2025 International Conference on Intelligent Computing
July 26-29, Ningbo, China
https://www.ic-icc.cn/2025/index.php

Efficient and Lightweight Federated Learning Scheme for Privacy Protection and Enhancement

Xuyan Zhang[1[0009−0004−2627−8147]], Zhencheng Fan[1[0009-0002-2467-1725]], Da Huang[1*[0009−0008−9665−9737]], Yuhua Tang[1[0000−0002−4956−3379]], Xiyao Liu[2[0000-0003-2718-659X]]

[1] College of Computer Science and Technology, National University of Defense Technology, Changsha 410073, China
[2] School of Computer Science and Engineering, Central South University, Changsha 410073, China
huangda1109@163.com

Abstract. Deep Neural Networks (DNNs) are widely used in computer vision, speech recognition, and recommender systems but raise privacy concerns due to data collection. Federated learning (FL) addresses this by enabling collaborative model training without sharing raw data. However, traditional FL is vulnerable to malicious attacks and inefficient for resource-constrained clients. This paper proposes an efficient and privacy-preserving lightweight federated learning (PL-FL) scheme combining Differential Privacy (DP) and ring-based Fully Homomorphic Encryption (FHE). The approach perturbs local model parameters with a Gaussian mechanism and uses FHE to prevent data theft. Formal analysis shows the scheme ensures model convergence with lower communication costs and robust privacy. Experiments confirm competitive performance and efficiency compared to baseline FL, while privacy tests demonstrate strong protection against data theft.

Keywords: Federated Learning, Privacy-preserving, Differential Privacy, Fully Homomorphic Encryption.

1 Introduction

The constantly developing mobile computing technology has generated huge amounts of data, and data-based artificial intelligence (AI) technology is widely used in fields such as mobile social networks, smart cities, and so on. The integration of the limited and uneven-quality data scattered at the edge of the network faces enormous obstacles, and data security and privacy protection have gradually become major global issues.

To address these challenges, Federated Learning (FL) [1] enables multiple clients to collaborate in the training of models, fulfilling the requirement of "data doesn't move, the model does". However, as shown in Figure 1, the actual implementation of this framework will inevitably face resource and privacy issues on both sides.

To prevent the privacy of data of clients from being threatened during parameter transmission and sharing, several technologies to further improve the privacy protection

of FL have been proposed. The algorithm proposed by Geyer et al [2] utilize differential privacy (DP) to hide user contributions and dataset related information and Agarwal et al [3] masked sensitive data for privacy protection by adding binomial noise. However, methods used DP fine tunes the gradient that causes precision loss and still suffers from data recovery attacks. Fang et al [4] proposed PFMLP based on partially homomorphic encryption (HE). The client only transmits encrypted gradients through HE. Park et al [5] used HE to construct security analysis protocols to protect privacy. Phong et al [6] designed a deep learning system based on additive HE, ensuring data privacy and maintaining model accuracy. Such methods can only avoid the privacy of data when transmitted to the cloud server, but cannot avoid collusion between clients and malicious clients to steal and recover data through means.

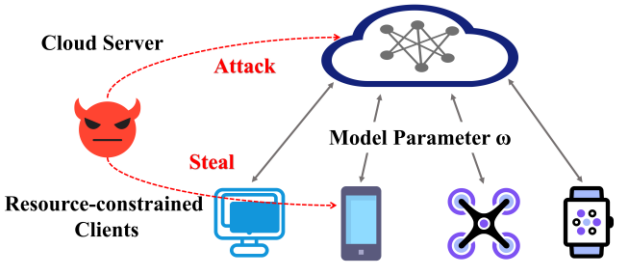

Fig. 1. Privacy and resource concerns in federated learning process.

This article proposes an innovative, secure, and lightweight federated learning privacy protection mechanism (PL-FL), the main contributions of this work (PL-FL) are summarized as follows:

(1) The innovative integration of dual privacy strategies: This paper is the first to combine differential privacy with homomorphic encryption to form a dual privacy protection mechanism. This fusion strategy significantly increases the strength of privacy protection under the federated learning framework.

(2) To prevent malicious collusion between clients and to effectively prevent the overall reduction in training efficiency caused by collusion, differential privacy technology based on the Gaussian mechanism is used to interfere with the local parameters of clients during local training. Differential privacy reduces the risk of data leakage by adding random noise to the data.

(3) In the parameter transfer phase after model training, a FHE CKKS supporting floating-point vectors is also used to establish a continuous line of defense. It ensures that malicious attackers cannot steal the model parameters transferred to the cloud server and stored by the client, effectively controlling the recovery and theft of the client's original data by malicious attackers.

(4) Numerous experiments have shown that our work can achieve competitive performance compared to various baseline FL methods. Privacy analysis proves that our method has better privacy protection against parameter attacks compared to the baseline FL method. In terms of communication consumption and time, model parameters of different sizes can also be trained and transmitted with minimal communication time consumption.

2 Related Work

The essence of federated learning is to use shared training model parameters instead of sharing raw data, which enhances user privacy protection, but the model may still leak sensitive user information from various aspects. Malicious attackers may obtain global parameters and control the upload of these parameters, or they may steal model parameters by maliciously modifying the input and output values of the model.

On this issue, researchers at home and abroad have conducted extensive and in-depth studies and proposed diverse types of attacks, such as membership inference attacks and attribute inference attacks. Carlini et al [13] discuss privacy leakage in the GPT-2 language model and find that although the training set of the model is tremendous, it still has memory for individual training samples. Song et al [14] conducted a study to identify and assess the privacy risks associated with user-level participation in FL. They proposed a multi-task generative adversarial network framework as a solution to these risks. It can identify the identity of the client, authenticity of the samples and the generator generates training samples against the attacking client.

The commonly used methods to protect federated privacy include differential privacy (DP), homomorphic encryption (HE) and secure multi-party computation. It was introduced in the previous section.

In a secure multi-party computation system, each participant holds its private data and cooperates to compute a public function, and the information of other participants is not available except for the result of the function computation. Lin et al [15] proposed FR-FMSS, a federated recommendation framework, which uses pseudo-tagging and secret sharing techniques to modify the shared data, and protects the list of items rated by the user along with the rated values for the items. Huang et al [16] proposed the hybrid federated learning framework, which combines a trusted execution environment with multi-party secure computing for secure key distribution, encryption, and decryption, and provides an authentication mechanism for each participant to ensure the security of the participant's data. However, the scalability of the multi-party secure computing system is poor, and the communication complexity is greatly increased in large-scale end-device scenarios.

We aim to design a hybrid encryption-based, privacy-preserving FL scheme that can resist poisoning attacks, reduce computational costs, and provide privacy protection. Meanwhile, our solution should achieve almost the same accuracy as FedAvg. Specifically, we are committed to achieving the goal of robustness. Our scheme should be able to resist malicious attacks, which means the accuracy of the global model should not be affected by the privacy of malicious clients. Our goal is to protect customer data from damage. Neither third-party nor malicious software can obtain or infer the original information of customers. Compared to traditional privacy-preserving FL schemes, our approach should reduce the computational overhead caused by encryption, while not sacrificing accuracy while protecting privacy and resisting malicious attacks.

3 Method

As illustrated in Figure 2, the main process of PL-FL is as follows: in each FL communication round, each client locally updates its model parameters and uses a differential privacy protection mechanism to add noise to the local model parameters. After perturbing parameters with Gaussian Mechanism, we utilize the FHE mechanism CKKS to ensure the successive defense of transmission between the cloud server and clients.

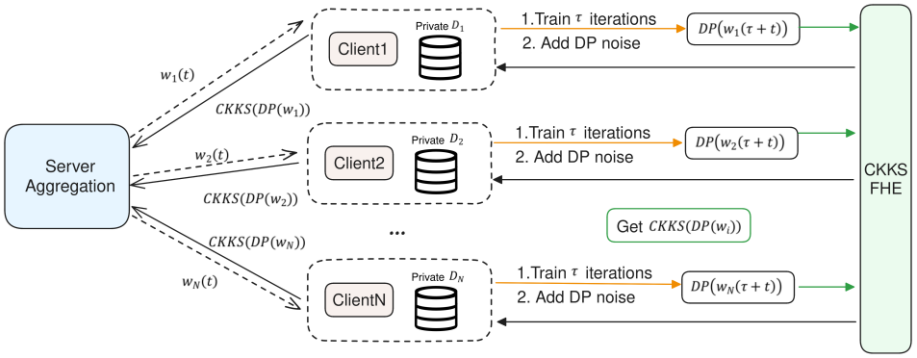

Fig. 2. The process of our work.

Table 1. Description of the main notations in PL-FL.

Notation	Description
C_i	Resource-limited client i
N	Number of clients
D_i	Training datasets of the client C_i
τ	Client training iteration rounds
t	Iteration of adjacent aggregations
K	Rounds of global aggregation
$W(t)$	Global model parameters in iteration t
ε, δ	Privacy budget, possibility of privacy broken
p, q, L	Base integer and modulus, number of levels for HE
R	The ring for HE

3.1 Problem Formulation

In this work, federated learning is set up with an honest cloud server and N clients holding their private datasets $D_1, D_2, ..., D_n$. Considering the setting of horizontal FL, these private data have the same feature space but different sample spaces. Table 1 lists the main notations. As shown in Figure 3, typical FL is generally a cyclic execution of the following steps:

- Initialize the global model on the cloud server if there is no model pre-saved. The global model parameters in iteration t are represented as $w(t)$ and sent to a subset of available clients.
- Client i receives global model parameters w(t)from the cloud server to initialize the local parameters $w_i(t)$. Then it trains the model with local datasets x_i by

stochastic gradient descent algorithm (SGD) for local iterations τ. The client i sends $w_i(\tau+t)$ to the cloud server after updating.

- The cloud server aggregates $w(\tau+t)$ from client i and utilizes FedAvg to update global model parameters $w_i(\tau+t)$. This process involves K rounds of global aggregation and update so that the total number of iterations is represented by $I = K\times \tau$.

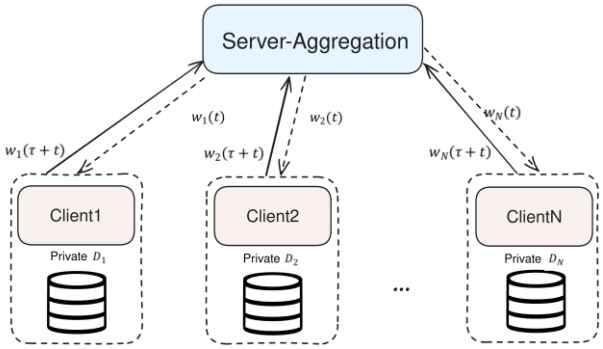

Fig. 3. Typical federate learning steps.

3.2 Client Update

Each client uses its local private dataset for the local training and update phase through a federated, collaborative, and distributed pattern. It trains neural network models and updates parameters through iterative execution of the stochastic gradient descent algorithm (SGD) while Algorithm 1 describes the detailed procedure. Specifically, client i holds the parameter $w_i(t)$ of the t^t round of global iteration and possesses its local dataset D_i. Given the loss function L, clients i utilizes SGD to update parameters $w_i(t)$ by private data D_i and obtains $w_i(\tau+t)$.

$$w_i(t+1) \leftarrow w_i(t) - \mu\nabla_w L(b, w_i(t)) \tag{1}$$

where μ is the learning rate, t is the t^t round of global iteration, ∇ represents the gradient, and b is the mini-batch of the private training dataset D_i. After performing τ rounds local update, the client gains the updated parameter $w_i(\tau+t)$, but it is not directly transmitted to the cloud server for aggregation. Instead, the parameters will be operated through a series of privacy enhancement techniques as follows.

3.3 DP-Privacy Preservation

In the differential privacy preservation stage, the updated parameter w_i from the client i is used as input for the DP noise addition function. After passing through a DP algorithm based on the Gaussian mechanism, we can obtain DP(w_i) that is represented as parameters disturbed by noise. The DP function will be explained in detail, which can improve the availability of data analysis to a certain extent and minimize the possibility of personal privacy leakage.

Algorithm 1 Client Update

Input: clients' dataset D_i, learning rate μ, local training epoch τ, batchsize B

1: Clients receive $w(t)$ from the cloud server.
2: **for** each client $i = 1, ..., N$ in parallel **do**
3: Splitting local dataset into multiple datasets B
4: **for** each batch $b \in B$ **do**
5: Execute gradient descent:
6: $wi\ (t+1) \leftarrow wi\ (t) - \mu \nabla w\ L(b, wi\ (t))$
7: **end for**
8: Execute Algorithm 2 to encrypt
9: **end for**

Output: Updated parameters

3.4 Gaussian Mechanism

To achieve (ε, δ)-DP, Gaussian mechanism builds a noised function M which adds the original query function f_{qr} and the random noise that follows the normalized distribution $N\left(0, s_f^2 \times \sigma^2\right)$. S_f is the sensitivity of f_{qr}, which means the maximum difference between query results on adjacent datasets. In our protocol, S_f can be set to 1. Then, the noise addition function DP is shown in the formula:

$$\mathrm{DP}\left(w_i^j\right) = w_i^j + N(0, \sigma^2) \tag{2}$$

where σ represents the intensity of noise and w_i^j represents the j^{th} layer model parameters of client i. Under this mechanism, $M \triangleq M_G(x, f(\cdot), \varepsilon) + (Y_1, Y_2, \ldots Y_k)$. The standard deviation δ of Gaussian distribution determines the scale of noise; ε represents privacy budget, which is negatively correlated with noise;δ represents the relaxation term, for example, if set to 10^{-5}, it means that only a probability of 10^{-5} can be tolerated to violate strict differential privacy. The partial proof of the Gaussian mechanism is as follows: S_1 represents the part that strictly adheres to DP, and S_2 represents the part that violates strict DP. In relaxed differential privacy, the output is divided into two parts, one strictly adhering to DP and the other violating DP. Therefore, it is necessary to separate the output set into two parts. The following formula proves that the first part is constrained by ε, while the second part is less than δ.

$$\begin{aligned} \Pr_{x\sim\mathcal{N}(0,\sigma^2)}[f(x)+x\in S] &= \Pr_{x\sim\mathcal{N}(0,\sigma^2)}[f(x)+x\in S_1] \\ &\quad +\Pr_{x\sim\mathcal{N}(0,\sigma^2)}[f(x)+x\in S_2] \\ &\le \Pr_{x\sim\mathcal{N}(0,\sigma^2)}[f(x)+x\in S_1]+\delta \\ &\le e^{\varepsilon}\Pr_{x\sim\mathcal{N}(0,\sigma^2)}[f(y)+x\in S_1]\Big)+\delta \end{aligned} \tag{3}$$

If σ is set to be $\sqrt{2\log\frac{5}{4\delta}}/\varepsilon$, then each process of adding noise satisfies (ε, δ)-DP so that we can obtain the perturbed parameters.

3.5 CKKS Secure Transmission

CKKS is an FHE scheme that supports the addition, subtraction, and multiplication of floating-point vectors in ciphertext space while maintaining homomorphism. It includes three basic operations: key generation, encryption, and decryption.

As shown in Figure 4, the steps of CKKS are as follows. Given safety parameters λ, and choose the power N of two integers. Respectively set distribution X_{key}, X_{error}, and X_{enc} for key, error learning, and encryption on $R = \mathbb{Z}[X]/(X^N + 1)$.

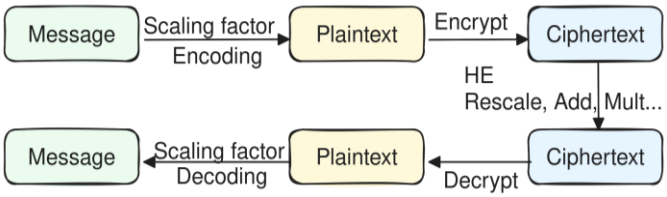

Fig. 4. The process of CKKS encryption algorithm.

For a base integer p and the number of levels L, as well as a base modulus q_0, set the modulus of the ciphertext $q_l = p^l \cdot q_0 (1 \le l \le L)$. For the ring R, let k=2 and then randomly generate the parameter to output: $(N, X_{\text{key}}, X_{\text{error}}, X_{\text{enc}}, L, q_l)$.

- Generate a key pair consisting of the secret key sk, the public key pk, and the evaluation key evk, with the public key used for encrypting data and the private key used for decrypting data.
- Encryption operation. The plaintext data is converted into complex form and encrypted into ciphertext using the public key. Randomly generate r$\leftarrow X_{enc}$ and e_0 output ciphertext c, where c ϵ R_{qL}^k .

$$. \qquad c = r \cdot pk + (m + e_0, e_1)(mod q_L) \tag{4}$$

- Decryption operation. Use the secret key to convert encrypted data into plaintext data. For ciphertext of the same level, decrypt to obtain the plaintext result.

$$m' = \langle c, sk \rangle (mod q_l) \tag{5}$$

Algorithm 2 Encrypted process

Input: parameters w_i , DP function, empirical values θ , intensity of noise σ, $CKKS$ function

1: **for** each network layer j in w_i **do**

2: $DP(w_i^j) = \theta \frac{w_i^j}{\|w_i^j\|} + N(0, \theta^2 \sigma^2)$

3: **end for**

4: Get $DP(w_i)$.

5: Encryption process: $CKKS$ (DP (w_i)). (The detailed process of CKKS is presented in the 3.5 section)

Output: Encrypted parameters $CKKS$ (DP (w_i)).

Due to the homomorphic properties of the CKKS encryption algorithm, the computation of ciphertext by the cloud server in the FL scheme is equivalent to computation on plaintext, ensuring clients' privacy and security to a certain extent. Specifically, client i first rescales the DP-processed parameters and then encodes them to obtain plaintext in the complex field. After the above CKKS-FHE operation, we package the above encryption process as the function CKKS, and the ciphertext CKKS (DP (w_i)) is obtained. The client i transmits the ciphertext to the cloud server for aggregation. The overall privacy protection process is shown in Algorithm 2.

3.6 Cloud Server Aggregation

After receiving encrypted parameters from multiple clients, the honest cloud server aggregates the encrypted model parameters globally using the FedAvg algorithm. Algorithm 3 describes the detail procedure of cloud server aggregation.

Algorithm 3 Cloud Server Aggregation

Input: size of clients' data X_i, global training iteration T

1: cloud server receives updated w_i $(\tau + t)$ from clients.

2: **for** $t \in T$ **do**

3: Do FedAvg aggregation:

4: $w(t+1) \leftarrow \sum_{i=1}^{N} \frac{X_i}{X} w_i(t+\tau)$

5: **end for**

Output: Global parameters $w\,(t+1)$

6: cloud server sends $w\,(t+1)$ to all clients

By utilizing the properties of CKKS-FHE, the result obtained by homomorphic addition and multiplication of parameters in the ciphertext space and decryption it to plaintext data is equivalent to the result obtained by performing the same calculation on the original plaintext data. Among them, homomorphic addition Add (c,c') to c_{add}, for ciphertext c, c', the ciphertext addition result is

$$c_{add} = c + c'(mod q_l) \tag{6}$$

Homomorphic multiplication $Mult_{evk}$ (c,c')$\rightarrow cmult$, for ciphertext c, c',

$$c = (c_0, c_1), c' = (c_0', c_1') \in R_{q_l}^2 \tag{7}$$

$$(d_0, d_1, d_2) = (c_0 c_0', c_0 c_1' + \bar{c}_0' c_1, c_1 c_1')(mod q_l) \tag{8}$$

and the result of multiplying its ciphertext is

$$c_{mult} = (d_0, d_1) + \lfloor P^{-1} \cdot d_2 \cdot sk \rceil (mod q_l) \tag{9}$$

Due to the homomorphic property of CKKS, when the cloud server receives parameters $CKKS\ (DP(w_i))$ uploaded by N clients, it performs a weighted average calculation as shown in the following formula:

$$w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_t^k \tag{10}$$

which is equivalent to the same computation on plaintext DP (w_i). To some extent, this process prevents client data from being stolen and restored through parameters. The clients use the private key to decrypt the received global encryption parameters based on the decryption function in the CKKS algorithm, and continue with local training updates. Ultimately, after repeating the FL scheme for a certain number of iterations, end the training.

3.7 Security Analysis

According to the description of the above methods, our PL-FL mainly focuses on DP and CKKS-FHE methods to protect user data privacy and security, which can effectively prevent the leakage of local parameters on the client side and attacks during transmission to a certain extent. In each local training iteration, the clients interfere with the intermediate features of each data batch with noise intensity ε. The maximum difference in the intermediate features between the two data samples remains within the sensitivity Sf in DP. It also strictly follows the Gaussian mechanism. Due to each batch being independent and satisfying the same DP, based on the parallel combination theorem in [7], the entire dataset is protected by the same DP protection strength.

In the entire FL scheme, the client stores the secret key, while the public is shared between the client and the cloud server. The parameters transmitted between the cloud server and the client are operated by the CKKS encryption algorithm, and their security can be attributed to the CKKS. It is impossible to recover the key and original data from the intermediate encrypted results for a third-party attacker.

Assuming a malicious client colludes with other honest clients to obtain their private key and DP encrypted parameters. In the description of the above method, when σ is set to be $\sqrt{2\log\frac{5}{4\delta}}/\varepsilon$, Gaussian mechanism satisfies $(\varepsilon, \delta) - DP$. The Gaussian mechanism in DP can effectively prevent parameter leakage and provide security guarantees for the data privacy of honest clients.

3.8 Communication Cost Analysis

In terms of communication consumption, let the weight matrix of a neural network be p×q, with a bias term of 1×p. E(x) is a function that maps the size x of the list to the corresponding ——expansion factor, size(x) is a function that calculates the storage space of a list of size x, O is the other communication cost. The maximum expansion factor of communication cost, G_{biggest}, is shown in the following formula.

$$\begin{aligned} G_{\text{biggest}} &= \frac{\text{size}(p(q+1))\left(\frac{pq}{p(q+1)}E(q)+\frac{p}{p(q+1)}E(p)\right)+O}{\text{size}(p(q+1))+O} \\ &\leq \frac{pq}{p(q+1)}E(q)+\frac{p}{p(q+1)}E(p) \end{aligned} \tag{11}$$

We compared the time and number of homomorphic operations on BFV [8], TFHE [9], and the CKKS homomorphic encryption method used in this paper, as shown in

Table 2. It can be seen that in terms of time, the ciphertext generated by the PL-FL method in this article is more than twice as small as the ciphertext generated by BFV, which can be much faster than BFV operation. Although using TFHE, due to the small parameters used in the password system, the preparation and encryption steps are swift. However, compared to the method in this article, it consumes more time in the process of weight encryption and network inference.

Table 2. Number of homomorphic operations and Time Usage

	BFV	CKKS	TFHE		BFV	CKKS	TFHE
Additions	21	26	-	KeyGen	4.5s	1.96s	0.16s
Multiplications	10	16	-	Weights encryption	0.24 s	0.35s	16 s
Bootstrappings	-	-	25000000	Network evaluation	2.16 s	0.5s	$>$ 3days

Although there is an increase in communication consumption after CKKS encryption of parameters, subsequent experiments have shown that their computational efficiency is still relatively high.

4 Evaluation

4.1 Experiments Settings

Dataset and Model Settings. We use CIFAR-10, the original MNIST dataset and non-independent identically distributed cases of MNIST in our experiments and evaluation. We use LeNet5, a simple CNN, and 2NN as the backbone network for image classification tasks in the FL scheme. LeNet5 has one input layer, two convolutional layers, two pooling layers, and three fully connected layers, and the Sigmoid activation function. The 2NN only has two fully connected layers and CNN has the same structure as LeNet5 but different internal parameters.

Baselines. We compare our PL-FL with the following baselines, including FedAvg, FedProx [10], FLGDP [11], and individual local training of the model. Specifically, FedAvg is the most common FL algorithm that aggregates model parameters through weighted averaging. FedProx introduces proximal terms and constraints, which is the generalization and reconstruction of FedAvg. FLGDP provides members with inference resistance by implementing global DP on the aggregation model of universal FL.

Configurations. For the MNIST dataset, we set the number of clients N=5. During model training, we perform 20 global communication rounds and 3 local epochs, and each local epoch options a mini-batch of 4, with a learning rate of η set to 0.01. For the CIFAR-10 dataset, we set N=10 and perform 20 global communication rounds and 15 local epochs, and each local epoch options a batch size of 100, with η set to 0.01. The

degree of the cyclotomic polynomial in the CKKS is set to 4096, and the prime factorization of coefficient modulus is set to [22, 20, 20, 22].

4.2 Experiment Results

We first evaluate the performance of our work by comparing the average accuracy of clients based on the aforementioned dataset, model, and baseline.

Performance Evaluation. We first evaluate the performance of our work by comparing the average accuracy of clients based on the above dataset, model, and baseline. Table 3 shows the comparison results. It demonstrates that our method can still achieve results similar to FedAvg and FedProx even with added noise.

Table 3. Comparison of our work with baselines in terms of top-1 test accuracy.

Dataset	MNIST		CIFAR-10		MNIST-non-IID
Method	2NN	LeNet5	CNN	LeNet5	LeNet5
Local	97.17	97.94	54.84	54.34	68.94
FedAvg	98.28	99.07	60.10	58.44	74.16
FedProx	99.03	99.13	60.36	58.92	74.53
FLGDP	98.15	99.10	55.21	55.42	72.13
Ours	98.88	99.06	58.82	57.44	73.19

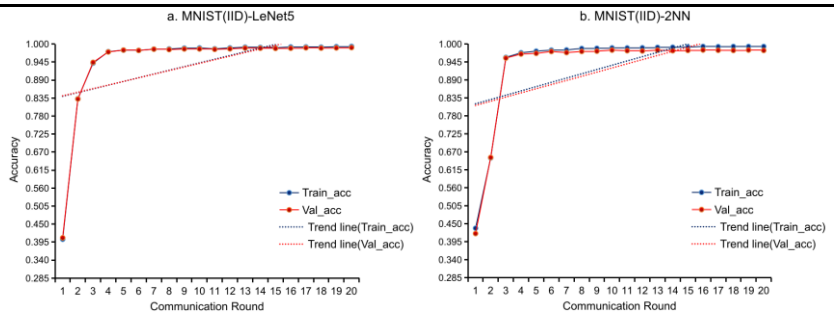

Fig. 5. Figures a-b respectively shows the accuracy trends of independent and identically distributed MNIST

Compared with the FLGDP method, our method, with LeNet5, can achieve an accuracy of 99.06% on the MNIST dataset and 57.44% on the CIFAR-10, with almost little performance loss. For non-IID datasets, the results are almost similar to those of local training, achieving an accuracy of 73.19%.

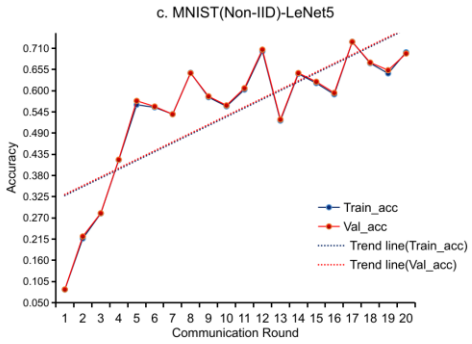

Fig. 6. The accuracy trends of non-independent and identically distributed MNIST.

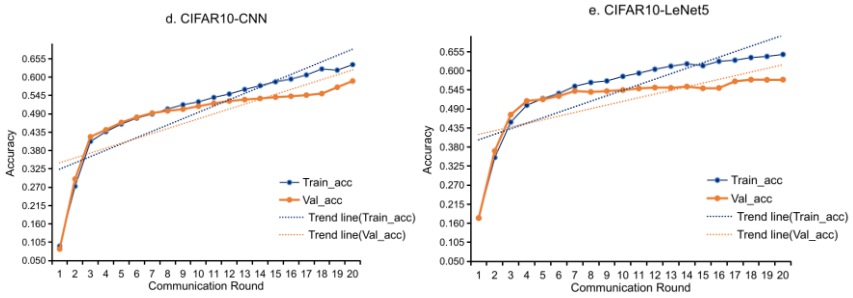

Fig. 7. Figures d-e respectively shows the accuracy trends of CIFAR-10

Figure 5 and 7 shows that the proposed method achieved good accuracy in five global iteration rounds on both the MNIST and CIFAR-10 datasets during 20 transmission processes. The MNIST dataset can achieve a training accuracy of 99.02%, meanwhile, CIFAR-10 can achieve 65.78%. For non-IID datasets in Figure 6, although the accuracy fluctuation is unstable, the overall improvement continues with iteration rounds. It can be seen that our work provides privacy protection, while also ensuring the performance of the model.

Table 4 simulates the results of only a portion of clients participating in federated training after being attacked or taken offline. It can be seen that even if some clients do not participate in the model aggregation process, it will not affect the model training and accuracy of other clients.

In addition, we encrypted parameters of different sizes, ranging from 10 bytes to 8000 bytes, using PL-FL and compared and analyzed the storage space before and after encryption. By displaying the encrypted multiples in the Figure 8, we can obtain an approximate expansion factor of communication cost, which is consistent with the results obtained from the Equation 16, in the above security analysis. Moreover, it is very friendly to clients with limited resources in terms of communication memory and communication time costs, and will not bring too much pressure.

Table 4. The training results of the model on the remaining clients on the MNIST and CIFAR-10 datasets after a certain number of clients are taken offline.

Model	Method	Clients Number	Client-Offline rate			
			0%	15%	25%	35%
MNIST	2NN	10	97.94	97.91	97.89	97.73
	LeNet5	10	99.06	98.78	98.77	98.91
CIFAR10	CNN	20	54.33	54.22	54.81	53.64
	LeNet5	20	56.40	56.33	55.59	56.06

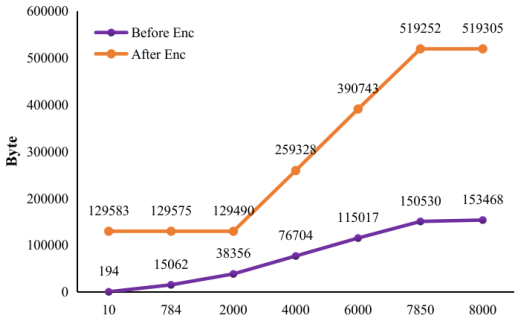

Fig. 8. Comparative analysis of storage space.

Paillier homomorphic encryption methods are also commonly used in traditional federated leading to protect privacy. We also compared the time required for baseline including local training, FedAvg, and Paillier-FL [12]. As shown in Table 5, local training and FedAvg have little time consumption and can complete an iteration in a very short time and quickly reach convergence. After setting the same key size, comparing Paillier-FL with our work, our scheme consumes approximately one-fourth of Paillier-FL. The comparison of model parameter quantities in Table 5 is shown in the Figure 9. It indirectly demonstrates the efficiency of our method with large parameter quantities.

Table 5. Comparison of computation time consumption between our work and other baseline methods in each iteration.

Method	MNIST		CIFAR-10	
	LeNet5	2NN	LeNet5	CNN
FedAvg	6.60 s/ it	4.12 s/ it	1.17 s/ it	1.15 s/ it
FLGDP	6.67 s/ it	4.29 s/ it	1.21 s/it	1.17 s/ it
Paillier-FL	68.5 s/ it	83.14 s/ it	109.48 s/it	106.39 s/ it
Ours	17.58 s/ it	18.09 s/ it	25.25 s/ it	20.62 s/ it

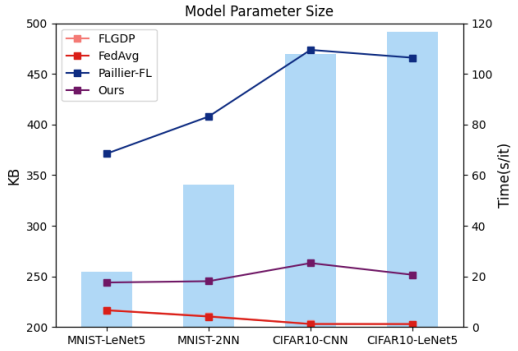

Fig. 9. The model size of LeNet5, 2NN and CNN on MNIST and CIFAR-10.

Privacy Evaluation. In terms of privacy protection evaluation, we manipulate the Peak Signal to Noise Ratio (PSNR) to quantify the similarity between the original data image and the image restored by DLG. PSNR is one of the indicators used to measure image quality. Based on the definition of mean square error (MSE), for a given original image I of size $m \times n$ and a noisy image K with added noise, PSNR can be defined as:

$$PSNR = 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right) = 20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right) \tag{12}$$

We use three parameters of LeNet5 to calculate the DLG restored images and PSNR values, including the original parameters for local training, and the encrypted parameters of FLGDP and our method.

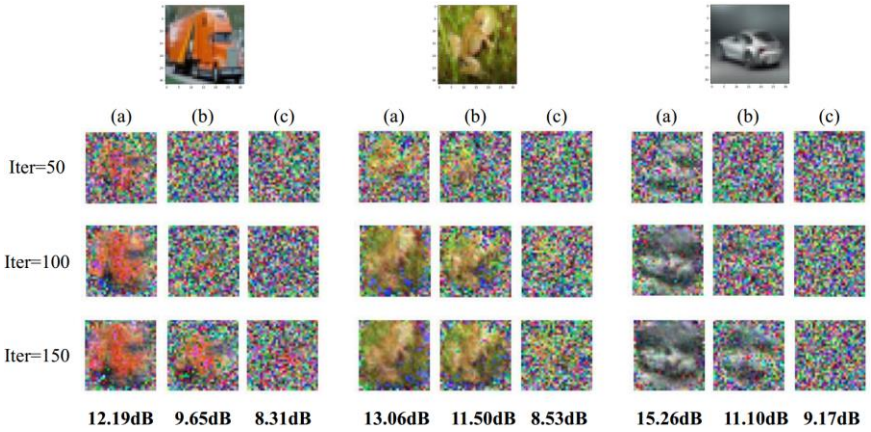

Fig. 10. Comparison of DLG recovery using (a): the original parameters, (b): FLGDP, (c): our work processed parameters, as well as the calculated PSNR values.

Figure 10 shows a comparison of random images and DLG restoration using three parameters. It can be seen that our work can prevent DLG from restoring the original data, protect data privacy up to a point, and have certain advantages compared to other methods. However, a significant restoration effect can be achieved through original

parameters. For FLGDP, PSNR reduces an average of 19.53% compared to the original parameters. From Figure 10, it can be seen that our work can effectively protect data privacy, achieving an average PSNR that is 35.80% lower than the original parameters. This also proves the effectiveness of the method proposed in this article for privacy protection.

5 Conclusion

The privacy-preserving lightweight federated learning (PL-FL) mechanism integrating differential privacy and homomorphic encryption proposed in this paper provides a new solution for data privacy protection in distributed machine learning. Its unique dual privacy protection strategy, lightweight design, comprehensive security and privacy analyses, and flexible adaptability demonstrate significant innovation and application value. To prevent collusion between clients, a DP technique based on the Gaussian mechanism is used to perturb the parameters of local model training of clients. In the parameter transmission phase after model training, a fully homomorphic CKKS encryption scheme supporting floating-point vectors is used to further strengthen the defense, ensuring that the model parameters transmitted to the cloud and stored in the client are not subject to malicious attacks, and effectively curbing the recovery and theft of the original data. Many experiments prove that the scheme has high computational efficiency and accuracy. Under the premise of ensuring privacy and security, future work will further explore the potential application of this mechanism in more complex and practical scenarios focusing on the combination of migration learning and joint learning and continuously optimizing its performance and efficiency.

Acknowledgments. This study was funded by Institute for Quantum Information & State Key Laboratory of High Performance Computing Fund: 202401-13.

References

1. H.Brendan McMahan, EiderB Moore, Daniel Ramage, Seth Hampson, and BlaiseAgüeray Arcas. 2016. Communication-Efficient Learning of Deep Networks from Decentralized Data. arXiv: Learning, arXiv: Learning (Feb 2016).
2. R. Geyer, TassiloJ. Klein, and Moin Nabi. 2017. Differentially Private Federated Learning: A Client Level Perspective. Cornell University - arXiv, Cornell University - arXiv (Dec 2017).
3. Naman Agarwal, Ananda Theertha Suresh, FelixX. Yu, Sanjiv Kumar, and H.Brendan McMahan. 2018. cpSGD: Communication-efficient and differentially private distributed SGD. arXiv: Machine Learning, arXiv: Machine Learning (May 2018).
4. Haokun Fang and Quan Qian. 2021. Privacy Preserving Machine Learning with Homomorphic Encryption and Federated Learning. Future Internet (Apr 2021),94. https://doi.org/10.3390/fi13040094
5. Guanjin Qu, Naichuan Cui, Huaming Wu, Ruidong Li, and Yuemin Ding. 2022.ChainFL: A Simulation Platform for Joint Federated Learning and Blockchain in Edge/Cloud

Computing Environments. IEEE Transactions on Industrial Informatics (May 2022), 3572–3581. https://doi.org/10.1109/tii.2021.3117481

6. Jaehyoung Park and Hyuk Lim. 2022. Privacy-Preserving Federated Learning Using Homomorphic Encryption. Applied Sciences (Jan 2022), 734. https://doi.org/10.3390/app12020734
7. Cynthia Dwork and Aaron Roth. 2013. The Algorithmic Foundations of Differential Privacy. Foundations and Trends® in Theoretical Computer Science (Jan 2013),211–407. https://doi.org/10.1561/0400000042
8. Junfeng Fan and Frederik Vercauteren. 2012. Somewhat Practical Fully Homo-morphic Encryption. IACR Cryptology ePrint Archive, IACR Cryptology ePrint Archive (Jan 2012).
9. Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. 2016. Faster Fully Homomorphic Encryption: Bootstrapping in Less Than 0.1 Seconds.3–33.https://doi.org/10.1007/978-3-662-53887-6_1
10. Paul C. Wallbott, Sascha Grollmisch, and Thomas Köllmer. 2023. Examining Speaker and Keyword Uniqueness: Partitioning Keyword Spotting Datasets for Federated Learning with the Largest Differencing Method. 167–177. https://doi.org/10.1007/978-3-031-39144-6_11
11. Houda Hafi, Bouziane Brik, PantelisA. Frangoudis, and Adlen Ksentini. 2023.Split Federated Learning for 6G Enabled-Networks: Requirements, Challenges and Future Directions. (Sep 2023).
12. Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. 2019.LOGAN: Membership Inference Attacks Against Generative Models. Proceedings on Privacy Enhancing Technologies (Jan 2019), 133–152. https://doi.org/10.2478/popets-2019-0008
13. Nicholas Carlini, Florian Tramèr, Eric Wallace, Jagielski Matthew, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, T.B. Brown, Dawn Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. 2020. Extracting Training Data from Large Language Models. USENIX Security Symposium, USENIX Security Symposium (Jan 2020).
14. Li Tian, AnitKumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2018. Federated Optimization in Heterogeneous Networks. arXiv: Learning, arXiv: Learning (Dec 2018).
15. Zhaohao Lin, Weike Pan, and Zhong Ming. 2021. FR-FMSS: Federated Recomdtion via Fake Marks and Secret Sharing. In Fifteenth ACM Conference on Recommender Systems. https://doi.org/10.1145/3460231.3478855
16. Anbu Huang, Yang Liu, Tianjian Chen, Yongkai Zhou, Quan Sun, Hongfeng Chai,and Qiang Yang. 2021. StarFL: Hybrid Federated Learning Architecture for Smart Urban Computing. ACM Transactions on Intelligent Systems and Technology 12, 4 (Aug 2021), 1–23. https://doi.org/10.1145/3467956