b] *School of Mathematics and Statistics, Fuzhou University, Fuzhou, Fujian 350108, China*
[c] *Center for Applied Mathematics of Fujian Province, Fuzhou, Fujian 350108, China*

ARTICLE INFO

Keywords:
Privacy-preserving
Federated learning
Differential privacy
Model poisoning

ABSTRACT

Although federated learning can provide privacy protection for individual raw data, some studies have shown that the shared parameters or gradients under federated learning may still reveal user privacy. Differential privacy is a promising solution to the above problem due to its small computational overhead. At present, differential privacy-based federated learning generally focuses on the trade-off between privacy and model convergence. Even though differential privacy obscures sensitive information by adding a controlled amount of noise to the confidential data, it opens a new door for model poisoning attacks: attackers can use noise to escape anomaly detection. In this paper, we propose a novel model poisoning attack called Model Shuffle Attack (MSA), which designs a unique way to shuffle and scale the model parameters. If we treat the model as a black box, it behaves like a benign model on test set. Unlike other model poisoning attacks, the malicious model after MSA has high accuracy on test set while reducing the global model convergence speed and even causing the model to diverge. Extensive experiments show that under FedAvg and robust aggregation rules, MSA is able to significantly degrade performance of the global model while guaranteeing stealthiness.

1. Introduction

The rise of artificial intelligence has brought great convenience to people's lives. It has been widely applied in computer sciences fields, such as natural language processing [1], recommendation systems [2], and computer vision [3]. In these applications, it is necessary to rely on a large amount of data to train a specific neural network model. However, collecting data has become strict with improving privacy awareness and related laws and regulations. To address this problem, Google proposed federated learning in 2016 [4]. Briefly, there is a central server and multiple clients in a federated learning system. In each communication round, clients download the global model from the central server, train the model using their local data, and then upload the trained model parameters or gradients back to the central server. The central server updates the global model when receiving the models or gradients. As the communication rounds increase, the global model eventually converges. In short, the data does not move, the model moves, and clients only upload the model parameters or gradients.

* Corresponding author at: School of Mathematics and Statistics, Fuzhou University, Fuzhou, Fujian 350108, China.
** Corresponding author.
E-mail addresses: hcheng@fzu.edu.cn (H. Cheng), snbnix@gmail.com (X. Liu).

https://doi.org/10.1016/j.ins.2023.02.025
Received 11 August 2022; Received in revised form 29 November 2022; Accepted 8 February 2023

Available online 13 February 2023

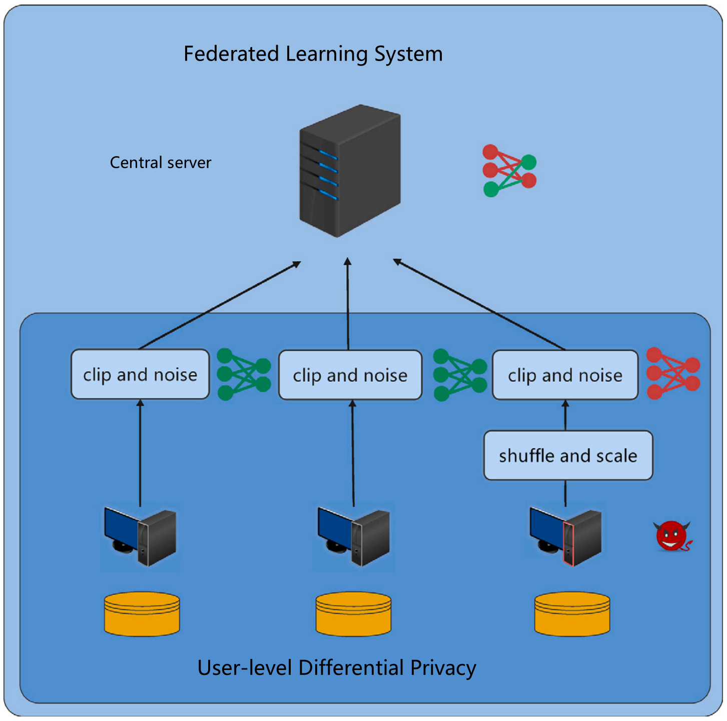


Fig. 1. Overview of the attack. Benign participants use their local data to train the model, clip the model, and add noise to achieve user-level differential privacy. The attacker compromises one or more participants, uses our shuffle-and-scale technique, and submits the resulting model, which destroys the global model in aggregation.

There is no doubt that federated learning provides a different way to protect privacy. Nevertheless, it faces many serious problems [5,6] at the same time, such as model poisoning attacks [7–10] and inference attacks [11–13]. The untargeted attack is one of the types of model poisoning attacks. With this attack, malicious clients send arbitrary or fake parameters to the central server to degrade the performance of the global model or diverge the global model. Inference attacks allow the attacker to recover the training data and corresponding labels using gradients [14]. Subsequently the relationship between the sign of the gradient and the data labels is also studied [15]. Therefore, it is necessary to protect the gradient from leaking sensitive information in federated learning.

Extensive research has been conducted on these sensitive information leakage problems. At present, the common privacy-preserving techniques include secure multi-party computation [16], homomorphic encryption [17], and differential privacy [18]. The former two have massive computational overhead, while differential privacy only achieves privacy protection by perturbing the original data, and the computational overhead is relatively small. Furthermore, the attacker cannot fully recover the original data under differential privacy compared with encryption-based privacy protection schemes. More importantly, differential privacy can always balance accuracy and privacy. In federated learning, differential privacy can be divided into instance-level differential privacy [19–21] and user-level differential privacy [22–24]. The former protects each instance of the user's dataset, while the latter safeguards the whole data of the user. And our work investigates poisoning attacks under user-level differential privacy. Before clients upload the model to the central server, each client can add a specific random noise to the local model to achieve user-level differential privacy.

Although differential privacy can protect private data by adding noise, it also opens a new door for model poisoning attacks: attackers may exploit noise to evade anomaly detection. Inspired by the potential risk, we propose a new model poisoning attack called Model Shuffle Attack (MSA). Fig. 1 gives a high-level overview of this attack. The key insight of our attack is that by shuffling and scaling model parameters to generate a malicious model, which differs from the benign model in parameter values and locations. Such model can degrade the performance of the global model or slow down the convergence speed after the aggregation. Unlike other model poisoning attacks, our proposed scheme can attack the global model while ensuring the stealth of the malicious model.

Our main contributions are as follows:

- We propose a novel model poisoning attack called Model Shuffle Attack (MSA). This model poisoning attack can effectively slow down the global model's convergence rate or degrade the global model's performance.
- Our proposed MSA has high stealth. With this attack, the model's accuracy on test set remains unchanged after shuffling the model parameter positions. At the same time, scaling the model parameters overcomes the influence of other models in the aggregation process, making the overall model parameter distribution similar to that before scaling.
- Extensive experiments show that the model modified by our MSA can significantly degrade performance of the global model while maintaining high accuracy on test set. Besides, our MSA can disable the robust aggregation rules, Krum and Trimmed Mean.

2. Related work

2.1. Poisoning attacks and defenses in federated learning

Different from traditional centralized machine learning, federated learning combines multiple trainers to obtain a global model while preserving the privacy of the data. However, federated learning suffers from various model poisoning attacks. For example, attackers can manipulate model parameters or gradients sent back to the central server and then damage the global model during aggregation. The poisoning attacks on federated learning systems can be roughly divided into untargeted attacks [8,25] and targeted attacks [10,26,27]. Untargeted attacks aim to slow down the convergence rate of the global model or even make the model diverge. These powerful attacks are also known as the Byzantine attacks [28]. Although Byzantine attacks can cause massive damage to the global model, the attacker must have sufficient knowledge of the central server aggregation rules. Otherwise, these attack methods may have little effect. The goal of the targeted attack is to make the behavior of malicious model fit the attacker's expectations. As a typical targeted attack, label-flipping [7] changes the label of a specific class in the dataset, that is, the label of one class is modified to another. A malicious model trained on such a dataset will have lower accuracy on the attacked class, while the accuracy on other classes will not be affected. A backdoor attack is also a type of targeted attack. Unlike the label-flipping attack, the attacker must construct an additional training data set, a trigger set, using specific patterns. After training on the dataset and trigger set, the model is embedded with a backdoor. Such models perform well (high accuracy) when input data does not have a specific pattern. Once input data with a specific pattern (input trigger set), the model gives an abnormal output. C. Xie proposed a distributed backdoor attack (DBA) [26] in federated learning, which is more stealthy than centralized poisoning. Generally speaking, targeted attacks are more difficult to detect than untargeted attacks, especially for backdoor attacks, since the prediction results of each class have a high accuracy before no trigger set is taken as input.

FedAvg [4] is a standard server aggregation scheme in federated learning, but this aggregation rule is quite vulnerable to attacks. Some work has proposed robust aggregation algorithms against Byzantine attacks, such as Krum [29], which computes the distance between each model and gives a model that is most similar to others, treating the model as a global one. Although Krum has strong robustness, it is difficult to apply due to the large differences between models, especially under Non-IID condition. Poisoning attacks, such as Byzantine attacks and label-flipping attacks, reduce the model's accuracy. The central server can test the accuracy of the model on test set before aggregation, or identify whether it is a malicious model by analyzing parameter statistical information. For some targeted attacks, such as backdoor attacks, some outlier detection algorithms [30,31] were proposed to distinguish whether a model is embedded with a backdoor, and then they used inverse triggers to eliminate the backdoor. Fung et al. proposed FoolsGold [32] to adjust the learning rate of each client according to the contribution similarity, which resists label-flipping attacks and backdoor attacks.

2.2. Differential privacy in federated learning

Depending on the data collection methods, differential privacy can be classified into centralized differential privacy and local differential privacy. Centralized differential requires a trusted third-party data collector to perform the privacy processing of the data. For local differential privacy (LDP), each user can process individual data independently, which means that the privacy process is transferred from the data collector to the individual user, thus eliminating the need for a trusted third party to intervene and potential privacy attacks from untrusted third-party data collectors. In federated learning, instance-level differential privacy and user-level differential privacy should be considered. The concept of user-level differential privacy was introduced in [33], and the authors proposed DP-FedAvg, which guarantees user-level differential privacy through Gaussian mechanisms. In [24], K. Wei et al. proposed a new user-level differential privacy algorithm, finds an optimal number of communication rounds for a given level of privacy and designs communication rounds discounting (CRD) methods to better balance complexity and convergence. In [34], Cheng et al. proposed Bounded Local Update Regularization (BLUR) and Local Update Sparsification (LUS) to address model degradation caused by clipping and adding noise. Existing work on differential privacy-based federated learning generally focuses on the trade-off between privacy and model convergence.

3. Preliminaries

In this section, we illustrate the differential privacy-based federated learning setup, make assumptions about the attacker's capabilities and goals, and explore the defenses that a central server might take when aggregating models. A summary of main notations is listed in Table 1.

Furthermore, we assume that the central server is honest and adopts the FedAvg algorithm and two robust aggregation algorithms (Krum, Trimmed Mean) as the aggregation rules for the central server. Here, a total of N clients jointly train a global model. In each round of communication t, the central server randomly selects K clients from N clients to distribute the global model. Each client i receives the global model w_t from the central server, trains it with local privacy data $\mathcal{D}_i$, and then uploads the local model w_t^i. After receiving the local models $\{w_t^1, w_t^2, \dots w_t^k, \dots w_t^K\}$ from K clients, the central server aggregates the models to generate a new global model w_{t+1}:

$$w_{t+1} = \mathcal{A}(w_t^1, w_t^2, \dots w_t^k, \dots w_t^K), \tag{1}$$

Table 1
Summary of Main Notation.

ϵ, δ	The parameters related to LDP
ϵ_i, δ_i	The parameters related to LDP for the i-th client
$\mathcal{D}_i$	The database held by the client i
N	Total number of all clients
K	The number of chosen clients $(1 < K < N)$
t	The index of the t-th communication round
T	The number of communication round
w_t	The global model in t-th communication round
$W^{[i]}$, $b^{[i]}$	Weight and bias in certain layer i
F^i	Loss function from the i-th client
q	The sample rate
w_t^k	Client k's model in t-th communication round
$\hat{w}_t^k$	Model after clipping
$\widetilde{w}_t^k$	Model after adding noise
C	Clipping threshold
c	The number of malicious clients
ϵ_b	Benign client's privacy budget
ϵ_m	Malicious client's privacy budget
$\lvert \cdot \rvert$	The cardinality of a set
$\lVert \cdot \rVert_2$	l_2 norm
α	Scaling factor in convolutional layer
β	Scaling factor in fully connected layer
$\mathcal{A}$	Aggregation method

where w_t^k is the model parameter uploaded by the k-th client in the t-th round. As the number of communication rounds increases, the global model converges.

In federated learning, although the clients' private data does not leave the local when clients and the central server communicate, the attacker may intercept models and use them to infer the private information. Fig. 1 shows the user-level differential privacy in federated learning. Before uploading the model to the central server, clients need to clip the norm of the model parameters to limit their norm to the threshold C and add noise to the model parameters w_t^k according to their privacy budget to prevent the leakage of sensitive information:

$$\begin{aligned} \hat{w}_t^k &= w_t^k / \max\left(1, \frac{\left\lVert w_t^k \right\rVert_2}{C}\right), \\ \widetilde{w}_t^k &= \hat{w}_t^k + noise, \end{aligned} \tag{2}$$

where $\hat{w}_t^k$ is the model after clipping, and $\widetilde{w}_t^k$ is the model that the client eventually uploads to the server. Furthermore, the user-level differential privacy algorithm UDP [24] is used in this paper. The involved standard deviation of Gaussian noise for each client i satisfies (ϵ_i, δ_i)-LDP, namely,

$$\sigma_i = \frac{\Delta\ell\sqrt{2qT\ln\left(1/\delta_i\right)}}{\epsilon_i}, \tag{3}$$

where q represents the sampling ratio K/N, T is the total number of rounds of communication, and $\Delta\ell$ indicates the sensitivity of local training. In [24], the author proves that $\Delta\ell$ meets the following condition:

$$\Delta\ell \le \frac{2\eta C}{\lvert \mathcal{D}_i \rvert}, \tag{4}$$

where η is the learning rate of the local client, and $\lvert \mathcal{D}_i \rvert$ is the number of local samples.

Attacker's knowledge: We assume that the attacker can control one or more clients, these controlled ones are malicious clients, and the rest are benign clients. Under the assumption, the attacker has access to the local data of these malicious clients and can modify the training hyperparameters, such as learning rate η and privacy budget ϵ, but has no knowledge of the configuration of the benign clients and cannot modify them. In addition, the attacker knows the aggregation rules of the central server and the total number of rounds of communication. This knowledge can allow attackers to do great harm to federated learning systems.

Attacker's goals: The proposed MSA is an untargeted attack. Once these malicious models participate in the aggregation process, they can significantly degrade performance of the global model and reduce the rate of model convergence, even causing the model convergence to fail.

Central server defense: The central server adopts two security checks (model performance evaluation and parameter information analysis) and robust aggregation rules as defensive measures. Although the defenses are not necessary for standard federated learning, the central server must take relevant defenses to resist poisoning attacks from clients.

Model performance evaluation: After receiving models from clients, the central server can evaluate the performance of these models on test set. If the model has low accuracy on test set, the central server will reject the model from participating in aggregation.

Parameter information analysis: The central server can analyze the parameter information of the models. Suppose a model with parameter information significantly differs from other models. It is reasonable for the central server to determine that the model is malicious and excludes this model from aggregation.

Robust aggregation rules: The central server uses Krum and Trimmed Mean robust aggregation rules as a defense. There are assumed to have c malicious clients for Krum. When receiving the models from K clients, the central server calculates the Euclidean distance between a given model and other models and assigns the model's score with the sum of $K-2-c$ smallest distances. The model with the minor score is defined as the global model. Based on the Trimmed Mean rule, the central server sorts each parameter j in the model from smallest to largest, removes the smallest and largest c values, and uses the average of the remaining $K-2c$ parameters as the global model for parameter j.

4. Proposed method

In this section, we propose a novel model poisoning attack, MSA (Model Shuffle Attack), outlined in **Algorithm 1**. It can attack a federated learning system based on differential privacy.

Algorithm 1: MSA in DP-based federated learning. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, η is the learning rate, and $\mathcal{A}$ is the aggregation rule.

```
Server executes:
initialize $w_0$
for each round $t = 1,2,...T$ do
    $K \leftarrow q\cdot N$
    $S_t \leftarrow$ (random set of $K$ clients)
    $w_t \leftarrow w_{t-1}$
    for each client $k \in S_t$ in parallel do
        $w_{t+1}^k \leftarrow$ ClientUpdate $(k, w_t)$
    end
    $w_{t+1} \leftarrow \mathcal{A}(w_{t+1}^1, w_{t+1}^2, ...w_{t+1}^k, ...w_{t+1}^K)$
end

ClientUpdate($k$, $w_t$):
$w_t^k \leftarrow w_t$
$\mathcal{B} \leftarrow$ (split $\mathcal{D}_k$ into batches of size $B$)
for each local epoch $i$ from 1 to $E$ do
    for batch $b \in \mathcal{B}$ do
        $w_t^k \leftarrow w_t^k - \eta \nabla F^k(w_t^k; b)$
    end
end
if client $k$ is malicious: then
    Perform MSA: shuffle the position of the model parameters, and scale the parameters
end
$\hat{w}_t^k = w_t^k / \max\left(1, \frac{\|w_t^k\|_2}{C}\right)$
Calculate $\sigma_i$ according to LDP parameters ($\epsilon_i$, $\delta_i$):
$\widetilde{w}_t^k = \hat{w}_t^k + \mathcal{N}\left(0, \sigma_i^2 \mathbf{I}\right)$
return $\widetilde{w}_t^k$ to server
```

4.1. Attack model

As we all know, multiple clients jointly train a global model in federated learning. Intuitively, as the number of training rounds increases, the model parameters of each local client tend to be similar. Inspired by the result, one can shuffle the order of kernels in the convolutional layer, neurons in the fully connected layer, and scale the model parameters. These shuffling and scaling operations will generate models that are different from other benign models. Thus, the global model is affected when aggregating.

Fig. 2 shows the shuffling and scaling operations in convolutional layers. In Fig. 2(a), take an image of size $28\times28\times1$ as input, and suppose the first convolutional layer has three kernels with size $3\times3\times1$. In Fig. 2(b), if we shuffle the order of the kernels in the first convolutional layers, the generated $feature\,maps\,1'$ is also in shuffled order. To ensure that a model before and after shuffling remains the accuracy unchanged on test set, we can shuffle the channels inside each kernel $(3\times3\times3)$ in the next convolutional layer in the same order so that the generated $feature\,maps\,2$ is the same as the benign model. As shown in Fig. 2(b), if we multiply each kernel by a scaling factor α in the first convolutional layer:

$$\alpha = \left(\alpha_1, \alpha_2 ... \alpha_m\right), \tag{5}$$

where m is the number of kernels, and each kernel i corresponds to an α_i that satisfies $\alpha_i > 0$. Then the $feature\,map\,1'$ output is also scaled. In order to achieve the same $feature\,map\,2$ as the benign model, we multiply each kernel by $1/\alpha$ in the next convolutional layer.

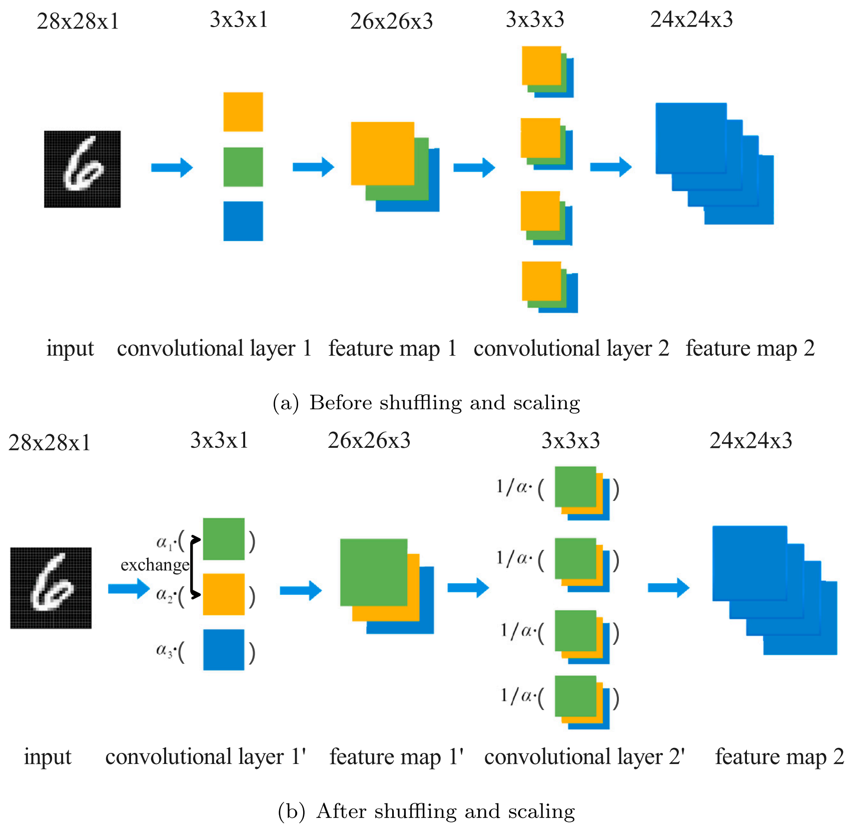


Fig. 2. Example illustration of shuffling and scaling operations in convolutional layers. (a) is before shuffling and scaling operations. (b) is after shuffling and scaling operations. For shuffling, the attacker exchanges the first and the second kernel in convolutional layer 1, then exchanges each kernel inside in the same order in convolutional layer 2. For scaling, the attacker multiplies a scaling factor α_i for each kernel in convolutional layer 1', then multiplies a scaling factor $1/\alpha$ for each kernel in convolutional layer 2', where $\alpha = (\alpha_1, \alpha_2 \dots \alpha_m)$.

Fig. 3 illustrates the shuffling operations in fully connected layers. According to the input x, the activation value $a^{[1]}$ of the $hidden\,layer\,1$ can be calculated, then the activation value $a^{[2]}$ of $hidden\,layer\,2$ can be calculated based on $a^{[1]}$:

$$\begin{aligned} a^{[1]} &= \mathrm{ReLU}\left(W^{[1]}x + b^{[1]}\right), \\ a^{[2]} &= \mathrm{ReLU}\left(W^{[2]}a^{[1]} + b^{[2]}\right). \end{aligned} \tag{6}$$

If we shuffle $W^{[1]}$ and $b^{[1]}$ by row and shuffle $W^{[2]}$ by column in the same order, the model after shuffling outputs the same activation value $a^{[2]}$ as before:

$$\widetilde{W^{[1]}} = \begin{bmatrix} \boldsymbol{w^1_{21}} & \boldsymbol{w^1_{22}} & \boldsymbol{w^1_{23}} & \boldsymbol{w^1_{24}} & \boldsymbol{w^1_{25}} \\ \boldsymbol{w^1_{11}} & \boldsymbol{w^1_{12}} & \boldsymbol{w^1_{13}} & \boldsymbol{w^1_{14}} & \boldsymbol{w^1_{15}} \\ w^1_{31} & w^1_{32} & w^1_{33} & w^1_{34} & w^1_{35} \\ w^1_{41} & w^1_{42} & w^1_{43} & w^1_{44} & w^1_{45} \end{bmatrix}, \tag{7}$$

$$\widetilde{b^{[1]}} = \begin{bmatrix} \boldsymbol{b^1_2} \\ \boldsymbol{b^1_1} \\ b^1_3 \\ b^1_4 \\ b^1_5 \end{bmatrix}, \tag{8}$$

$$\widetilde{W^{[2]}} = \begin{bmatrix} \boldsymbol{w^2_{12}} & \boldsymbol{w^2_{11}} & w^2_{13} & w^2_{14} \\ \boldsymbol{w^2_{22}} & \boldsymbol{w^2_{21}} & w^2_{23} & w^2_{24} \\ \boldsymbol{w^2_{32}} & \boldsymbol{w^2_{31}} & w^2_{33} & w^2_{34} \end{bmatrix}. \tag{9}$$

Like the scaling operations in the convolution layer, we can also multiply the fully connected parameters by a scaling factor β:

$$\beta = (\beta_1, \beta_2 \dots \beta_m), \tag{10}$$

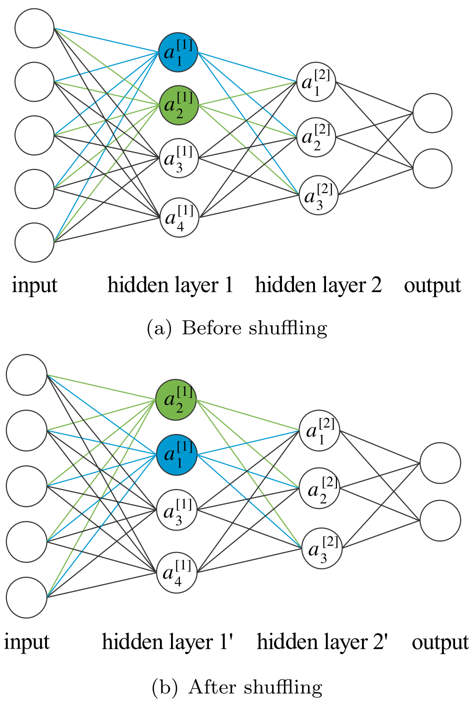


Fig. 3. Shuffling operations in fully connected layers. (a) is before shuffling operations. (b) is after shuffling operations. The attacker exchanges the first and the second row in $W^{[1]}$ and $b^{[1]}$, then the order of the activation values in hidden layer 1' is also changed. The attacker exchanges the first column and the second column in $W^{[2]}$ to keep the activation value of the hidden layer 2' the same as the hidden layer 2.

where $\beta_i (\beta_i > 0)$ is the scaling factor for each row in W. Then we multiply by $1/\beta$ in the next fully connected layer to achieve the same activation value $a^{[2]}$ as before:

$$\begin{aligned}
\widetilde{a^{[1]}} &= \text{ReLU}\left(W^{[1]}\beta x + b^{[1]}\beta\right) \\
&= \beta \cdot \text{ReLU}\left(W^{[1]}x + b^{[1]}\right) \\
&= \beta \cdot a^{[1]}, \\
a^{[2]} &= \text{ReLU}\left(W^{[2]}(1/\beta)\widetilde{a^{[1]}} + b^{[2]}\right) \\
&= \text{ReLU}\left(W^{[2]}(1/\beta)(\beta)a^{[1]} + b^{[2]}\right) \\
&= \text{ReLU}\left(W^{[2]}a^{[1]} + b^{[2]}\right).
\end{aligned} \tag{11}$$

If we treat the neural network as a black box, given an input, the output of the malicious and benign models are the same. The only difference is that the parameters inside the network are shuffled and scaled. However, the local model, after shuffling and scaling, will cause the global model to deviate from the benign model.

4.2. Bypass security checks

We set up two security checks: model performance evaluation and parameter information analysis. Although these two operations are not necessary for federated learning, they offer protection against poisoning attacks.

Model performance evaluation: According to the calculation process of the feedforward neural network, the output of the malicious model obtained by MSA is the same as the original model. Even though the model parameters are shuffled and scaled, the accuracy on test set is consistent with the original model. It thus can easily evade model performance evaluation.

Parameter information analysis: It should be noted that the shuffling operations in MSA are performed at one layer of the model. If the central server analyzes the distribution of parameters at each layer of the model, it cannot distinguish which ones are malicious because the parameter distribution does not change. In MSA, the scaling of model parameters can better overcome the impacts of benign models on malicious models during aggregation. Nevertheless, the scaling operations may destroy the parameter distribution of the model. It has experimentally demonstrated that malicious models can avoid detection in the following ways: First, choose a small scaling factor. Although the effectiveness of the attack may be weakened, it still achieves a better attack effect as the number of malicious clients increases. Second, keep a similar distribution of parameters in a specific layer. It can be achieved by controlling the scaling ratio in different parts of the parameters. Last but not least, a threshold is set to limit the magnitude of the parameters to prevent them from being easily detected.

Table 2
The CNN architecture of the global model used for FeMnist and CIFAR-10.

Models	Layers	Size
FeMnist-CNN	Input	28x28x1
	Convolution + ReLU	5x5x32
	Max Pooling	2x2
	Convolution + ReLU	5x5x64
	Max Pooling	2x2
	Fully Connected + ReLU	512
	Fully Connected + ReLU	62
CIFAR-CNN	Input	32x32x3
	Convolution + ReLU	3x3x64
	Max Pooling	2x2
	Convolution + ReLU	3x3x128
	Max Pooling	2x2
	Convolution + ReLU	3x3x256
	Max Pooling	2x2
	Fully Connected + ReLU	128
	Fully Connected + ReLU	256
	Fully Connected + ReLU	10

4.3. *Attack on robust aggregation*

In addition to FedAvg, we also use two robust aggregation rules, Krum and Trimmed Mean, to demonstrate the effectiveness of our MSA.

It is known to all that Krum selects models based on model similarity. However, the malicious model generated after MSA is no longer the same as the benign model because the parameters have been shuffled and scaled. It implies that the Euclidean distance between the malicious and benign models is larger than between benign models. Therefore, the malicious model is difficult to be selected under the Krum aggregation rule. To enable the malicious model to be captured by Krum, we propose the following solutions. The attacker increases the privacy budget of the malicious model so that the privacy budget ϵ_m (ϵ_b) of the malicious (benign) model meets the requirement of $0 < \epsilon_b < \epsilon_m$. It means the magnitude of the noise added to the malicious model is smaller than the benign model. Thereby, the distance between the malicious and benign models is similar to that between benign models. Also, because the malicious models are supposed to keep the same update, the Euclidean distance between them is 0. In this case, the malicious models are highly likely to be hit based on the Krum rule. It implies that our MSA can resist Krum's robust aggregation.

The same idea is used for Trimmed Mean, and the attack results are given in the experimental section.

5. Evaluation

5.1. *Experimental details*

Datasets: FeMnist and CIFAR-10. FeMnist is an extension of Mnist, which has 62 different handwritten digits and letters (digits 0 to 9, 26 lowercase letters, and 26 uppercase letters). The dataset contains handwritten numbers and letters from 3550 users with a total number of images of 805,263 and an average number of 226.83 per user. CIFAR-10 is a dataset with 10 categories (e.g., airplanes, horses, boats, etc.), which contains 5,0000 training images and 1,0000 test images with three channels and the size of 32x32.

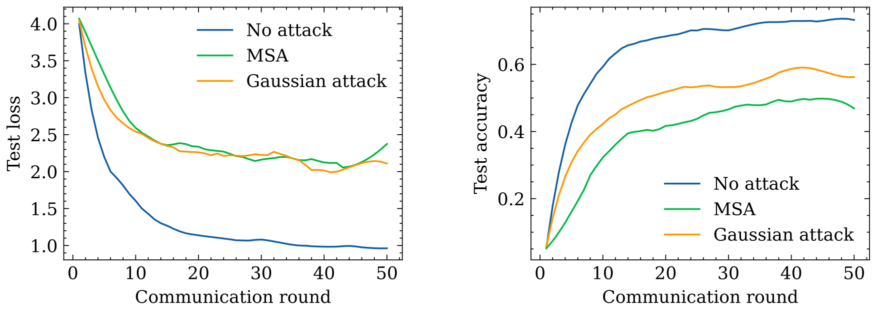
Models: Both FeMnist and CIFAR-10 use CNN, and the structure of CNN is shown in Table 2. If not specially stated, for FeMnist-CNN and CIFAR-CNN, the attacker shuffles all nodes in a particular convolutional layer and fully connected layer by default.

Compared attacks: Following [8], Gaussian attack is taken as a comparison to evaluate the effectiveness of our MSA. Gaussian attack uses local data to train the model and then estimates the Gaussian distribution of the model. Each parameter in the model is replaced with a number randomly sampled from the Gaussian distribution of the model, and finally, a malicious model is created.

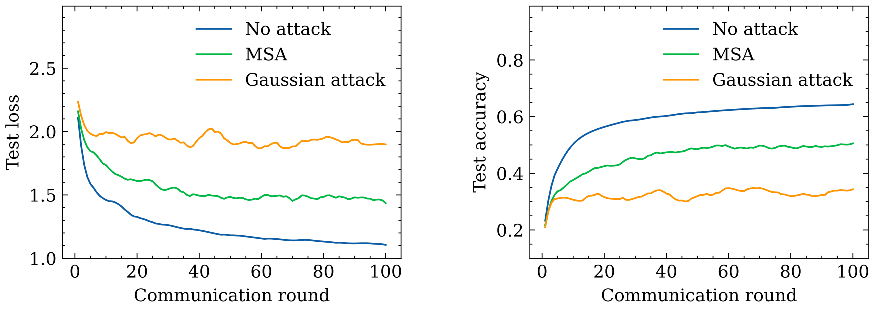
Parameter settings: In this federated learning system, the total number of clients $N = 100$ and the sampling rate $q = 0.1$, namely, ten clients participate in each round of training. For the FeMnist and CIFAR-10, set the total number of communication rounds $T = 50, 100$. The local epoch of clients for FeMnist / CIFAR-10 is 5 / 7. The optimizer adopts adam, and the learning rate is 0.001. For FeMnist / CIFAR-10, we set $\delta = 0.01$, the privacy level ϵ is 20 / 40, and the clipping threshold C is 20 / 40. MSA increases the malicious client's privacy level to reduce the malicious model's noise magnitude while maintaining the same privacy level as the baseline for Gaussian attack. Besides, the scaling factors α, β are set between 0.5 and 2.

5.2. *Attack FedAvg*

Fig. 4 shows the loss values and accuracy rates (compromised rate is 20%) of our MSA and Gaussian attack on FeMnist and CIFAR-10. Both attacks can successfully attack FedAvg and degrade the performance of the global model. Among them, MSA



(a) Test loss and Test accuracy on FeMnist



(b) Test loss and Test accuracy on CIFAR-10

Fig. 4. Evaluation of MSA and Gaussian attack under FedAvg, compromised rate is 20%. (a) attacks on FeMnist, benign client $\epsilon_b = 20$, malicious client $\epsilon_m = 50$; (b) attacks on CIFAR-10, benign client $\epsilon_b = 40$, malicious client $\epsilon_m = 100$.

outperforms Gaussian attack on the FeMnist dataset, while Gaussian attack is relatively better on CIFAR-10. Further, the impact of different compromised rates on the global model performance is illustrated in Fig. 5. As the compromised rate increases, the performance of the global model drops sharply. It can be found that in Fig. 5(a), MSA is more destructive to the global model than Gaussian attack at different compromised rates. However, in Fig. 5(b), the destructiveness of MSA is not as great as that of Gaussian attack. This result is because MSA has to make a trade-off between stealthiness and destructiveness.

Although Gaussian attack is better than MSA on the CIFAR-10, it can be easily detected if the server takes relevant security checks such as model performance assessment. The reason is, in Gaussian attack, the parameters of malicious model are randomly sampled from a Gaussian distribution, and the model has low accuracy. In contrast, our MSA is able to ensure the malicious model has a high accuracy while still corrupting the performance of the global model. Here, we counted the accuracy ranking of the malicious models among the models involved in each round of aggregation and calculated the average accuracy ranking throughout the training process. As shown in Fig. 6, the average ranking of the Gaussian attack accuracy is above eight for different compromised rates (ten models in total). In difference, in our proposed MSA, the average ranking of accuracy stays at a better level (staying in the top six). These are mainly due to MSA's special way of shuffling and scaling, and the higher privacy budget ϵ_m is less harmful to the model's performance. MSA can do damage to the global model while maintaining high accuracy on the test set.

Moreover, we explored the effect of the number of nodes shuffled on the global model. We compared the effect of shuffling 25%, 50%, 75%, and 100% of the number of nodes on the attack performance. As shown in Fig. 7, the more nodes shuffled, the worse the global model performance.

Obviously, the shuffling of parameters in MSA does not destroy the original model parameter distribution. However, MSA involves the scaling of parameters. In order to deceive parameter information checks of the central server, the scaling factor of half of the convolutional/fully connected layer can be set between 1 and 2 for scaling up, and the remaining half can be set between 0.5 and 1 for scaling down. Scale-up and scale-down operations are taken on the same layer to reduce the impact of the scaling process on the parameter distribution of the whole model. Fig. 8 gives the comparison of the distribution of the MSA-based model and the benign model. It can be observed that the parameter distribution of the malicious and benign models are similar. Thence, MSA can achieve great stealthiness because of the balance between scaling up and scaling down.

5.3. Attack Krum

We compared the effectiveness of MSA with Gaussian attack under the Krum aggregation rule (compromised rate is 20%). As shown in Fig. 9, MSA is more likely to escape Krum detection than Gaussian attack both on FeMnist and CIFAR-10. The main reason is that MSA makes the Euclidean distance between the malicious and benign models similar to one between the benign models by

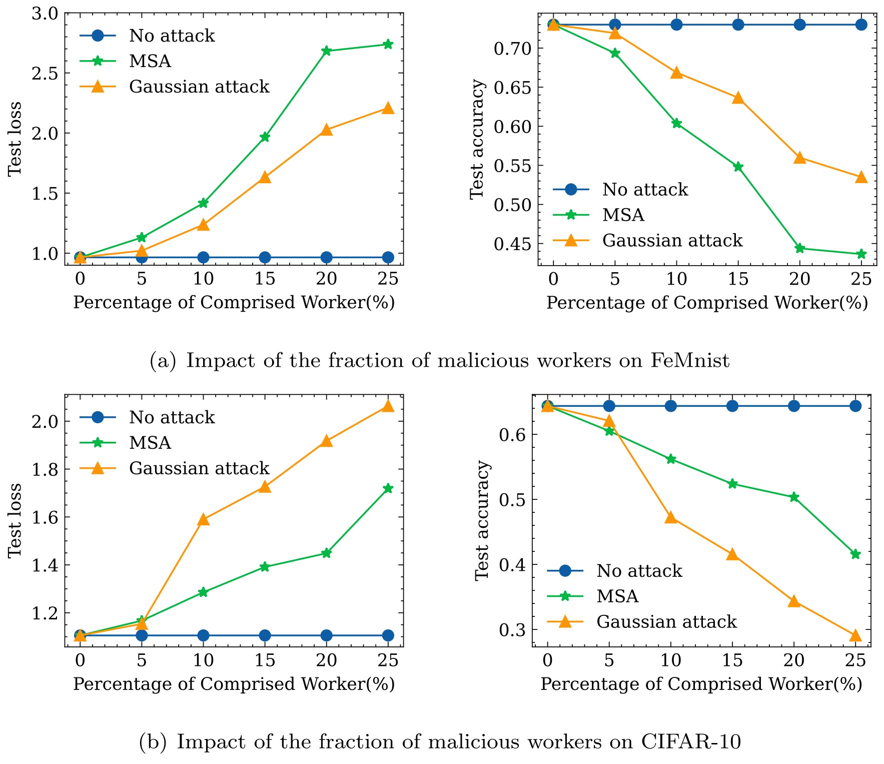


Fig. 5. Impact of the fraction of malicious works under MSA and Gaussian attack. The aggregation rule is FedAvg: (a) global model test loss and accuracy on FeMnist after training; (b) global model test loss and accuracy on CIFAR-10 after training.

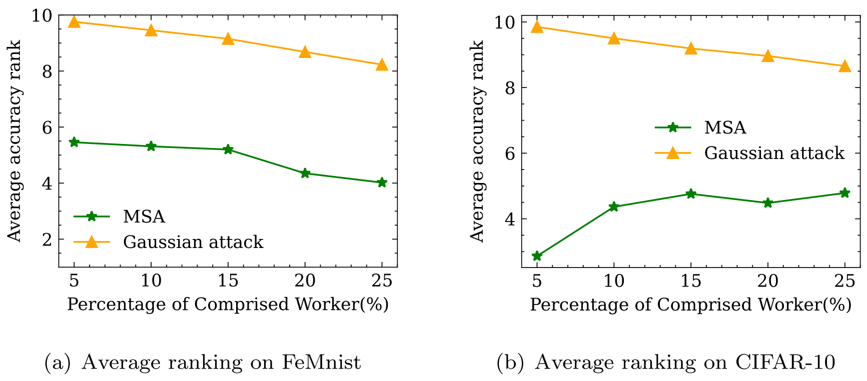


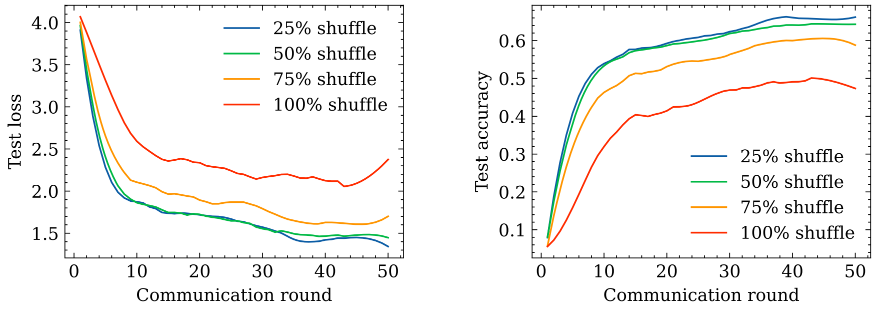
Fig. 6. MSA vs. Gaussian attack: average accuracy ranking of malicious models throughout the whole training process. The aggregation rule is FedAvg. (a) average accuracy ranking under different compromised rated on FeMnist; (b) average accuracy ranking under different compromised rated on CIFAR-10.

increasing the privacy budget ϵ_m. Meanwhile, the models after MSA keep the same update so that the distance between the malicious models is 0, which increases the probability of malicious models being selected by Krum.

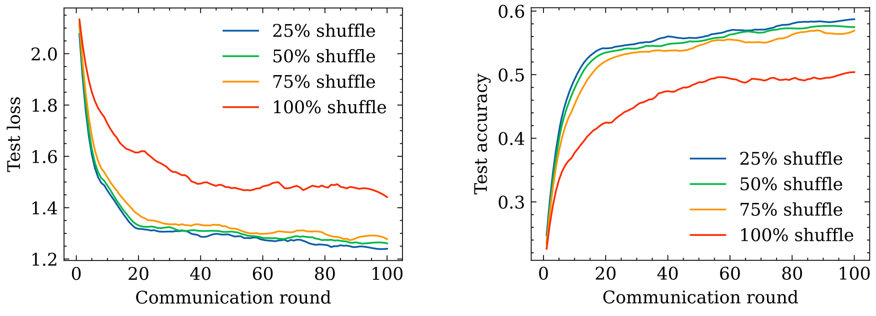
Fig. 10 depicts the effectiveness of the attacks of the two methods at different compromised rates and the average ranking of the accuracy of the malicious models. Under the Krum aggregation rule, Gaussian attack does not have much influence on the global model, while MSA yields better results than Gaussian attack on both FeMnist and CIFAR-10. It is clear that our MSA can harm the performance of the global model while guaranteeing the performance of the malicious models, whereas the Gaussian attack cannot achieve.

5.4. Attack Trimmed Mean

Fig. 11 provides the comparison of the effectiveness of the two attacks under the Trimmed Mean aggregation rule. It is noticed that both MSA and Gaussian attack are effective for Trimmed Mean. On FeMnist, both attacks slow down the speed of global model

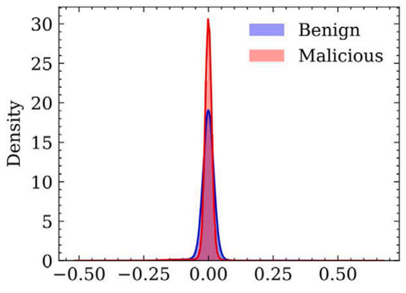


(a) Impact of the fraction of nodes shuffled on FeMnist

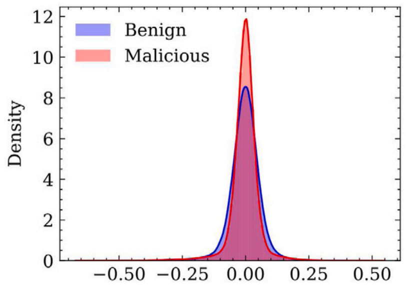


(b) Impact of the fraction of nodes shuffled on CIFAR-10

Fig. 7. Impact of the fraction of nodes shuffled under FedAvg, compromised rate is 20%. (a) global model test loss and accuracy on FeMnist under different fraction of nodes shuffled; (b) global model test loss and accuracy on CIFAR-10 under different fraction of nodes shuffled.



(a) Distribution on FeMnist



(b) Distribution on CIFAR-10

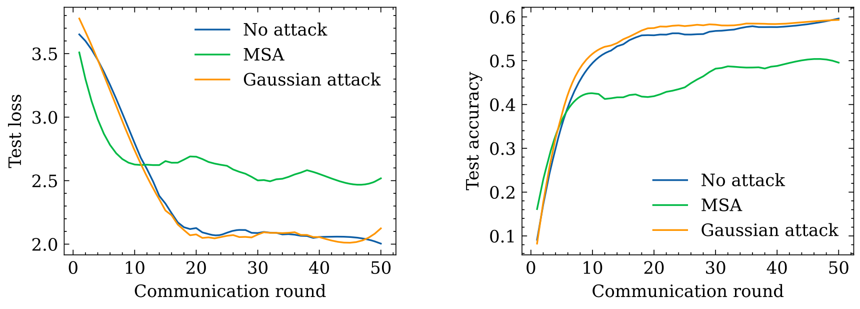
Fig. 8. Comparison of parameter distribution for benign and malicious models. Half of the values of α and β are set between 1 and 2, and the other half are set between 0.5 and 1.

convergence, and MSA is more destructive than the Gaussian attack. On CIFAR-10, both methods make the global model challenging to converge, and Gaussian attack exhibits greater destructiveness than MSA.

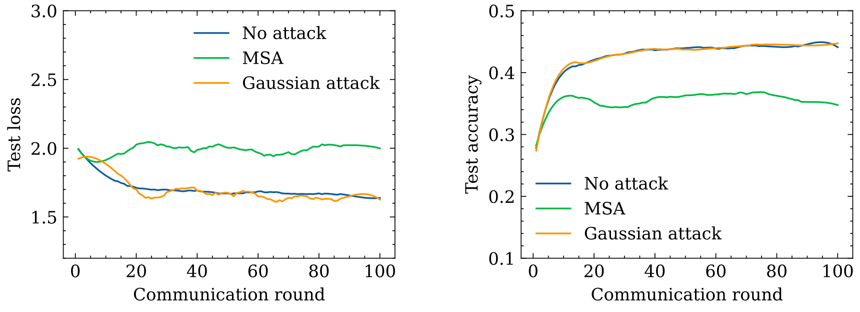
To further verify the degradability of both attacks, we tested the effectiveness of the two attack methods at different compromised rates and calculated the average accuracy ranking of the malicious models. As shown in Fig. 12, MSA offers stronger destructiveness than Gaussian attack on FeMnist in different compromised rates. However, MSA is not as destructive as Gaussian attack on CIFAR-10, but Fig. 12(e) and Fig. 12(f) indicate that MSA makes a trade-off between aggressiveness and stealthiness.

6. Conclusion and future work

In this paper, we explore the possibility of model poisoning in differential privacy-based federated learning and propose a novel model poisoning attack called MSA. The critical difference between our proposed MSA and existing approaches for model poisoning attacks is that if we treat the model as a black box, then the model after MSA behaves no differently from the benign model on test set. Such a unique property means our MSA has more vital stealth than other attacks. Extensive experiments are conducted to demonstrate that our proposed MSA can invalidate the global model while guaranteeing the performance of the malicious model. The experiments also show that our proposed MSA can attack robust aggregation rules, such as Krum and Trimmed Mean. Future

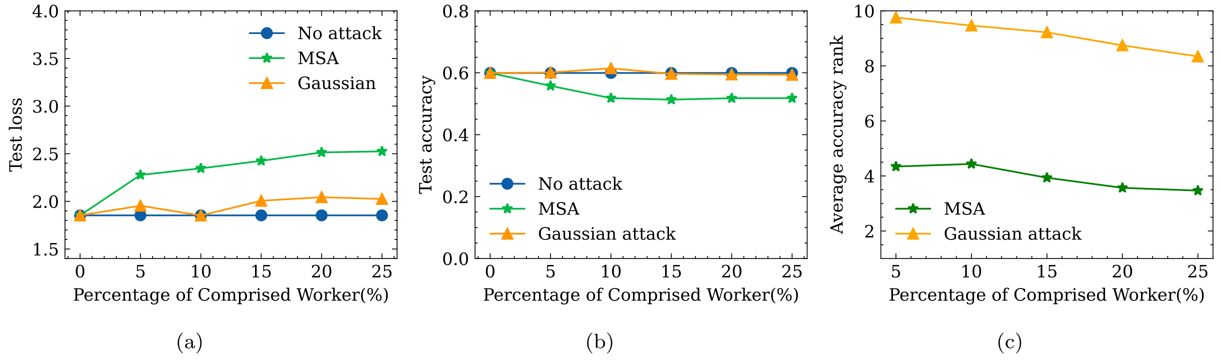


(a) Test loss and Test accuracy on FeMnist

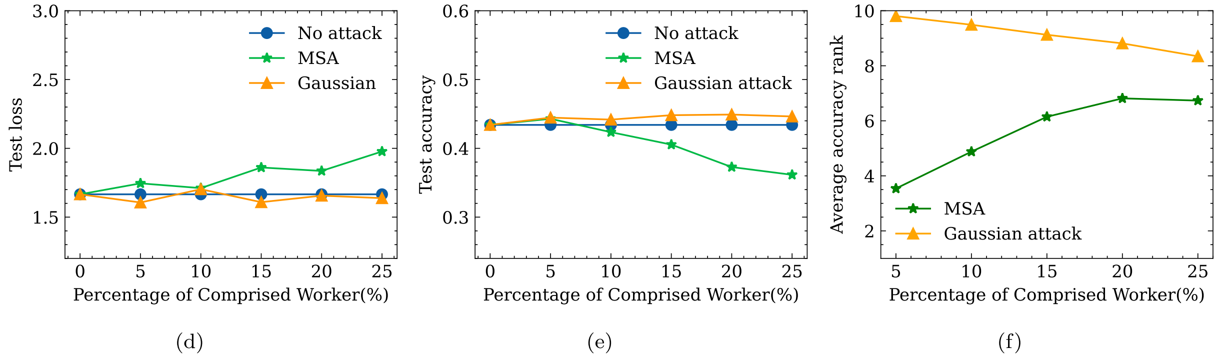


(b) Test loss and Test accuracy on CIFAR-10

Fig. 9. Evaluation of MSA and Gaussian attack under Krum, compromised rate is 20%. (a) attacks on FeMnist, benign client $\epsilon_b = 55$, malicious client $\epsilon_m = 90$; (b) attacks on CIFAR-10, benign client $\epsilon_b = 70$, malicious client $\epsilon_m = 150$.

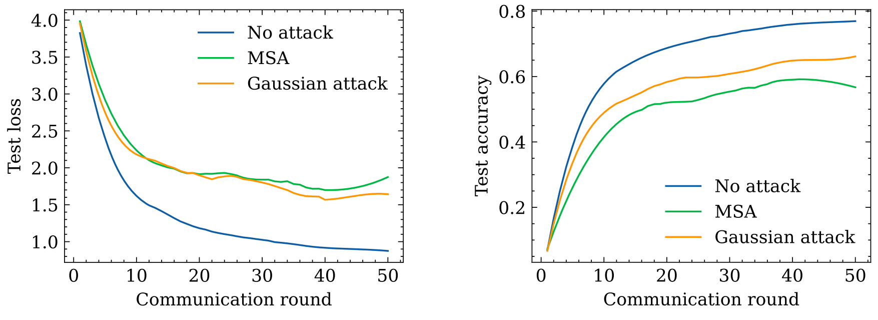


(a) (b) (c)

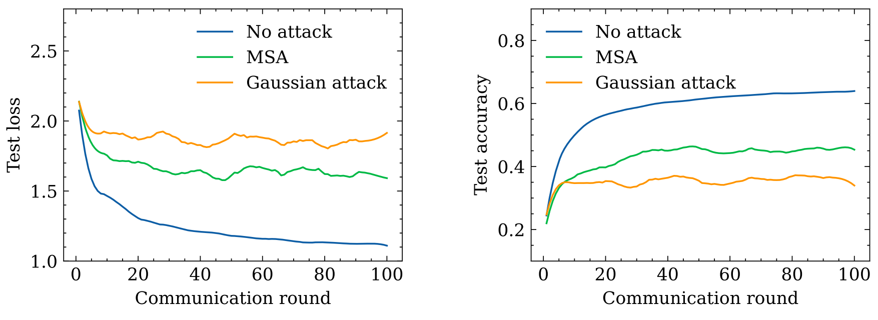


(d) (e) (f)

Fig. 10. Impact of the fraction of malicious workers under MSA and Gaussian attack. The aggregation rule is Krum. (a) and (b) are the global model test loss and accuracy on FeMnist after training; (c) is the average accuracy ranking of malicious on FeMnist; (d) and (e) are the global model test loss and accuracy on CIFAR-10 after training; (f) is the average accuracy ranking of malicious on CIFAR-10.

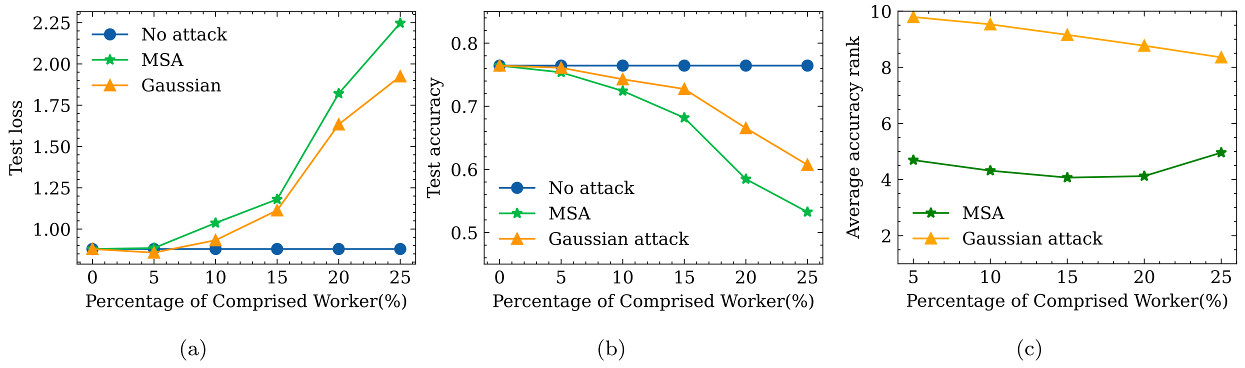


(a) Test loss and Test accuracy on FeMnist



(b) Test loss and Test accuracy on CIFAR-10

Fig. 11. Evaluation of MSA and Gaussian attack under Trimmed Mean, compromised rate is 20%. (a) attacks on FeMnist, benign client privacy budget $\epsilon_b = 50$, malicious client privacy budget $\epsilon_m = 100$; (b) attacks on CIFAR-10, benign client privacy budget $\epsilon_b = 40$, malicious client privacy budget $\epsilon_m = 100$.



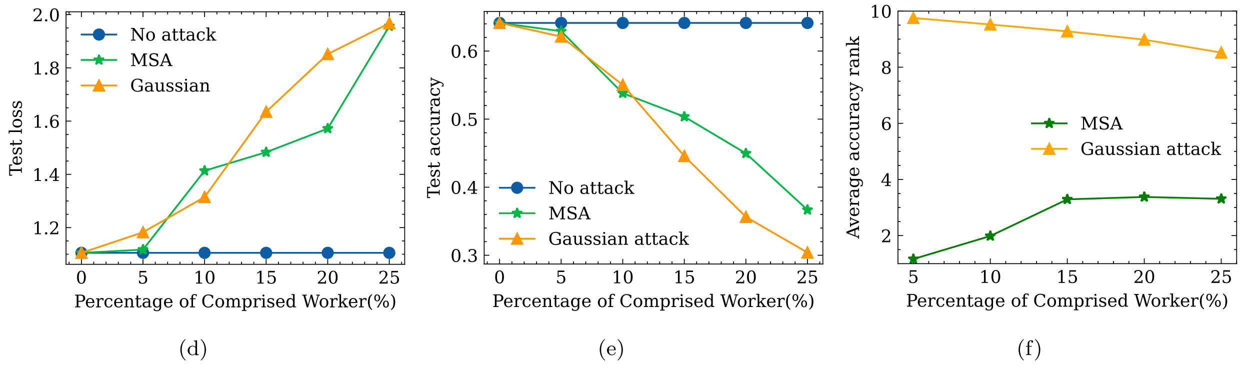


Fig. 12. Impact of the fraction of malicious workers under MSA and Gaussian attack. The aggregation rule is Trimmed Mean. (a) and (b) are the global model test loss and accuracy on FeMnist after training; (c) is the average accuracy ranking of malicious on FeMnist; (d) and (e) are the global model test loss and accuracy on CIFAR-10 after training; (f) is the average accuracy ranking of malicious on CIFAR-10.

work may focus on improving the stealth performance of MSA, like how to scale the model parameters while keeping the distribution of the model parameters unchanged.

CRediT authorship contribution statement

Ming Yang: Formal analysis, Methodology, Writing – original draft. **Hang Cheng:** Conceptualization, Resources, Writing – review & editing. **Fei Chen:** Project administration, Supervision. **Ximeng Liu:** Methodology, Validation, Writing – review & editing. **Meiqing Wang:** Data curation, Investigation. **Xibin Li:** Software.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grant 62172098, Grant 62072109, and Grant 61702105; in part by the Natural Science Foundation of Fujian Province under Grant 2020J01497; and in part by the Education Research Project for Young and Middle-Aged Teachers of the Education Research Project for Young and Middle-Aged Teachers of the Education Department of Fujian Province under Grant JAT200064 and Grant JAT190020.

References

[1] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: pre-training of deep bidirectional transformers for language understanding, preprint, arXiv:1810.04805.
[2] C. Sardianos, N. Tsirakis, I. Varlamis, A survey on the scalability of recommender systems for social networks, in: Social Networks Science: Design, Implementation, Security, and Challenges, Springer, 2018, pp. 89–110.
[3] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Commun. ACM. 60 (6) (2017) 84–90.
[4] B. McMahan, E. Moore, D. Ramage, S. Hampson, B.A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: Artificial Intelligence and Statistics, PMLR, 2017, pp. 1273–1282.
[5] Q. Yang, Y. Liu, T. Chen, Y. Tong, Federated machine learning: concept and applications, ACM Trans. Intell. Syst. Technol. 10 (2) (2019) 1–19.
[6] C. He, S. Li, J. So, X. Zeng, M. Zhang, H. Wang, X. Wang, P. Vepakomma, A. Singh, H. Qiu, et al., Fedml: a research library and benchmark for federated machine learning, preprint, arXiv:2007.13518.
[7] V. Tolpegin, S. Truex, M.E. Gursoy, L. Liu, Data poisoning attacks against federated learning systems, in: European Symposium on Research in Computer Security, Springer, 2020, pp. 480–501.
[8] M. Fang, X. Cao, J. Jia, N. Gong, Local model poisoning attacks to {Byzantine-Robust} federated learning, in: 29th USENIX Security Symposium (USENIX Security 20), 2020, pp. 1605–1622.
[9] J. Sun, A. Li, L. DiValentin, A. Hassanzadeh, Y. Chen, H. Li, Fl-wbc: enhancing robustness against model poisoning attacks in federated learning from a client perspective, Adv. Neural Inf. Process. Syst. 34 (2021) 12613–12624.
[10] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, V. Shmatikov, How to backdoor federated learning, in: International Conference on Artificial Intelligence and Statistics, PMLR, 2020, pp. 2938–2948.
[11] M. Nasr, R. Shokri, A. Houmansadr, Comprehensive privacy analysis of deep learning: passive and active white-box inference attacks against centralized and federated learning, in: 2019 IEEE Symposium on Security and Privacy (SP), IEEE, 2019, pp. 739–753.
[12] L. Melis, C. Song, E. De Cristofaro, V. Shmatikov, Exploiting unintended feature leakage in collaborative learning, in: 2019 IEEE Symposium on Security and Privacy (SP), IEEE, 2019, pp. 691–706.
[13] Y. Huang, S. Gupta, Z. Song, K. Li, S. Arora, Evaluating gradient inversion attacks and defenses in federated learning, Adv. Neural Inf. Process. Syst. 34 (2021) 7232–7241.
[14] L. Zhu, Z. Liu, S. Han, Deep leakage from gradients, Adv. Neural Inf. Process. Syst. 32 (2019) 14747–14756.
[15] B. Zhao, K.R. Mopuri, H. Bilen, idlg: improved deep leakage from gradients, preprint, arXiv:2001.02610.
[16] C. Zhao, S. Zhao, M. Zhao, Z. Chen, C.-Z. Gao, H. Li, Y.-a. Tan, Secure multi-party computation: theory, practice and applications, Inf. Sci. 476 (2019) 357–372.
[17] A. Acar, H. Aksu, A.S. Uluagac, M. Conti, A survey on homomorphic encryption schemes: theory and implementation, ACM Comput. Surv. 51 (4) (2018) 1–35.
[18] M. Abadi, A. Chu, I. Goodfellow, H.B. McMahan, I. Mironov, K. Talwar, L. Zhang, Deep learning with differential privacy, in: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 2016, pp. 308–318.
[19] R. Hu, Y. Gong, Y. Guo, Federated learning with sparsification-amplified privacy and adaptive optimization, preprint, arXiv:2008.01558.
[20] L. Sun, L. Lyu, Federated model distillation with noise-free differential privacy, preprint, arXiv:2009.05537.
[21] L. Sun, J. Qian, X. Chen, Ldp-fl: practical private aggregation in federated learning with local differential privacy, preprint, arXiv:2007.15789.
[22] R.C. Geyer, T. Klein, M. Nabi, Differentially private federated learning: a client level perspectiv, preprint, arXiv:1712.07557.
[23] P. Kairouz, Z. Liu, T. Steinke, The distributed discrete Gaussian mechanism for federated learning with secure aggregation, in: International Conference on Machine Learning, PMLR, 2021, pp. 5201–5212.
[24] K. Wei, J. Li, M. Ding, C. Ma, H. Su, B. Zhang, H.V. Poor, User-level privacy-preserving federated learning: Analysis and performance optimization, IEEE Trans. Mob. Comput. 21 (9) (2021) 3388–3401.
[25] V. Shejwalkar, A. Houmansadr, Manipulating the byzantine: optimizing model poisoning attacks and defenses for federated learning, in: NDSS, 2021.
[26] C. Xie, K. Huang, P.-Y. Chen, B. Li, Dba: distributed backdoor attacks against federated learning, in: International Conference on Learning Representations, 2019.
[27] H. Wang, K. Sreenivasan, S. Rajput, H. Vishwakarma, S. Agarwal, J.-y. Sohn, K. Lee, D. Papailiopoulos, Attack of the tails: yes, you really can backdoor federated learning, Adv. Neural Inf. Process. Syst. 33 (2020) 16070–16084.
[28] L. Lamport, R. Shostak, M. Pease, The byzantine generals problem, in: Concurrency: the Works of Leslie Lamport, 2019, pp. 203–226.

[29] P. Blanchard, E.M. El Mhamdi, R. Guerraoui, J. Stainer, Machine learning with adversaries: byzantine tolerant gradient descent, Adv. Neural Inf. Process. Syst. 30.
[30] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, B.Y. Zhao, Neural cleanse: identifying and mitigating backdoor attacks in neural networks, in: 2019 IEEE Symposium on Security and Privacy (SP), IEEE, 2019, pp. 707–723.
[31] L. Zhu, R. Ning, C. Wang, C. Xin, H. Wu, Gangsweep: sweep out neural backdoors by gan, in: Proceedings of the 28th ACM International Conference on Multimedia, 2020, pp. 3173–3181.
[32] C. Fung, C.J. Yoon, I. Beschastnikh, Mitigating sybils in federated learning poisoning, preprint, arXiv:1808.04866.
[33] H.B. McMahan, D. Ramage, K. Talwar, L. Zhang, Learning differentially private recurrent language models, preprint, arXiv:1710.06963.
[34] A. Cheng, P. Wang, X.S. Zhang, J. Cheng, Differentially private federated learning with local regularization and sparsification, preprint, arXiv:2203.03106.