



# FLCP: federated learning framework with communication-efficient and privacy-preserving

Wei Yang<sup>1</sup> · Yuan Yang<sup>1</sup> · Yingjie Xi<sup>2</sup> · Hailong Zhang<sup>2</sup> · Wei Xiang<sup>3,4</sup>

Accepted: 10 May 2024 / Published online: 27 May 2024  
© The Author(s) 2024

## Abstract

Within the federated learning (FL) framework, the client collaboratively trains the model in coordination with a central server, while the training data can be kept locally on the client. Thus, the FL framework mitigates the privacy disclosure and costs related to conventional centralized machine learning. Nevertheless, current surveys indicate that FL still has problems in terms of communication efficiency and privacy risks. In this paper, to solve these problems, we develop an FL framework with communication-efficient and privacy-preserving (FLCP). To realize the FLCP, we design a novel compression algorithm with efficient communication, namely, adaptive weight compression FedAvg (AWC-FedAvg). On the basis of the non-independent and identically distributed (non-IID) and unbalanced data distribution in FL, a specific compression rate is provided for each client, and homomorphic encryption (HE) and differential privacy (DP) are integrated to provide demonstrable privacy protection and maintain the desirability of the model. Therefore, our proposed FLCP smoothly balances communication efficiency and privacy risks, and we prove its security against “honest-but-curious” servers and extreme collusion under the defined threat model. We evaluate the scheme by comparing it with state-of-the-art results on the MNIST and CIFAR-10 datasets. The results show that the FLCP performs better in terms of training efficiency and model accuracy than the baseline method.

**Keywords** Artificial intelligence · Federated learning · Privacy protection · Communication efficiency

## 1 Introduction

Artificial intelligence has achieved excellent breakthroughs in the fields of intelligent control, target detection, driverless vehicles, and fault diagnosis in recent years [1–4]. The rapid development of deep learning technology benefits mainly from rich training data, innovative algorithms, and the improved performance of computing equipment. Among

these factors, the richness of training data can directly affect the performance of deep learning models. However, in some industries, like medical and financial institutions, data barriers can be formed because data privacy makes it difficult for multiple research institutions to conduct collaborative learning through data sharing. Federated learning (FL) is a new privacy-protected distributed framework that allows a cluster of devices (local clients) to collaboratively learn a global shared model coordinated by a central server without exposing their private data [5–7]. Therefore, the FL framework reduces a substantial amount of the privacy risk and cost associated with conventional centralized training. Recently, this framework has attracted great attention in a variety of industries.

With the extension of the existing research, FL still faces various challenges, among which communication cost is a critical issue [8, 9]. In typical FL, only model parameters, rather than raw data, are transmitted between the client and the cloud server. A modern neural network architecture may have millions of model parameters, resulting in each iterative use requiring a high bandwidth [10]. In addition, FL requires

✉ Yuan Yang  
yangyuan@xaut.edu.cn

✉ Wei Xiang  
w.xiang@latrobe.edu.au

<sup>1</sup> School of Automation and Information Engineering, Xi'an University of Technology, Xi'an 710048, China

<sup>2</sup> School of International Engineering, Xi'an University of Technology, Xi'an 710048, China

<sup>3</sup> School of Computing Engineering and Mathematical Sciences, La Trobe University, Melbourne 3086, Australia

<sup>4</sup> College of Science and Engineering, James Cook University, Cairns, QLD 4870, Australia

multiple iterations to achieve greater model accuracy, resulting in a higher amount of communication overhead in the whole training process [11]. Finally, the resource-constrained edge devices participating in the training lead to limited bandwidth between clients and the cloud server, particularly during uplink transmission. The communication cost becomes a bottleneck for FL with limited bandwidth. Therefore, improving the efficiency of FL communication is a crucial factor in promoting its development [12].

While FL no longer requires collecting raw data from clients, the privacy implied by the data is not guaranteed [13]. To train the FL model, the client must upload the model parameters or gradients, which are essentially mappings to the raw data under specific rules and contain almost all the data information. Many attack models have demonstrated that data information can be derived from the model parameters or gradients. Such attacks are either disguised as models for training participants or come directly from the server, and the attack models are divided into refactoring attacks [14], inference attacks [15], and eavesdropping attacks [16]. Hence, safeguarding each client in FL from such advanced privacy attacks without a completely trusted server presents a significant challenge that requires immediate attention [8].

A wide range of studies have been conducted on FL communication efficiency and privacy protection. Among them, the methods for improving communication efficiency include mainly the quantization method [17], compression method [11], and communication delay method [18]. The privacy protection methods include mainly differential privacy (DP) [19, 20], secure multiparty computation (SMC) [21] and homomorphic encryption (HE) [22, 23]. However, few studies consider both communication efficiency and privacy protection in FL. In this paper, we propose a novel FL framework with communication-efficient and privacy-preserving (FLCP) that reduces the required level of communication overhead and provides a formal guarantee of privacy without assuming a fully trusted server. A novel adaptive weight compression algorithm is proposed to improve communication efficiency. The typical gradient compression algorithm may affect the training model's convergence rate and accuracy. In addition, since the data distribution of FL is non-IID and unbalanced, the traditional compression algorithm is unsuitable for the FL setting. Thus, to take full advantage of the performance of the compression algorithm, it is necessary to set a reasonable compression rate [24]. Based on the fixed compression rate algorithm, the compression rate is adaptively adjusted by our method in the dynamic perception model's training stage, so that each client enjoys a unique compression rate. We also introduce a communication delay method, by which communication efficiency is improved by increasing the degree of local computation to reduce the degree of communication frequency [18, 25]. This is the first study to describe a compression scheme applied to FL with

consideration of actual FL characteristics. To protect clients' privacy without a completely trusted server, we design a lightweight encryption protocol that incorporates local DP to provide provable privacy protection while maintaining the performance of the training model.

The main contributions of this paper are summarized as follows:

- 1) We propose a novel FL framework called the FLCP for training on distributed data without a trusted server, with communication efficiency and privacy protection. This method possesses robustness and can tolerate arbitrary clients dropping out during the training process with a negligible loss of accuracy.
- 2) We develop a novel adaptive weight compression FedAvg algorithm (AWC-FedAvg) aimed at reducing overall communication costs. Specifically, the AWC-FedAvg method adaptively adjusts the compression rate based on the non-IID and unbalanced data distributions in FL.
- 3) We develop a hybrid protocol that combines lightweight HE and DP to ensure full protection for the model training process and its results. Specifically, our enhanced lightweight HE protocol, with reduced ciphertext operation volume, facilitates the widespread implementation of FL. Additionally, the DP system, based on the Laplace mechanism, preserves the confidentiality of local updates, effectively safeguarding clients' privacy against adversarial collisions.
- 4) We conduct an analysis of the FLCP in terms of convergence analysis and privacy guarantees at the theoretical level. Moreover, we evaluate its performance extensively. The results show that compared with the baseline method, the FLCP has superior convergence while maintaining model accuracy, significantly improving training and communication efficiency.

The remainder of this paper is as follows. In Section 2, related works are discussed, and a comprehensive comparison to the existing approaches is conducted. Next, the research background and preliminaries are presented in Section 3. Then, we introduce the FLCP in Section 4. Afterward, a theoretical analysis of the FLCP is conducted in terms of its convergence analysis and privacy guarantees in Section 5. In Section 6, experimental evaluations are performed. Finally, Section 7 concludes the paper.

## 2 Related work

### 2.1 Communication efficiency in FL

At present, the training framework for FL is based mainly on the parameter server. Each time a compute node completes an

iteration, the generated gradients or model parameters need to be synchronized to the parameter server, which brings about a high level of communication overhead when synchronizing either gradients or model parameters. The solutions for improving the communication efficiency of distributed machine learning include primarily communication delay, compression, and quantization methods. The communication delay method attempts to reduce communication frequency  $f$  to improve communication efficiency. Federated averaging (FedAvg) was proposed by McMahan et al. [5] to decrease communication bit-width. In the FedAvg method, instead of communicating with the server after each iteration, the client performs several iterations of stochastic gradient descent (SGD) to compute the updates. The gradient compression algorithm plays a vital role in reducing the communication overhead of FL. Strom et al. [26] first proposed that clients upload only the gradient with a larger absolute value than the fixed threshold in the training process. This method provides a compression rate of as many as 3 orders of magnitude when performing an acoustic modeling task. However, for the threshold, it is not easy to select a suitable value in practice because different model architectures and layers may vary greatly. Later, P. Luo et al. [27] proposed an adaptive gradient compression algorithm. Unlike Strom's scheme, the adaptive gradient compression algorithm no longer relies on a fixed threshold to select gradients. Instead, the gradients are sorted according to their absolute values, and then an appropriate proportion of the gradients is adaptively selected for uploading in descending order of magnitude. Furthermore, Nori et al. [28] proposed an enhanced FL scheme that jointly and dynamically adjusts the local update frequency and compression rate to minimize the learning error. Y. Mao et al. [29] proposed an efficient communication FL framework with an adaptive quantization gradient (AQG), which adaptively adjusts the quantization level according to the update of local gradients to reduce any unnecessary transmission.

## 2.2 Privacy protection in FL

In response to the leakage of private data in FL, some privacy protection methods have been proposed, including mainly encryption-based privacy protection methods represented by secure multiparty computing (SMC) and homomorphic encryption (HE) and disturbance-based privacy represented by differential privacy (DP). SMC involves two or more participants holding private data and obtaining the output through joint computation, which meets the security characteristics of correctness, privacy, and fairness. To ensure communication between the device and the cloud server, Keith et al. [32] introduced an SMC protocol based on secret

sharing, applicable within the parameter aggregation process of FL. HE schemes can guarantee that specific mathematical operations performed on the ciphertext have the same effect on the plaintext. Moreover, Ma et al. [33] proposed a multi-key HE protocol for secure aggregation in FL, ensuring the privacy of shared gradients within an incompletely trusted environment. This method effectively addresses the privacy-security concern without compromising model accuracy. However, since data encryption requires many computing resources, this method is unsuitable for the training of deep learning models in large-scale data environments. Aono et al. [22] proposed encrypting the gradients generated by each client in FL to ensure the local data privacy security of multiple participants in the training. Since the computation amount of gradient encryption is directly related to the amount of training data, this method increases the model training time and degree of computational consumption and does not take the bias term into account. DP technology refers to introducing randomness into the model training process, that is, adding a certain degree of random noise to make the output and actual results have a certain degree of deviation to prevent the attacker from reasoning. Stacey et al. [30] proposed a privacy protection scheme combining DP and SMC for the FL model, protecting data privacy while achieving a high degree of accuracy.

## 2.3 Function comparison

The functional advantages of the FLCF are analyzed compared to those of the state-of-the-art FL approaches with communication efficiency and privacy preservation, i.e., FL with differential privacy (NBAFL) [19], hybrid approach to privacy protection FL (TP-SMC) [30], robust and communication efficiency FL (RCEFL) [11] and privacy-preserving federated deep learning (PPFDL) [31], as listed in Table 1. Specifically, NBAFL utilizes DP to ensure the confidentiality of each client's local gradient, but this approach does not guarantee model accuracy, and the default aggregation result is a publicly available parameter. TP-SMC proposes a hybrid security protocol for FL, which combines DP and SMC to provide high-level privacy protection and reduce the amount of injected noise. RCEFL proposes a sparse ternary compression framework that is robust to dropping out and that can improve communication efficiency and support FL environments but does not provide privacy protection. PPFDL additively incorporates HE and Yao's garbled circuits, ensuring the high level of security of the information related to the client. However, PPFDL neither guarantees the confidentiality of the model parameters in a collusion attack nor provides excellent communication efficiency during the training

**Table 1** Comparison of functionality with existing models

Function \ Model	Communication Efficient	Non-IID Support	Protect of Client's gradients	Protection of aggregated result	Robust to clients dropping out	Collusion Resistance
NbAFL [19]	×	×	✓	×	×	×
TP-SMC [30]	×	✓	✓	×	×	×
RCEFL [11]	✓	✓	×	×	✓	×
PPFDL [31]	×	✓	✓	✓	✓	×
FLCP	✓	✓	✓	✓	✓	✓

process. Table 1 illustrates that none of the abovementioned state-of-the-art methods can completely solve the challenges of communication efficiency and privacy protection. Contrastingly, our proposed FLCP mitigates the abovementioned attacks by integrating HE and DP and improves communication efficiency through a compression algorithm.

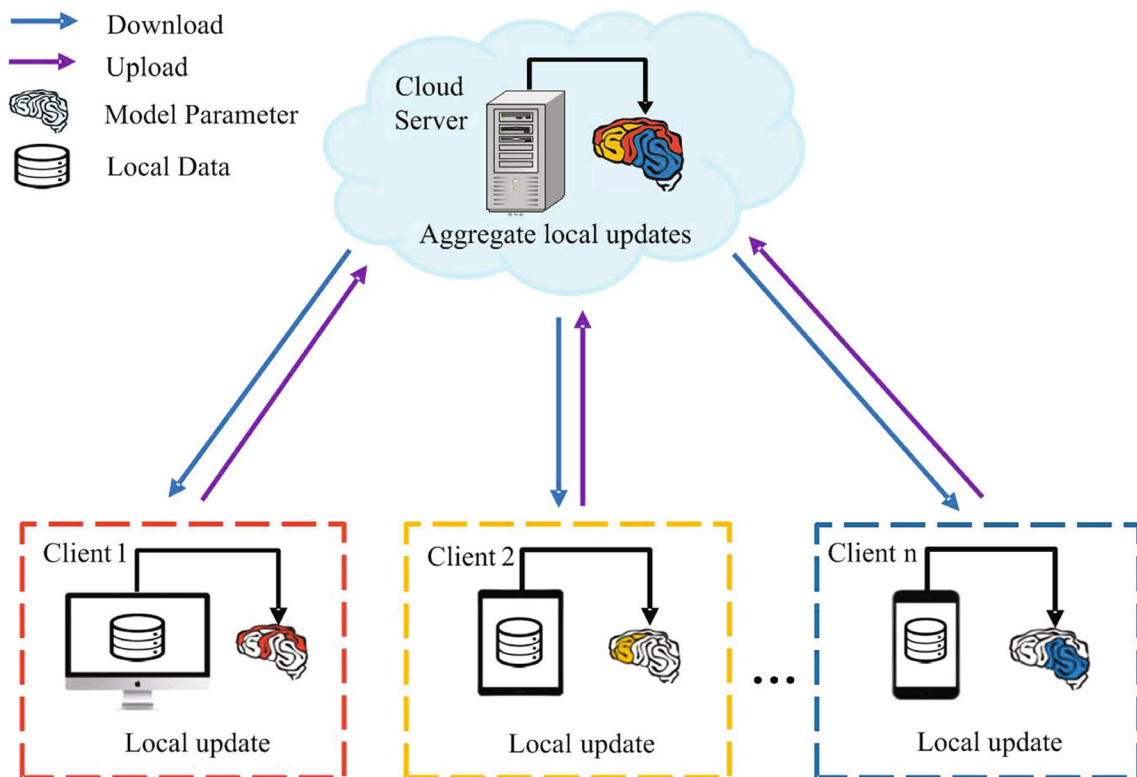
### 3 Background and preliminaries

This section first describes the architecture and the threat model of the FLCP system and then discusses the primary cryptographic techniques that constitute the FLCP. Finally,

this section illustrates the essential implementation steps of HE and introduces the basic principles and properties of DP.

#### 3.1 Federated learning system

We consider an FL system consisting of  $n$  clients and a cloud server, as shown in Fig. 1. Let  $D_i = \{X_1^i, X_2^i, \dots, X_m^i\}$  denote the local dataset with  $m$  data points, held by client  $U_i$ , where  $i \in \{1, 2, \dots, n\}$ . Under the server's orchestration, the client uses local data to train the shared model  $\theta$  collaboratively. Because of the high amount of latency and privacy issues in the process of transmitting local data points to the central



**Fig. 1** System architecture of FL

server, FL enables clients to retain data locally while training the model. Specifically, the model parameter  $\theta$  is trained by minimizing the global loss function,

$$\min_{\theta} f(\theta) = \frac{1}{n} \sum_{i=1}^n f_i(\theta) \text{ with } f_i(\theta) = \frac{1}{m} \sum_{X \in D_i} l(\theta, X), \quad (3.1)$$

where  $l(\theta, X)$  denotes the loss function of model  $\theta$  at sampling point  $X$  and  $f_i$  is client  $i$ 's local empirical risk function. For ease of description, some main notations in this paper are summarized in Table 2.

In FL, a global shared model is collectively learned by  $n$  clients that possess different data structures under the central server's orchestration. The training process of this model generally involves the following four stages: *local training*, *model uploading*, *model aggregation* and *model broadcasting*. For centralized training, computational costs are the main bottleneck, while communication costs dominate in FL. Since the local dataset is usually very small relative to the entire dataset, and the current smart devices on the edge have high-speed processors, the computing costs are nearly negligible. However, in cloud computing, the communication resources between the client and the server are usually limited, particularly for uplink transmission. Therefore, how to improve communication efficiency becomes the

main challenge for FL. To solve this problem, an adaptive compression algorithm is proposed to improve communication efficiency during training. In addition, cryptographic methods are utilized to ensure the privacy of the transmission updates and model accuracy. Ultimately, both communication efficiency and privacy protection are achieved under our proposed scheme.

### 3.2 Threat model

The proposed scheme assumes that the clients and the cloud server are "honest-but-curious" semitrusted entities. In other words, all entities (clients and servers) faithfully follow the designed training scheme but may attempt to deduce private information from the shared message. Moreover, we also assume that an adversary colludes with the cloud server or clients. This situation means that corrupted participants may disclose sensitive information to other malicious entities. Furthermore, the opponent may also be the passive external aggressor. Although these adversaries can eavesdrop on all shared information during training, they do not actively interrupt the transmission of information or inject false information. Therefore, the security objective of our protocol is to protect clients' privacy throughout the training process. According to the above privacy threat model, the following confidentiality requirements are developed:

- *Privacy security for local client parameters:* An adversary, malicious cloud server or client, can utilize global parameters and shared model updates to recover sensitive client information, such as contribution and membership information. Therefore, the client's local parameters need to be encrypted before being transmitted to the server to prevent the client's private information from being leaked.
- *Privacy security for aggregated results:* To make the training process non-discriminatory and fair, the reliability of each client ought to be kept confidential and cannot be inferred by other entities in the training process. Moreover, the aggregated result can be considered valuable intellectual property generated from multiple resources, potentially containing proprietary information about certain clients. As a result, the aggregation results are kept confidential from the opponent, except for those clients participating in training.

### 3.3 Homomorphic encryption

Based on the type and method of the supported mathematical operations, HE schemes can be divided into the following: partially HE algorithms that support a mathematical operation without limiting the number of operations; somewhat HE algorithms that support specific mathematical operations

**Table 2** Summary of mathematical symbols

Symbol	Definition
$N, i$	Total number and index of clients
$D, X, m$	Total number, datapoint and index of samples
$T, t$	Total number and index of communication rounds
$\lambda, c$	Total number and index of local updates
$B$	Batch size
$\theta$	Global model
$\theta_i$	Local model of client $i$
$\theta_i^*$	Compressed local model of client $i$
$\theta_i^{t,c}$	$i^{\text{th}}$ Client's model at round $t$ and step $c$
$l(\theta, X)$	Loss function of model $\theta$ at the datapoint $X$
$f(\theta)$	Global loss function
$f_i(\theta)$	Local loss function of client $i$
$p$	Compression rate
$R$	Residual term
$\eta$	Learning rate
$\varepsilon$	Privacy budget
$\mu$	Change in accuracy
$\delta$	Accuracy deviation
$C_i$	Encrypted local model of client $i$
$C_{agg}$	Encrypted global model

with a limited number of operations; and fully HE algorithms that support unlimited types of mathematical operations with unlimited operations. This research adopts the Paillier HE algorithm [34–36], which includes the following four steps:

1) *KeyGen*( $\rightarrow(pk, sk)$ ): Randomly select two large prime numbers  $b$  and  $d$  randomly, satisfying  $gcd(pq, (b - 1)(d - 1)) = 1$ . Calculate  $N = bd, \lambda = lcm(b - 1, d - 1)$ . Randomly select  $g \in Z_{N^2}^*$ , then public key  $pk = (N, g)$ , secret key  $sk = (\lambda)$ .

2) *Encryption*( $pk, m$ )  $\rightarrow C$ : Enter the public key  $pk$  and plaintext  $m$ . Randomly select  $r \in Z_{N^2}^*$ , ciphertext  $C = g^m r^N \pmod{N^2}$ .

3) *Decryption*( $sk, C$ )  $\rightarrow m$ : Enter the secret key  $sk$  and ciphertext information  $C$ , where  $L(x) = \frac{x-1}{N}$ . Calculate plaintext  $m = \frac{L(C^\lambda \pmod{N^2})}{L(g^\lambda \pmod{N^2})} \pmod{N}$ .

4) *Verification algorithm*:

$$\begin{aligned} & E(m_1) \times E(m_2) \\ &= (g^{m_1} r_1^N \pmod{N^2}) \times (g^{m_2} r_2^N \pmod{N^2}) \\ &= g^{m_1+m_2} (r_1 + r_2)^N \pmod{N^2} \\ &= E(m_1 + m_2). \end{aligned} \tag{3.2}$$

### 3.4 Differential privacy

Differential privacy (DP) is a cryptographic mechanism with powerful mathematical underpinnings, ensuring that the overall statistical information remains the same, regardless of changes in individual tuples. In 2008, Dwork et al. [37] proposed DP protection by injecting noise to make the query results of the two datasets undistinguishable.

**Definition 1** A randomized query function  $F$  satisfies the  $\epsilon$ -differential if for any adjacent databases  $D$  and  $\bar{D}$ , and any  $S \subset Range(F)$ ,

$$\left| \log \frac{P[F(D) = S]}{P[F(\bar{D}) = S]} \right| \leq \epsilon. \tag{3.3}$$

Parameter  $\epsilon$  refers to the privacy budget, which indicates the current level of privacy protection. Generally, a smaller  $\epsilon$  value provides a higher privacy protection level, but the training model under such a condition is less accurate.

Given the above DP mechanism, choosing a suitable noise level is a meaningful research issue that affects the client’s privacy security and the model’s convergence in the FL process.

## 4 Proposed FLCP

This section presents the FLCP to address communication overhead and privacy disclosure issues in FL. First, we

outline how to use the AWC-FedAvg method to reduce the communication cost of the training model. Next, a hybrid approach is developed to protect the privacy of each client within the system. Finally, an overall algorithm including all of these components is proposed, and the system architecture of the FLCP is presented.

### 4.1 Improving communication efficiency with the AWC-FedAvg method

The FedAvg-based weight compression algorithm is proposed to reduce the amount of single communication traffic and communication time in this paper. Because the fixed compression rate adopted in the traditional compression algorithm affects the convergence rate and model accuracy, and the training data of FL are non-IID and unbalanced, this algorithm is not suitable for the FL setting. Therefore, it is necessary to set a reasonable compression rate to fully utilize the performance of the compression algorithm. The adaptive weight compression FedAvg (AWC-FedAvg) method is based on the Top-K compression algorithm [27], which adaptively adjusts the compression rate in the dynamic perception model’s training stage. The algorithm can appropriately increase or reduce the number of local update parameters according to the model’s training conditions and improve training efficiency while ensuring the training accuracy of the neural network model. Figure 2 illustrates the overall flow of the AWC-FedAvg algorithm, of which there are four fundamental stages: *local update*, *compression weight*, *server aggregation update*, and *feedback regulation*.

1) *Local update*: Once the cloud server initializes the model parameters, it randomly selects a set of  $r$  clients  $N_t$  to broadcast the model parameters to the local client. Then, the selected clients perform several local iterations and send the local update results to the cloud server. Specifically, let  $\theta_i^{(t,c)}$  be client  $i$ ’s model at the  $c$ -th local iteration of the  $t$ -th round. In each local iteration  $c = 0, \dots, \lambda - 1$ , client  $i$  updates the model by

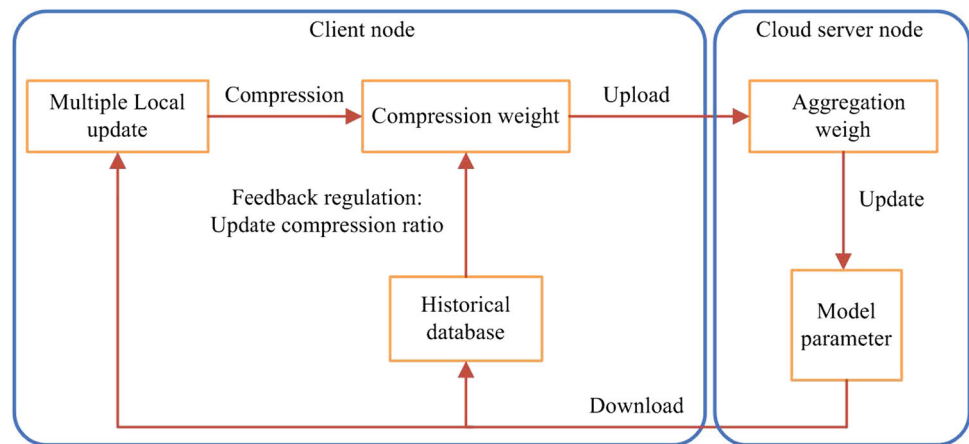
$$\theta_i^{(t,c+1)} = \theta_i^{t,c} - \eta g(\theta_i^{t,c}), \tag{4.1}$$

where  $g(\theta_i^{t,c}) = \frac{1}{B} \sum_{X \in D_i} \nabla l(\theta_i^{t,c}, X)$  represents the mini-batch SGD computed on the basis of a batch of  $B$  sampling points  $X_i$  of the local dataset  $D_i$ .

2) *Compression weight*: After several local SGD updates, the weight is compressed before uploading to the server. In this paper, the compression algorithm developed by P. Luo et al. [27] is improved, so the transmission data is decreased through transmitting Top-K weight parameters with adaptive compression rate, expressed as

$$\theta_i^* = Top_p(sort(|\theta_1|, |\theta_2|, \dots, |\theta_n|)), \tag{4.2}$$

**Fig. 2** The overall flow of the AWC-FedAvg algorithm



**Algorithm 1** Adaptive adjustment of compression rate.

**Input:** Server aggregate update  $\nabla\theta$

**Output:** Compression rate  $p$

```

1: Init: All clients  $U_i$  use the maximum compression rate  $p_{\max}$ 
2: for  $i \in N_t \subseteq \{1, \dots, n\}$  in parallel do
3:   Client  $U_i$  does:
4:      $\Delta\theta \leftarrow \text{Download}(\Delta\theta)$ 
5:      $accuracy_{\text{history}} \leftarrow \text{database}$ 
6:      $accuracy_{\text{current}} \leftarrow \text{verification set}(\Delta\theta)$ 
7:      $\mu \leftarrow |accuracy_{\text{history}} - accuracy_{\text{current}}|$ 
8:      $p \leftarrow p_{\max} - \frac{2(p_{\max} - p_{\min})}{1 + e^{|\mu|*\delta}}$ 
9:      $\text{database} \leftarrow accuracy_{\text{current}}$ 
10: end for
11: return  $p$ 

```

where  $|\theta_n|$  is the absolute value of the weight, arranged in descending order.  $Top_p$  chooses the weight of the largest  $p$ , which denotes the selection operator.  $p$  is the weight's compression rate, which is given by

$$p = \frac{\text{size}[\text{compressed}(\theta)]}{\text{size}[\theta]} \times 100\%, \quad (4.3)$$

where  $\text{size}[\text{compressed}(\theta)]$  represents the size of compressed model parameters, and  $\text{size}[\theta]$  represents the size of global model parameters. This method solves the problem that the weight is too small to be updated when the threshold is used as the cut-off point. In order to make full use of all the parameters of the iteration, AWC-FedAvg employs the residual accumulation method to accumulate small weights so that it can synchronize with the cloud server at a certain future moment without losing the model accuracy. The residual term is updated after each communication round by

$$R_\tau = \sum_{t=1}^{\tau} (\theta_t - \theta_t^*) = R_{\tau-1} + \theta_t - \theta_t^*, \quad (4.4)$$

where  $R_\tau$  represents the residual term. Eventually, these weights become sufficiently large to transmit.

3) *Server aggregate update:* The client compresses its local model parameters and uploads them to the cloud server.

The global shared model is aggregated by

$$\Delta\theta \leftarrow \frac{1}{|N_t|} \sum_{i \in N_t} \theta_i^*. \quad (4.5)$$

After the cloud server updates the global model by aggregating local parameters, it broadcasts the global shared model to the client.

4) *Feedback regulation:* As a critical step in the algorithm, feedback regulation refers to adaptively adjusting the compression rate of the present iteration based on the accumulated information. Algorithm 1 demonstrates the flow of the adaptive adjustment compression rate in which  $p_{\min}$  is the minimum compression rate,  $p_{\max}$  represents the maximum compression rate, and  $\delta$  denotes the accuracy deviation. The weight compression algorithm is designed with the sigmoid function, and its compression rate update is expressed as

$$p \leftarrow p_{\max} - \frac{2(p_{\max} - p_{\min})}{1 + e^{|\mu|*\delta}}, \quad (4.6)$$

where  $\mu$  represents the change in accuracy. Since the focus only lies on the trend of accuracy change, instead of the positive and negative fluctuations of accuracy change, the compression algorithm takes the absolute value of the accuracy change. The deviation of the accuracy change is denoted as  $\delta$ . It is a certain constant and is employed to control the sensitivity of the compression rate variation. Specifically, the larger  $\delta$  makes the compression rate become more sensitive to the trend of accuracy change and the compression rate change become faster.

Finally, the client updates the compression rate  $p$  through feedback regulation, and waits to enter the next cycle. The details are formalized in Algorithm 2. The proposed method's strength lies in adaptively adjusting the compression rate of each iteration through the model performance. This algorithm is applied to each client. Based on the data distribution of different clients, the client can enjoy a specific

**Algorithm 2** Adaptive Weight Compression FedAvg (AWC-FedAvg).

**Input:** Number of rounds  $T$ , number of selected clients  $r$ , round length  $\lambda$ , stepsize  $\eta$ .  
**Output:** Improved parameters  $\theta$

- 1: **Init:** All clients  $U_i$  use the maximum compression rate  $p_{\max}$ , initial global weight update and the residuals are set to zero  $\Delta\theta, R_i \leftarrow 0$ .
- 2: **for**  $t = 1, \dots, T$  **do**
- 3:     Server uniformly selects  $r$  clients denoted by  $N_t$
- 4:     Server broadcasts  $\theta^t$  to all clients in  $N_t$
- 5:     **for**  $i \in N_t \subseteq \{1, \dots, n\}$  **in parallel do**
- 6:         Client  $N_i$  does:
- 7:          $\Delta\theta \leftarrow \text{Download}(\Delta\theta)$
- 8:          $\theta_i \leftarrow \Delta\theta + R_i$
- 9:         **for**  $c = 0$  to  $\lambda - 1$  **do**
- 10:             Randomly sample a batch of datapoints  $X_i$   
            with size  $B$  from the local dataset  $D_i$
- 11:              $\theta_i^{t,c+1} = \theta_i^{t,c} - \eta g(\theta_i^{t,c})$
- 12:             **end for**
- 13:              $\theta_i^* \leftarrow \text{Top}_p(\theta_i^{t,c+1})$
- 14:              $R_i \leftarrow \theta_i - \theta_i^*$
- 15:             Upload  $(\theta_i^*)$
- 16:         **end for**
- 17:     Server S does
- 18:     Gather  $(\theta_i^*), i \in N_t$
- 19:      $\Delta\theta \leftarrow \frac{1}{|N_t|} \sum_{i \in N_t} \theta_i^*$
- 20:     Broadcast  $(\Delta\theta)$
- 21: **end for**
- 22: **return**  $\theta$

compression rate, which makes the training convergence faster and the convergence accuracy higher.

## 4.2 Improving security performance with the FLCP

### 4.2.1 Preventing privacy leakage with differential privacy

The above approach to reducing communication overhead can effectively prevent the direct leakage of client information by saving the raw data locally. However, this approach cannot prevent more advanced attacks such as [15] and [38], which infer sensitive information from the local dataset by eavesdropping on the message traffic between the client and the server. According to the proposed threat model, the client and the server are semi-trusted “honest-but-curious” entities, and malicious adversaries outside the system may collude with the honest client to eavesdrop on the sent model parameters. These attackers can receive the latest global shared model  $\Delta\theta$  that the cloud server sends to the client and the local model  $\theta_i^*$  that the client sends to the cloud server, which contain private information from the client’s dataset. Our security goal is to protect against the disclosure of these two types of private information.

Within our setting, a straightforward approach is the use of the Laplace mechanism. Specifically, each client  $i \in N_t$  adds proper Laplace noise into the local model parameters before uploading the information, and the local model update

is shown in (4.7). Thus, in this case, the adversary does not acquire private information about the individual sample in  $D_i$  based on the obtained model parameters.

$$\bar{\theta}_i^* \leftarrow \theta_i^* + \text{Lap}\left(\frac{\Delta f}{\epsilon}\right). \tag{4.7}$$

Here,  $\Delta f$  denotes global sensitivity, expressed as the maximum value of  $|f(D) - f(\bar{D})|_1$  for adjacent datasets  $D$  and  $\bar{D}$ .  $\text{Lap}(\frac{\Delta f}{\epsilon})$  is the random variable sampled from the Laplace distribution that satisfies  $Pr[\text{Lap}(\frac{\Delta f}{\epsilon}) = x] = \frac{\epsilon}{2\Delta f} e^{-\frac{|x| \epsilon}{\Delta f}}$  [39]. In our scheme, function  $f$  calculates each client’s weight during one epoch.

### 4.2.2 Secure aggregation scheme

Although the Laplace mechanism can be used to implement DP, the model parameters uploaded in each iteration are plaintext and are, thus, still exposed to the adversary, resulting in the disclosure of private information. In fact, the server merely requires that the average of the local model be obtained. Therefore, the privacy leakage of the client can be reduced by concealing the client’s local model and limiting the server to merely obtaining the aggregation results of the encrypted local models without affecting the training process. In brief, this approach can be implemented through a secure aggregation scheme, in which the server merely obtains the aggregation results of the encrypted local model without knowing the local model of each client. Thus, the FL privacy protection communication scheme proposed in [22] is adopted to establish different TLS/SSL secure channels between the client and the server. In addition, a lightweight encryption protocol is developed based on HE, which contributes to alleviating the high levels of calculation and communication overhead needed for all communication rounds.

In our setup, the secure aggregation scheme hides clients’ individual information, restores the sum of clients’ individual information in each round, and keeps clients’ communication costs low [25]. Therefore, the proposed secure aggregation scheme consists of the following main steps:

1) *Initialization:* Considering the security parameter  $\rho$ , the secret key  $sk$  is generated and allocated to each client, which includes two large prime numbers  $b, d$  ( $|b| = |d| = \rho$ ). The public parameter is  $N = bd$ .

2) *Encryption phase:* Subsequently, the local weights  $\theta_i$  are encrypted with secret key  $b, d$ , that is, calculate  $\theta_{i,b} \equiv \theta_i \pmod b, \theta_{i,d} \equiv \theta_i \pmod d$ . Since we have  $1 \equiv b^{-1}b \pmod d, 1 \equiv d^{-1}d \pmod b$ , the local client computes the ciphertext like this:

$$C_i = d^{-1}d\theta_{i,b}^b + b^{-1}b\theta_{i,d}^d \pmod N, \tag{4.8}$$



where  $b^{-1}$  and  $d^{-1}$  represent the reciprocal of  $b$  and  $d$ , respectively. The encrypted local updates  $C_i$  from the clients participating in the training are transmitted to the server.

3) *Aggregation phase*: After obtaining the encrypted updates of the clients, given its computing power, the cloud server aims to perform the aggregation operation as follows:

$$\begin{aligned} C_{agg} &= \sum_{i=1}^n C_i \\ &= d^{-1}d \sum_{i=1}^n \theta_{i,b}^b + b^{-1}b \sum_{i=1}^n \theta_{i,d}^d \pmod{N} \\ &= d^{-1}d \left( \sum_{i=1}^n \theta_{i,b} \right)^b + b^{-1}b \left( \sum_{i=1}^n \theta_{i,d} \right)^d \pmod{N}. \end{aligned} \quad (4.9)$$

Afterwards, the server starts to communicate with the local clients and releases the encrypted global updates  $C_{agg}$  to avoid the adversary's attack.

4) *Decryption phase*: After the local clients receive the encrypted global updates  $C_{agg}$ , each client starts the decryption operation by

$$\begin{aligned} C_{agg} \pmod{b} &= d^{-1}d \left( \sum_{i=1}^n \theta_{i,b} \right)^b + b^{-1}b \left( \sum_{i=1}^n \theta_{i,d} \right)^d \pmod{b} \\ &= d^{-1}d \left( \sum_{i=1}^n \theta_{i,b} \right)^b \pmod{b} \\ &= d^{-1}d \left( \sum_{i=1}^n \theta_{i,b} \right)^{b-1} \left( \sum_{i=1}^n \theta_{i,b} \right) \pmod{b} \\ &= \left( \sum_{i=1}^n \theta_{i,b} \right) \pmod{b} \\ &= \theta_{agg,b} \pmod{b}. \end{aligned} \quad (4.10)$$

In the same way,

$$\begin{aligned} C_{agg} \pmod{d} &= d^{-1}d \left( \sum_{i=1}^n \theta_{i,b} \right)^b + b^{-1}b \left( \sum_{i=1}^n \theta_{i,d} \right)^d \pmod{d} \\ &= \left( \sum_{i=1}^n \theta_{i,d} \right) \pmod{d} \\ &= \theta_{agg,d} \pmod{d}. \end{aligned} \quad (4.11)$$

This formula applies Euler Theorem and  $gcd\left[\left(\sum_{i=1}^n \theta_{i,b}\right), b\right] = 1$ . According to the above operation, the local clients obtain the decrypted result model  $\theta_{agg}$  by utilizing the Chinese Remainder Theorem (CRT) as:

$$\begin{cases} \theta_{agg} \equiv \theta_{agg,b} \pmod{b}, \\ \theta_{agg} \equiv \theta_{agg,d} \pmod{d}. \end{cases} \quad (4.12)$$

Compute the congruence expressions:

$$\theta_{agg} = M_b d \theta_{agg,b} + M_d b \theta_{agg,d} \pmod{N}, \quad (4.13)$$

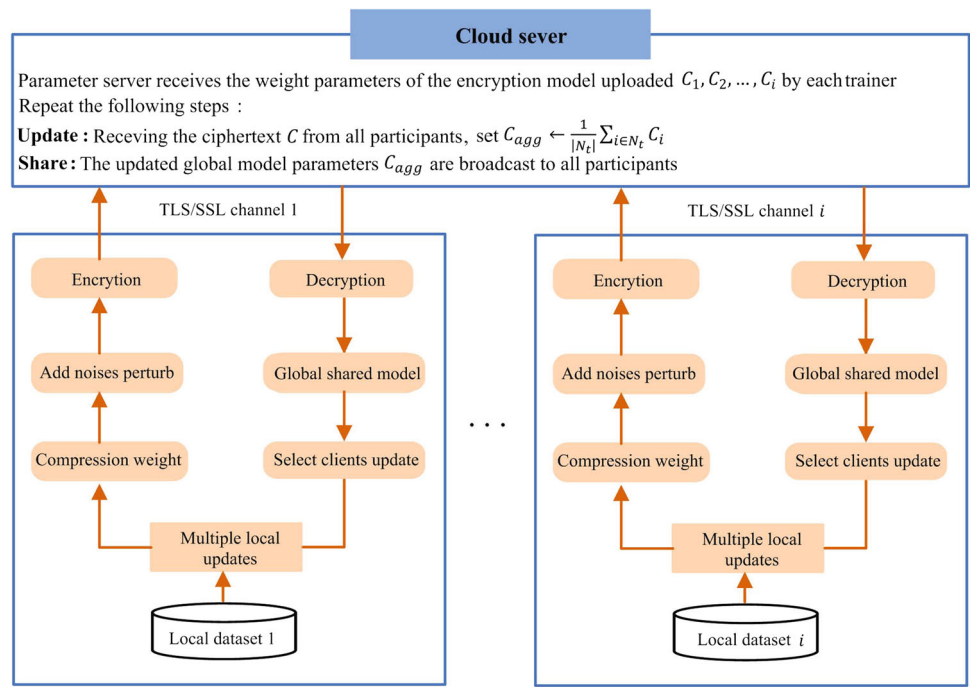
where  $M_b d \equiv 1 \pmod{b}$  and  $M_d b \equiv 1 \pmod{d}$ . Due to  $gcd(b, d) = 1$ , it is easy to calculate  $M_b$  and  $M_d$ . It is worth noting that since our proposed secure aggregation scheme requires only that each client encrypts the local model parameters using the same public key and uploads them to the cloud server, when the client drops out, the decryption is not affected. By comparing the TP-SMC [30] and PPFDL [31] encryption schemes, it is shown that the training process affects the subsequent ciphertext operation if any client fails to upload data. Therefore, the FLCP is robust to clients dropping out during training and is proven to have a higher degree of model accuracy than do the TP-SMC and PPFDL in Section 6. Furthermore, since the sender and receiver are identical in our system, our secure aggregation further removes the ciphertext component, supporting the usage of this system in large-scale scenarios. Finally, clients update the local parameter based on the global shared model.

### 4.3 The overall scheme of FLCP

In this paper, the trainers in FL, including computers, mobile phones, smart devices, etc., are collectively taken as clients. We construct the FLCP framework, which contains several clients and a cloud server, as shown in Fig. 3. The clients and cloud servers are "honest-but-curious" semi-trusted entities, by which all entities (clients and servers) will faithfully follow the designed training protocol but may attempt to deduce private information from the shared messages. On account of the issue of privacy, most clients are reluctant to expose their data to cloud servers and other clients, but they want to learn high-precision models from the data combination. In particular, the client trains the model on the local dataset in a synchronous manner, then compresses the updated model parameters and transmits them to the server in ciphertext form. According to the property of HE, the server aggregates the encrypted local model and broadcasts the aggregation results calculated on the ciphertext to the client to update the local model. Through multiple rounds of communication, the client finally obtains a global model that meets the convergence requirements without explicitly disclosing individual datasets.

The overall protocol of FLCP is summarized in Algorithm 3. Our protocol involves  $T$  communication rounds, and in each round, a group of clients is selected to execute  $\lambda$  local iterations. Specifically, at each round  $t = 0, \dots, T$ , the server initially selects  $r \leq n$  clients uniformly and randomly, denoted as  $N_t$ . Then the server broadcasts its current shared

**Fig. 3** System architecture of FLCP



model parameters  $\theta^t$  to the clients  $N_t$ , and each client  $i \in N_t$  executes  $\lambda$  local iterations in their local dataset  $D_i$  according to (4.2). After  $\lambda$  local iterations, the clients in  $N_t$  compress weights by  $p\%$  and upload an encrypted local model  $C_i$  to the server. Finally, the server aggregates the encrypted messages to compute the next global model, and the procedure is repeated for  $T$  rounds.

### 5 Theoretical analysis

In this section, the convergence of the FLCP is first analyzed under the condition of the use of the periodic averaging method and the adaptive compression method. Then, a rigorous security analysis of the FLCP is conducted.

#### 5.1 Convergence analysis

The convergence bounds for the FLCP with non-convex losses are presented. To deduce the convergence bounds of the model, we require the following assumptions.

- Assumption 1.** (Smoothness) Let's assume that a function  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  is  $L$ -smooth, for any  $x, y \in \mathbb{R}^d$ , we have  $f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2} \|x - y\|^2$ .
- (Unbiased gradient) For any  $i, x \in \mathbb{R}^d, \mathbb{E}[\nabla f_i(x)] = \nabla f(x)$ .
  - (Second moments and bounded variances) For any  $i, x \in \mathbb{R}^d$ , there exist positive constants  $\sigma$  and  $G, \mathbb{E}\|\nabla f_i(x)\|^2 \leq G^2, \mathbb{E}\left[\|\nabla f_i(x) - \nabla f(x)\|^2\right] \leq \sigma^2$ .

#### Algorithm 3 FLCP algorithm.

```

Input: Number of rounds  $T$ , number of selected clients  $r$ , round length  $\lambda$ , stepsize  $\eta$ .
Output: Improved parameters  $\theta$ 
1: Init: All clients  $U_i$  use the maximum compression rate  $p_{\max}$ , initial global weight update and the residuals are set to zero  $\Delta\theta, R_i \leftarrow 0$ .
2: for  $t = 1, \dots, T$  do
3:   Server uniformly selects  $r$  clients denoted by  $N_t$ 
4:   Server broadcasts  $\theta^t$  to all clients in  $N_t$ 
5:   for  $i \in N_t \subseteq \{1, \dots, n\}$  in parallel do
6:     Client  $N_i$  does:
7:      $C_{agg} \leftarrow \text{Download}(C_{agg})$ 
8:      $\Delta\theta \leftarrow \text{Decrypt}(C_{agg})$ 
9:      $\theta_i \leftarrow \Delta\theta + R_i$ 
10:    for  $c = 0$  to  $\lambda - 1$  do
11:      Randomly sample a batch of datapoints  $X_i$  with size  $B$  from the local dataset  $D_i$ 
12:       $\theta_i^{t,c+1} = \theta_i^{t,c} - \eta g(\theta_i^{t,c})$ 
13:    end for
14:     $\theta_i^* \leftarrow \text{Top}_p(\theta_i^{t,c+1})$ 
15:     $R_i \leftarrow \theta_i - \theta_i^*$ 
16:     $\tilde{\theta}_i^* \leftarrow \theta_i^* + \text{Lap}(\frac{\Delta f}{\epsilon})$ 
17:     $C_i \leftarrow \text{Encrypt}(\tilde{\theta}_i^*)$ 
18:    Upload  $(C_i)$ 
19:  end for
20:  Server S does
21:  Gather  $(C_i), i \in N_t$ 
22:   $C_{agg} \leftarrow \frac{1}{|N_t|} \sum_{i \in N_t} C_i$ 
23:  Broadcast  $(C_{agg})$ 
24: end for
25: return  $\theta$ 
    
```

- (Compression operator) For a constant  $\alpha \in (0, 1)$ , a compression operator  $\text{Top}_p : \mathbb{R}^d \rightarrow \mathbb{R}^d$  that for all  $x \in \mathbb{R}^d, \mathbb{E}\|x - \text{Top}_p(x)\| \leq (1 - \alpha) \|x\|^2$ .

where  $L$  represents the Lipschitz constant, implying that the loss function  $f$  is  $L$ -smooth. Conditions 2 and 3 in the assumption regarding the bias and variance of the mini-batch gradients are customary for SGD methods. Condition 4 in the assumption denotes selecting  $p$  out of  $d$  coordinates, and  $p$  coordinates with the highest magnitude values ( $Top_p$ ) give  $\alpha = \frac{p}{d}$ .

The protocol involves  $K$  iterations, and during iteration  $k$ , each client  $i$  calculates the stochastic gradient  $g(\theta_i^k)$  in its local dataset and updates the current model  $\theta_i^k$ . In addition, the residual accumulation method is adopted, and the errors generated in each iteration are accumulated in the memory of each client and compensated in future updates. This updating method is the key to maintaining model accuracy, which provides clients with a controlled way to use both the current update and the residual update from previous rounds of communication.

To facilitate the analysis of FLCP convergence, the update rule is redefined as

$$\begin{aligned}\bar{\theta}^{k+1} &= \bar{\theta}^k - \eta \left[ \frac{1}{n} \sum_{i=0}^n g(\theta_i^k) \right], \\ R^{k+1} &:= R^k + \sum_{i=1}^n \eta \nabla f(\theta_i^k) - \theta^*,\end{aligned}\quad (5.1)$$

where  $\bar{\theta}^k$  is the averaged model at iteration  $k$ , and  $R^k$  is the residual term at iteration  $k$ .

**Lemma 1** For every  $k \in \mathbb{Z}$  and fixed learning rate  $\eta$ , the following holds for each client  $i \in [n]$ :

$$\mathbb{E} \|R^k\|^2 \leq \frac{4(1-\alpha^2)}{\alpha^2} \eta^2 G^2. \quad (5.2)$$

**Proof** According to Conditions 3 and 4 in the assumption and the inequality  $\|a+b\|^2 \leq (1+\beta)\|a\|^2 + (1+\frac{1}{\beta})\|b\|^2$  for every  $\beta > 0$  (taking any  $z > 1$  in the following), we have that

$$\begin{aligned}\mathbb{E} \|R^{k+1}\|^2 &\leq (1-\alpha) \mathbb{E} \left\| R^k + \sum_{i=1}^n \eta \nabla f(\theta_i^k) \right\|^2 \\ &\leq (1-\alpha) \left( 1 + \frac{(z-1)\alpha}{z} \right) \mathbb{E} \|R^k\|^2 + \\ &(1-\alpha) \left( 1 + \frac{z}{(z-1)\alpha} \right) \mathbb{E} \left\| \sum_{i=1}^n \eta \nabla f(\theta_i^k) \right\|^2 \\ &\leq \left( 1 - \frac{\alpha}{z} \right) \mathbb{E} \|R^k\|^2 + \frac{z(1-\alpha^2)}{(z-1)\alpha} \eta^2 G^2.\end{aligned}\quad (5.3)$$

When (5.3) is expanded, we find that the upper bound of the residual is a geometric sum:

$$\begin{aligned}\mathbb{E} \|R^{k+1}\|^2 &\leq \frac{z(1-\alpha^2)}{(z-1)\alpha} \eta^2 G^2 \sum_{j=0}^{\infty} \left( 1 - \frac{\alpha}{z} \right)^j \\ &\leq \frac{z^2(1-\alpha^2)}{(z-1)\alpha^2} \eta^2 G^2.\end{aligned}\quad (5.4)$$

Inequality (5.4) is true for  $z > 1$ , and when  $z = 2$ , the value of the inequality is minimized. Taking  $z = 2$ , we obtain that

$$\mathbb{E} \|R^{k+1}\|^2 \leq \frac{4(1-\alpha^2)}{\alpha^2} \eta^2 G^2. \quad (5.5)$$

Since the right-hand side of this inequality does not depend on  $k$ , it follows that  $\mathbb{E} \|R^k\|^2 \leq \frac{4(1-\alpha^2)}{\alpha^2} \eta^2 G^2$  holds for every  $k \in K$ . In addition,  $\bar{\theta}^k$  is the average of the compression model, and the updated mean residual is  $\theta_i^k - \bar{\theta}^k = \frac{1}{n} \sum_{i=1}^n R^k$ .

**Theorem 1** Let  $f_i$  be  $L$ -smooth, each  $i \in [n]$ . For a constant  $\alpha \in (0, 1]$ , a compression operator  $Top_p : \mathbb{R}^d \rightarrow \mathbb{R}^d$ . Let  $\{\theta_i^k\}_{k=0}^{K-1}$  be generated based on Algorithm 3, for fixed learning rate  $\eta$ . Then we have that

$$\begin{aligned}\mathbb{E} \left[ \frac{1}{Kn} \sum_{k=0}^{K-1} \sum_{i=1}^n \|\nabla f(\theta_i^k)\|^2 \right] &\leq \frac{4[f(\bar{\theta}^0) - f^*]}{\eta K} \\ &+ \frac{2\eta\sigma^2 L}{n} + \frac{16(1-\alpha^2)}{n^2\alpha^2} L^2 \eta^2 G^2.\end{aligned}\quad (5.6)$$

**Proof** According to Condition 1 in the assumption, we have that

$$\begin{aligned}&f(\bar{\theta}^{k+1}) - f(\bar{\theta}^k) \\ &\leq \langle \nabla f(\bar{\theta}^k), \bar{\theta}^{k+1} - \bar{\theta}^k \rangle + \frac{L}{2} \|\bar{\theta}^{k+1} - \bar{\theta}^k\|^2 \\ &= -\eta \langle \nabla f(\bar{\theta}^k), g^k \rangle + \frac{\eta^2 L}{2} \|g^k\|^2 \\ &= -\eta \langle \nabla f(\bar{\theta}^k), g^k \rangle + \frac{\eta^2 L}{2} \|g^k - \hat{g}^k + \hat{g}^k\|^2 \\ &\stackrel{(a)}{\leq} -\eta \langle \nabla f(\bar{\theta}^k), g^k \rangle + \eta^2 L \|g^k - \hat{g}^k\|^2 + \eta^2 L \|\hat{g}^k\|^2 \\ &= -\frac{\eta}{n} \sum_{i=1}^n \langle \nabla f(\bar{\theta}^k), \nabla f_i(\theta_i^k) \rangle + \eta^2 L \left\| \frac{1}{n} \sum_{i=1}^n \nabla f(\theta_i^k) \right\|^2 \\ &+ \eta^2 L \|g^k - \hat{g}^k\|^2,\end{aligned}\quad (5.7)$$

where  $g^k$  represents the average mini-batch gradients,  $\hat{g}^k$  represents the average full-batch gradients, and inequality

(a) is based on Jensen’s inequality. Taking the expectation of sampling at iteration  $k$ , based on the Lipschitz continuity of the gradients of local functions, we have that

$$\begin{aligned}
 & \mathbb{E}[f(\bar{\theta}^{k+1})] - \mathbb{E}[f(\bar{\theta}^k)] \\
 & \leq -\frac{\eta}{2} \left\| \nabla f(\bar{\theta}^k) \right\|^2 - \frac{\eta}{2} \left\| \frac{1}{n} \sum_{i=1}^n \nabla f(\theta_i^k) \right\|^2 \\
 & \quad + \frac{\eta}{2} \left\| \nabla f(\bar{\theta}^k) - \frac{1}{n} \sum_{i=1}^n \nabla f(\theta_i^k) \right\|^2 \\
 & \quad + \eta^2 L \left\| \frac{1}{n} \sum_{i=1}^n \nabla f(\theta_i^k) \right\|^2 + \frac{\eta^2 \sigma^2 L}{2n} \\
 & \leq -\frac{\eta}{2n} \sum_{i=1}^n \left( \left\| \nabla f(\bar{\theta}^k) \right\|^2 - L^2 \left\| \bar{\theta}^k - \theta_i^k \right\|^2 \right) \tag{5.8} \\
 & \quad + \frac{2\eta^2 L - \eta}{2n} \sum_{i=1}^n \left\| \nabla f(\theta_i^k) \right\|^2 + \frac{\eta^2 \sigma^2 L}{2n} \\
 & \leq -\frac{\eta}{2n} \sum_{i=1}^n \left( \left\| \nabla f(\bar{\theta}^k) \right\|^2 + L^2 \left\| \bar{\theta}^k - \theta_i^k \right\|^2 \right) \\
 & \quad + \frac{2\eta^2 L - \eta}{2n} \sum_{i=1}^n \left\| \nabla f(\theta_i^k) \right\|^2 + \frac{\eta^2 \sigma^2 L}{2n} \\
 & \quad + \frac{\eta L^2}{n} \left\| \bar{\theta}^k - \theta_i^k \right\|^2.
 \end{aligned}$$

We bound the first term in terms of  $\left\| \nabla f(\theta_i^k) \right\|^2$  as:

$$\begin{aligned}
 \left\| \nabla f(\theta_i^k) \right\|^2 & \leq 2 \left\| \nabla f(\theta_i^k) - \nabla f(\bar{\theta}^k) \right\|^2 + 2 \left\| \nabla f(\bar{\theta}^k) \right\|^2 \\
 & \leq 2L^2 \left\| \theta_i^k - \bar{\theta}^k \right\|^2 + 2 \left\| \nabla f(\bar{\theta}^k) \right\|^2, \tag{5.9}
 \end{aligned}$$

where Inequality (5.9) follows from the  $L$ -Lipschitz gradient. Based on this, the expression is rearranged as

$$\begin{aligned}
 \frac{1}{n} \sum_{i=1}^n \mathbb{E} \left\| \nabla f(\theta_i^k) \right\|^2 & \leq \frac{4 \left[ \mathbb{E}[f(\bar{\theta}^k)] - \mathbb{E}[f(\bar{\theta}^{k+1})] \right]}{\eta} \\
 & \quad + \frac{2\eta\sigma^2 L}{n} + \frac{4L}{n} \mathbb{E} \left\| \bar{\theta}^k - \theta_i^k \right\|^2. \tag{5.10}
 \end{aligned}$$

From Lemma 1 we get  $\mathbb{E} \left\| \bar{\theta}^k - \theta_i^k \right\|^2 \leq \frac{4(1-\alpha^2)}{n\alpha^2} \eta^2 G^2$  and substitute it into formula (5.10) as

$$\begin{aligned}
 \frac{1}{n} \sum_{i=1}^n \mathbb{E} \left\| \nabla f(\theta_i^k) \right\|^2 & \leq \frac{4 \left[ \mathbb{E}[f(\bar{\theta}^k)] - \mathbb{E}[f(\bar{\theta}^{k+1})] \right]}{\eta} \\
 & \quad + \frac{2\eta\sigma^2 L}{n} + \frac{16(1-\alpha^2)}{n^2\alpha^2} L^2 \eta^2 G^2. \tag{5.11}
 \end{aligned}$$

By taking a telescopic sum from  $k = 0$  to  $k = K - 1$ , we obtain that

$$\begin{aligned}
 \mathbb{E} \left[ \frac{1}{Kn} \sum_{k=0}^{K-1} \sum_{i=1}^n \left\| \nabla f(\theta_i^k) \right\|^2 \right] & \leq \frac{4 \left[ f(\bar{\theta}^0) - f^* \right]}{\eta K} \\
 & \quad + \frac{2\eta\sigma^2 L}{n} + \frac{16(1-\alpha^2)}{n^2\alpha^2} L^2 \eta^2 G^2. \tag{5.12}
 \end{aligned}$$

## 5.2 Security analysis

### 5.2.1 Security against the cloud server

After reviewing the training process of the above FL algorithm, the intermediate data obtained by the client and the cloud server are illustrated in Table 3, which shows that in FL training, each client can obtain the global model parameters through secret key decryption but cannot acquire other client model parameters  $\theta_{par}$ , gradient  $g$ , *Predictionresults*, and *Loss*. Moreover, the cloud server obtains the encrypted local model parameter  $Enc(\theta_{par})$  and the encrypted global model parameter  $Enc(\theta_{global})$ . The cloud server does not have a secret key and cannot decrypt the parameter data.

As shown in Fig. 3, the central server averages the model parameters uploaded by clients and updates the global parameters of the neural network model. Moreover, the validity of the ciphertext calculation formula in the figure is guaranteed by the property of the HE scheme.

**Definition 2** (CPA-Security) [40]. For the HE scheme in this work, we describe the chosen plaintext attacks (CPA) as a game between adversary  $\mathcal{A}$  and challenger  $\mathcal{C}$ :

- **Initialization.**  $\mathcal{C}$  creates the CPA system and sends the generated public parameters  $pp$  and the key pairs  $(pk, sk)$  to  $\mathcal{A}$ .
- **Challenge.** Adversary  $\mathcal{A}$  selects two plaintexts  $m_0$  and  $m_1$  with the same length and sends them to  $\mathcal{C}$ . Challenger  $\mathcal{C}$  selects  $b \in \{0, 1\}$  at random and computes  $C^* = Enc(pk, m_b)$ . Then the challenge ciphertext  $C^*$  is returned to  $\mathcal{A}$ .
- **Guess.** Adversary  $\mathcal{A}$  guesses whether the plaintext encrypted in the previous step of challenger  $\mathcal{C}$  is  $m_0$  or

**Table 3** Data information obtained by the participant and parameter server

Name	Client $i$	Cloud sever
Intermediate data	$(\theta_{global})$	$Enc(\theta_{global})$
	$Predictionresults$	$Enc(\theta_{par,1})$
	$Loss_i$	$Enc(\theta_{par,2})$
	$g_i$	...
	$(\theta_{par,i})$	$Enc(\theta_{par,n})$

$m_1$ , and outputs the guess result, which is recorded as  $b'$ . If  $b' = b$ , the adversary attack succeeds.

The advantage of an adversary attack can be defined as the following function:

$$\text{Adv}_{\mathcal{A}}^{\text{CPA}}(\kappa) = |\Pr[b = b'] - \frac{1}{2}|, \quad (5.13)$$

where  $\kappa$  denotes the length of the encryption scheme key. HE scheme is secure against the chosen plaintext attacks (CPA-secure) if the advantage of an adversary attack is negligible in  $\kappa$ .

**Theorem 2** *Our protocol does not leak information from the datasets to an “honest-but-curious” parameter server as long as the HE scheme is CPA-secure.*

**Proof** It is assumed that adversary  $\mathcal{A}$  can compromise the cloud server and participants other than  $\gamma$  and  $\nu$  in each round of aggregation and query their secret key  $\{sk_i\}_{i \neq \gamma, i \neq \nu}$ . Then,  $\mathcal{A}$  encrypts the plaintext  $m_i$  of arbitrarily appointed participant  $i$  ( $i \neq \gamma, i \neq \nu$ ) using the secret key  $sk_i$ . Even if  $\mathcal{A}$  can obtain the key  $SK_c$  and the ciphertext of the participant, it can merely obtain the sum of  $m_\gamma$  and  $m_\nu$ . In other words,  $\mathcal{A}$  still fails to identify the ciphertext of  $\gamma$  and  $\nu$  from the random values. If  $\mathcal{A}$  can identify the ciphertext of  $\gamma$  and  $\nu$  from the random values, then the advantage of an adversary attack is not negligible in  $\kappa$ . That is,  $\mathcal{A}$  can solve the decision-augmented learning with error (LWE) problem, which is difficult in reality. As a result, our encryption protocol is CPA-secure, safeguarding the client’s data from compromise.

### 5.2.2 Security against the cloud server and compromised clients

The adversary may compromise with some clients by stealing the system’s private keys and the honest clients’ privacy. Thus, DP is utilized to provide strict privacy protection and prove that the local perturbation model satisfies different privacy requirements.

**Theorem 3** *The local model preserves  $\varepsilon$ -differential privacy. For any two adjacent datasets  $D$  and  $\bar{D}$ ,*

$$|\log \frac{P(\bar{\theta}_i^* | D)}{P(\bar{\theta}_i^* | \bar{D})}| \leq \varepsilon. \quad (5.14)$$

**Proof** Let  $\chi$  be the noise injected into  $\theta$  and  $\chi \sim \text{Lap}(\frac{\Delta f}{\varepsilon})$ . We have

$$\begin{aligned} P(\bar{\theta}_i^* | D) &= P[f_i(D) + \chi = t] \\ &= P[\chi = t - f_i(D)] \\ &= \frac{\varepsilon}{2\Delta f} \exp\left(\frac{-|t - f_i(D)|}{\Delta f}\right). \end{aligned} \quad (5.15)$$

Similarly,

$$P(\bar{\theta}_i^* | \bar{D}) = \frac{\varepsilon}{2\Delta f} \exp\left(\frac{-|t - f_i(\bar{D})|}{\Delta f}\right). \quad (5.16)$$

Thus,

$$\begin{aligned} \frac{P(\bar{\theta}_i^* | D)}{P(\bar{\theta}_i^* | \bar{D})} &= \frac{P[f_i(D) + \chi = t]}{P[f_i(\bar{D}) + \chi = t]} \\ &= \frac{\exp\left(\frac{-\varepsilon|t - f_i(D)|}{\Delta f}\right)}{\exp\left(\frac{-\varepsilon|t - f_i(\bar{D})|}{\Delta f}\right)} \\ &= \exp\left(\frac{\varepsilon(|t - f_i(\bar{D})| - |t - f_i(D)|)}{\Delta f}\right) \\ &\leq \exp\left(\frac{|f_i(\bar{D}) - f_i(D)|}{\Delta f}\right) \leq \exp(\varepsilon). \end{aligned} \quad (5.17)$$

Thus, perturbation of local updates preserves  $\varepsilon$ -DP, and the proposed security scheme tolerates the server to collude with any client, but no valuable information can be inferred.

## 6 Experiments

This section mainly conducts experiments to evaluate FLCP performance. We first briefly describe the experimental setup. Then, the influence of the key factors in the FLCP on convergence is studied. Finally, the performance of the FLCP is compared with several baseline approaches in terms of model accuracy, training time, and communication efficiency.

### 6.1 Experimental setup

To evaluate the performance of the FLCP, all clients are allowed to run a unified convolutional neural network (CNN), which includes two convolutional layers, an average pooling layer, and two fully connected layers. MNIST [41] and CIFAR-10 [42] are used as benchmarks. The former includes 60,000 training examples and 10,000 testing examples, each of which is a  $28 \times 28$  size gray-level image, while the latter contains 50,000 training examples and 10,000 testing examples, consisting of 10 classes of  $32 \times 32$  images with three channels (RGB). To simulate the FL setting, we assume that there are 12 clients in the system. Each local client’s dataset is assigned an approximately identical distribution by randomly shuffling and evenly partitioning the training dataset. Several state-of-the-art FL approaches with communication efficiency and privacy preservation, including NbAFL [19], TP-SMC [30], RCEFL [11], and PPFDL [31] listed in Table 1, as well as the well-known classical FL algorithm FedAvg [5], are selected as baselines for comparing the performance of our proposed scheme.

Comprehensive information about the cloud server and client module configurations in the experimental scenario is provided in Table 4. All clients have sufficient computing power to run the encryption and decryption algorithms. The compression and privacy protection algorithms are simulated in the Python3 language, and we use the PyTorch library. The parameters of the neural network model are encrypted and decrypted by using the open-source Paillier library.

### 6.2 Convergence property

In this subsection, the influence of several key factors on the convergence characteristics of the scheme is studied. Moreover, the influence of the key factors with different values on the degree of convergence is compared.

#### 6.2.1 Impact of compression rate $p$

To test the properties of the adaptive compression algorithm in the FLCP, we compare it with fixed compression rates of 0.00% (uncompressed), 99.00%, 99.60%, and 99.99% under the same conditions. In the adaptive compression algorithm, the minimum compression rate  $p_{min}$  is set to 99.00%, and the maximum compression rate  $p_{max}$  is set to 99.99%. The variation in the degrees of training loss between the fixed compression rate scheme and the adaptive compression scheme under the condition that the model is trained with the early stop strategy is illustrated in Fig. 4.

Figure 4 illustrates that the adaptive compression algorithm smooths the model convergence curve because the model’s training loss decreases rapidly during the initial

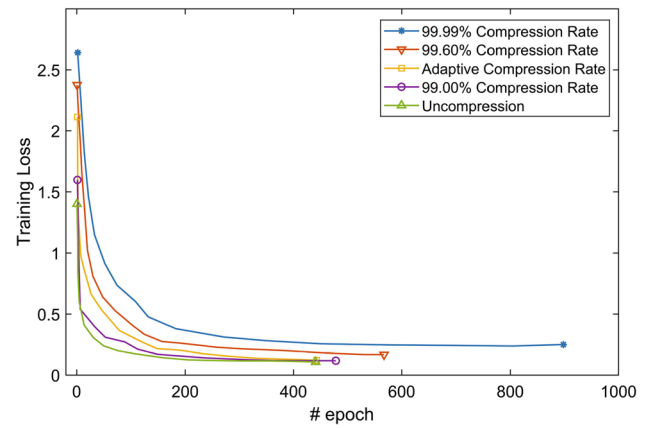


Fig. 4 Change comparison in training loss

training stage, making a relatively large compression rate adoption suitable. When the convergence rate reaches a higher level, the model’s training loss decreases more slowly, prompting an adaptive adjustment of the compression rate to a smaller value. In terms of the iteration number, the convergence of the adaptive compression scheme is comparable to that of the fixed 99.00% compression and uncompression schemes, and it can reach the optimum in fewer iteration cycles. In addition, adaptive compression, 99.00% compression, and uncompressed schemes can converge with better accuracy. Thus, our experiments demonstrate that the adaptive compression algorithm performs better in terms of model convergence and iteration number.

Table 5 intuitively compares the performance of the fixed compression rate and adaptive compression algorithms in terms of accuracy and total time. The total training time of the model is composed of the accumulated time across iterative rounds. In terms of accuracy, the adaptive compression algorithm maintains the model’s accuracy at a high level (99.07%), which is close to the uncompressed accuracy level. In terms of total training time, the adaptive compression algorithm performs better than other control groups, significantly improving training efficiency. In particular, the total time of the adaptive compression algorithm is reduced by approximately 91% compared to that of the uncompressed algorithm. Therefore, our experiments have proven that the adaptive gradient compression algorithm ensures model accuracy and can converge to the optimum in the shortest time, thus improving the speed of model training.

Table 4 Configuration details of cloud server and client modules

Component	Cloud sever	Client
Hardware specifications		
CPU	Intel Xeon E5-2620	Intel Core i7-8700K
GPU	NVIDIA Tesla V100	NVIDIA GeForce GTX 1080 Ti
RAM	64 GB DDR4	32 GB DDR4
Software versions		
Operating system	Ubuntu Server 18.04	Windows 10
Deep learning framework	PyTorch 1.8.1	PyTorch 1.8.1
Network configuration		
Network Bandwidth	10Gbps	1Gbps
Latency	2 ms	5 ms

Table 5 Comparison of accuracy and total time

Compression rate	0.00%	99.00%	99.60%	99.99%	Ada
Accuracy rate	99.10%	99.02%	98.67%	97.26%	99.07%
Total time (s)	26652	5267	2807	4610	2218

### 6.2.2 Impact of local iteration number $\lambda$ and privacy budget $\epsilon$

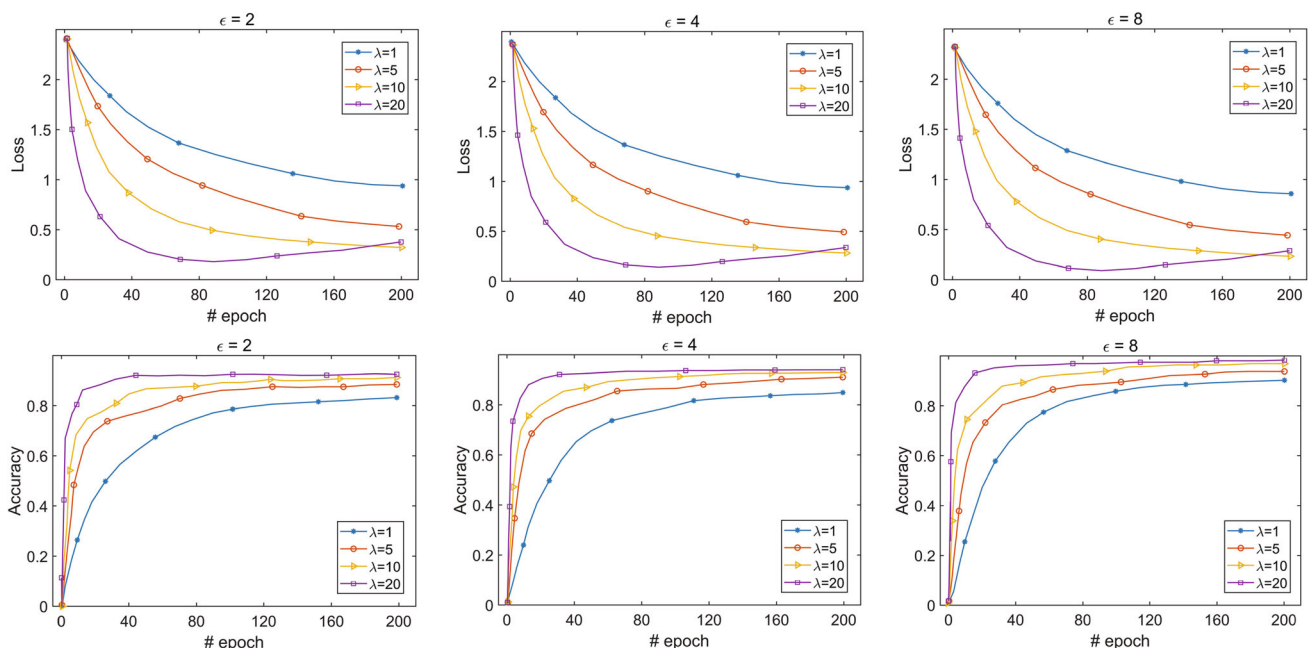
This part shows the degree of the FLCP's convergence regarding communication round number  $T$  under different settings of privacy budget  $\epsilon$  and local iteration number  $\lambda$ . As shown in Figs. 5 and 6, we conduct experiments with different numbers of privacy budgets and local iterations on the MNIST and CIFAR-10 datasets, respectively, to evaluate the convergence of the FLCP. Specifically, we demonstrate the testing accuracy and the training loss concerning communication round number  $T$  when  $\epsilon = \{2, 4, 8\}$ . For each case, we set 4 diverse values of the local iteration, i.e.,  $\lambda = \{1, 5, 10, 20\}$ .

For the MNIST dataset experiments, the testing accuracy and training loss usually first sharply then slowly change. As the privacy budget  $\epsilon$  increases, the size of the testing accuracy curve increases, and the training loss curve of the CNN converges to a lower bound, indicating that the size of the privacy budget  $\epsilon$  has some influence on testing accuracy. This finding corresponds to the FLCP's convergence, where a higher privacy budget  $\epsilon$  produces a smaller convergence error but provides a lower privacy security level. For specific privacy budget settings, the initial training loss decreases significantly with the increasing local iteration  $\lambda$  and eventually reaches a lower stationary point, which shows that a larger  $\lambda$  value implies a minor convergence error. When  $\lambda = 20$ , we observe that the training loss decreases to the lowest value after utilizing approximately 70 iterations, and then it grows with an increasing number of iterations. The reason

for this is that continuous training introduces additional noise to the well-trained model after the loss reaches a stationary point; thus, the size of the training loss curve increases. Similar trends have been observed for the CIFAR-10 dataset experiments. Different privacy budget values  $\epsilon$  have different effects on accuracy. A smaller privacy budget  $\epsilon$  can provide a stronger DP guarantee, but it also results in a loss of precision. However, the FLCP adopts a security aggregation protocol that achieves a high level of accuracy at the same security level. It is noted that a reasonable number of local iterations  $\lambda$  is critical for model training. The local update number  $\lambda$  is too small, and additional rounds of communication are thus needed to aggregate the updates. Conversely, the local update number  $\lambda$  is too large, and it is difficult for the loss function to converge.

### 6.3 Comparison of model accuracy, training time and communication efficiency

This subsection primarily compares the FLCP with the benchmark methods introduced in the experimental setup in three aspects: model accuracy, training time, and communication efficiency. To better evaluate the results, two common baselines are added. The first approach is standalone training, which trains the model only on local datasets without collaborating with other clients and has the most robust degree of privacy preservation. The second approach is centralized dataset training, which ignores privacy issues but has the highest degree of model accuracy.



**Fig. 5** Convergence of the training loss and the model accuracy on MNIST dataset

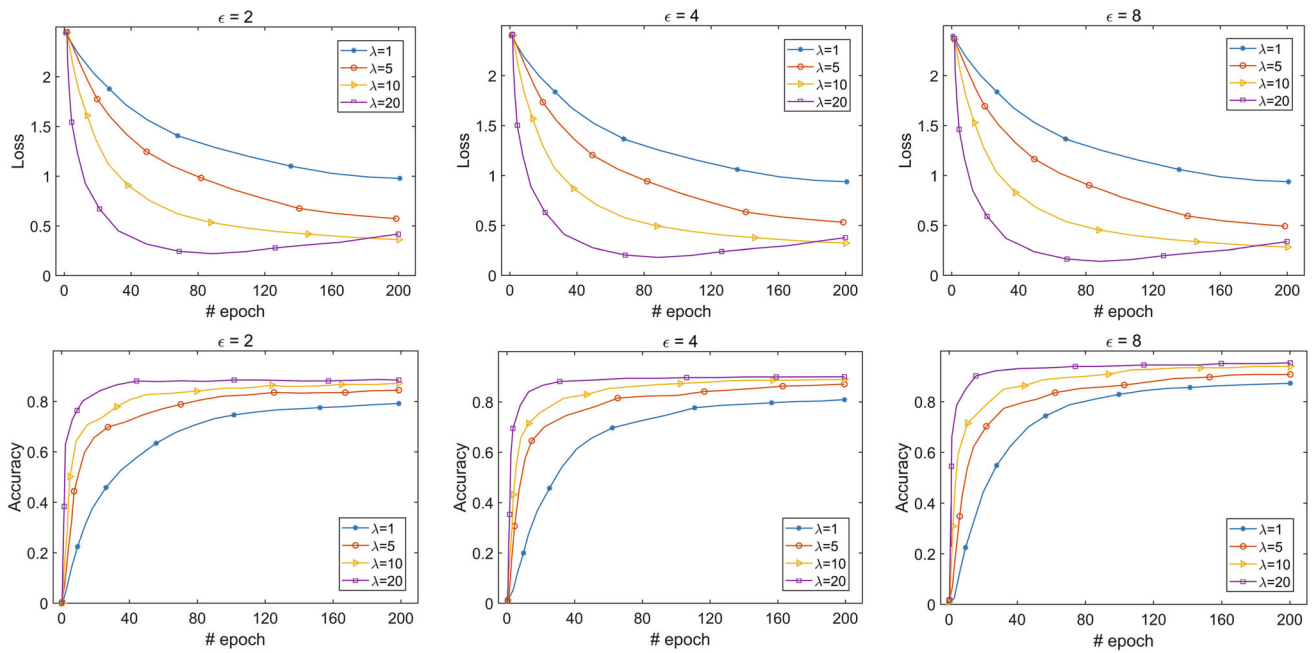


Fig. 6 Convergence of the training loss and the model accuracy on CIFAR-10 dataset

Fig. 7 Comparison of model accuracy on MNIST and CIFAR-10 datasets

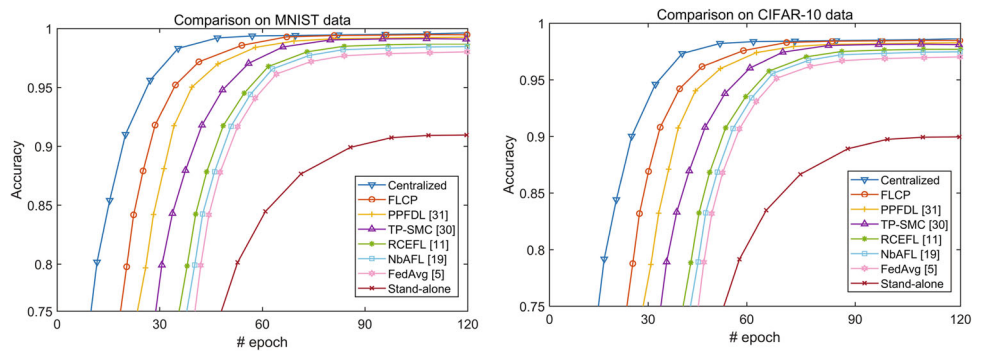
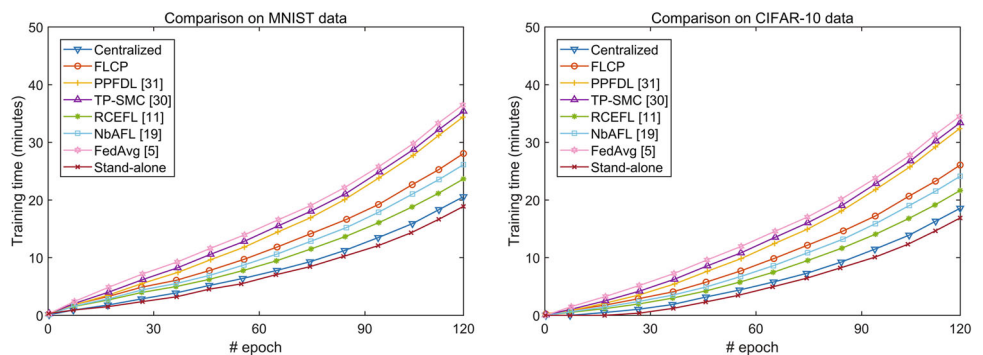


Table 6 Model accuracy with different numbers of clients dropping out

Method	MNIST			CIFAR-10		
	$D = 1$	$D = 3$	$D = 5$	$D = 1$	$D = 3$	$D = 5$
TP-SMC	99.10%	98.34%	97.80%	97.06%	96.32%	95.29%
PPFDL	99.23%	99.12%	98.75%	97.21%	97.08%	96.47%
FLCP	99.30%	99.27%	99.21%	97.36%	97.32%	97.25%

\*  $D$  denotes the number of clients dropping out during training

Fig. 8 Comparison of training time on MNIST and CIFAR-10 datasets





**Table 7** Time of encryption and decryption

Model parameters	36800	52730	87450	120000
Encryption (ms)	46.8	97.5	156.3	221.7
Decryption (ms)	44.3	89.9	143.5	198.2

### 6.3.1 Model accuracy

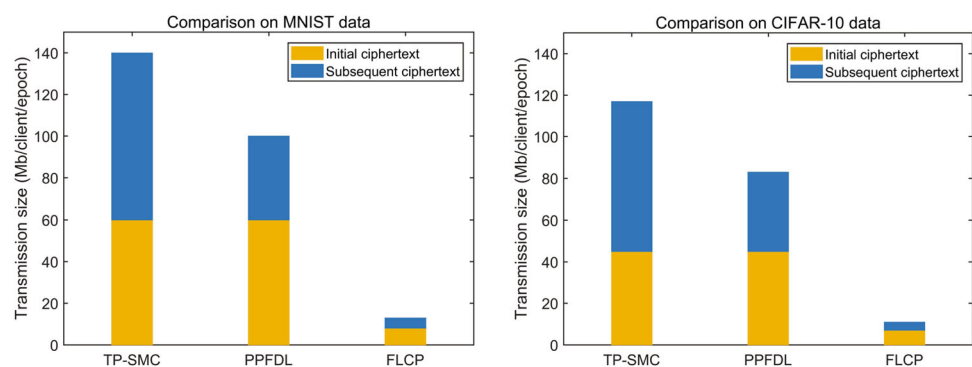
To evaluate model performance, we compare the accuracy of the resulting models of these methods, as shown in Fig. 7. It can be observed that the centralized training method possesses the highest model accuracy (99.4% in MNIST and 97.8% in CIFAR-10), while the stand-alone method has the lowest model accuracy (91.6% in MNIST and 89.3% in CIFAR-10). As a naive FL algorithm, FedAvg's model accuracy is slightly lower than that of other recent FL results. RCEFL adopts the low-precision quantization method to improve communication efficiency, which causes an inevitable loss of precision. In addition, we can observe distinct trends depending on whether a specific privacy-preserving method is employed to safeguard the privacy of the training process and resultant model. Among these trends, the accuracy of methods utilizing encryption-based protocols (such as FLCP, PPFDL, and TP-SMC) is higher than that of those methods merely utilizing the DP protocol (such as NBAFL) because the noise injected by the latter decreases the degree of model accuracy. This finding shows the cost of guarding against the inference risks of the model. Furthermore, to prove that the FLCP is more robust to clients dropping out than the PPFDL and TP-SMC methods, we tested the model accuracy for different numbers of clients dropping out, and the results are shown in Table 6. As the number of clients dropping out increases, the model accuracy of the PPFDL and TP-SMC models decreases significantly, while that of the FLCP method remains almost unaffected. When the number of clients dropping out is  $D = 5$ , the model accuracy of the FLCP, PPFDL, and TP-SMC methods on the MNIST dataset are 99.21%, 98.75%, and 97.80%, respectively, indicating that the FLCP is more robust to clients

dropping out. The FLCP and PPFDL methods have the highest accuracy among all the privacy preservation methods, slightly lower than that of the centralized method, indicating that the FLCP has little effect on model accuracy. The pivotal factor of our approach is its ability to reduce noise by using encryption techniques. Hence, we demonstrate that the approach combining DP protection and HE can achieve a higher degree of model precision while protecting the privacy of inputs and outputs.

### 6.3.2 Training time

These methods are compared and analyzed in terms of training time, as shown in Fig. 8. It is noted that the trend of the training time curves of different methods increases with the epoch. The FedAvg method takes the longest among all algorithms (36.5 minutes on MNIST and 33.7 minutes on CIFAR-10). Moreover, it is observed that the runtime of encryption-based methods such as FLCP, TP-SMC, and PPFDL is longer than that of methods without encryption, such as NBAFL and RCEFL, because the ciphertext operation increases the training. Among these algorithms, the TP-SMC's training time is the longest, since each global update requires additional communication rounds and demands each client's participation in the training. We evaluate the time consumed by the client to perform encryption and decryption. The experimental results are shown in Table 7, demonstrating that the time consumed by ciphertext operations is closely related to the number of updated model parameters. When the number of model parameters is 36,800, the time taken by the client for encryption and decryption is 46.8 milliseconds (ms) and 44.3 ms, respectively. The FLCP adopts a parameter compression algorithm that significantly reduces the number of updated model parameters. Compared to PPFDL and TP-SMC with secure aggregation schemes, the training time of the FLCP (28.4 minutes on MNIST and 25.7 minutes on CIFAR-10) is significantly less than that of PPFDL (34.3 minutes on MNIST and 31.9 minutes on CIFAR-10) and TP-SMC (35.8 minutes on MNIST and 32.6 minutes on CIFAR-10). This finding demonstrates the effectiveness of

**Fig. 9** Total ciphertext size of different transmission approaches in an iteration



our proposed optimization approach concerning model training.

### 6.3.3 Communication efficiency

The experiment in this paper is carried out on a simulated FL framework, and thus, we measure network communication efficiency by comparing the volume of encryption parameters exchanged during transmission. Given that only encryption-based approaches involve transmitting encrypted parameters, we merely compare the FLCP, PPFDL, and TP-SMC. Figure 9 shows the total volume of ciphertext transmitted under various encryption-based methods in a global epoch. The orange bar indicates the initial ciphertext volumes of the model parameters, and the blue bar represents the subsequent ciphertext volumes, containing the multiplied ciphertext, partially decrypted ciphertext, and global parameters. In Fig. 9, it can be seen that compared to PPFDL and TP-SMC, the FLCP decreases the transmitted ciphertext volume of MNIST by 87% and that of CIFAR-10 by 89% on average, because it has a smaller initial ciphertext size than those of the other approaches due to the adaptive compression algorithm implemented in the FLCP. Therefore, our approach plays a significant role in improving communication efficiency.

## 7 Conclusion

This paper proposes a novel approach, namely, the federated learning framework with communication-efficient and privacy-preserving (FLCP), which can reduce communication costs and enhance privacy protection within FL settings. Specifically, we design an adaptive compression algorithm to improve communication efficiency and combine HE and DP to prevent privacy leakage. Compared to current compression methods, the FLCP adaptively compresses uploaded parameters based on the actual data distribution characteristics of the client, reducing communication overhead and ensuring superior model convergence. In addition, our proposed FLCP adopts a lightweight HE combined with DP to significantly mitigate security threats to local clients, providing a higher level of privacy security during the training process, even if the adversary colludes with the client. The experimental results on the MNIST and CIFAR-10 datasets indicate that the proposed FLCP is effective in terms of model accuracy and training efficiency. In the following studies, we will examine the performance of the FLCP in complex training tasks such as multitask learning for high-dimensional datasets.

**Acknowledgements** The work is supported by the grant of National Natural Science Foundation of China (No. 62174134), Shaanxi innovation capability support project (No. 2021TD-25) and Natural Science Basic Research Program of Shaanxi (No. 2021JQ-478).

**Data Availability** The raw or processed data required to reproduce these findings cannot be shared at this time as the data also forms part of an ongoing study.

## Declarations

**Financial Interests** We declare that we have no financial and personal relationships with other people or organizations that can inappropriately influence our work, there is no professional or other personal interest of any nature or kind in any product, service and/or company that could be construed as influencing the position presented in, or the review of, the manuscript.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

- Chen H, Zhang Z, Guan C, Gao H (2020) Optimization of sizing and frequency control in battery/supercapacitor hybrid energy storage system for fuel cell ship. *Energy* 197:117,285. <https://doi.org/10.1016/j.energy.2020.117285>
- Zeng Q, Lv Z, Li C, Shi Y, Lin Z, Liu C, Song G (2022) Fedprols: federated learning for iot perception data prediction. *Appl Intell*, pp 1–13. <https://doi.org/10.1007/s10489-022-03578-1>
- Yang W, Xiang W, Yang Y, Cheng P (2022) Optimizing federated learning with deep reinforcement learning for digital twin empowered industrial iot. *IEEE Trans Industr Inf* 19(2):1884–1893. <https://doi.org/10.1109/TII.2022.3183465>
- Dayan I, Roth HR, Zhong A, Harouni A, Gentili A, Abidin AZ, Liu A, Costa AB, Wood BJ, Tsai CS et al (2021) Federated learning for predicting clinical outcomes in patients with covid-19. *Nat Med* 27(10):1735–1743. <https://doi.org/10.1038/s41591-021-01506-3>
- McMahan B, Moore E, Ramage D, Hampson S, Arcas BA (2017) Communication-efficient learning of deep networks from decentralized data. In: *Artificial intelligence and statistics*. PMLR, pp 1273–1282. <https://proceedings.mlr.press/v54/mcmahan17a.html>
- Xiong B, Yang X, Qi F, Xu C (2022) A unified framework for multimodal federated learning. *Neurocomputing* 480:110–118. <https://doi.org/10.1016/j.neucom.2022.01.063>
- Nguyen DC, Ding M, Pham QV, Pathirana PN, Le LB, Seneviratne A, Li (2021) Federated learning meets blockchain in edge computing: opportunities and challenges. *IEEE Internet Things J* 8(16):12,806–12,825. <https://doi.org/10.1109/JIOT.2021.3072611>
- Kairouz P, McMahan HB, Avent B, Bellet A, Bennis M, Bhagoji AN, Bonawitz K, Charles Z, Cormode G, Cummings R et al (2021) Advances and open problems in federated learning. *Found Trends Mach Learn* 14(1–2):1–210. <https://doi.org/10.1561/22000000083>
- Sun H, Li S, Yu FR, Qi Q, Wang J, Liao J (2020) Toward communication-efficient federated learning in the internet of things

- with edge computing. *IEEE Internet Things J* 7(11):11,053–11,067. <https://doi.org/10.1109/JIOT.2020.2994596>
10. Wu C, Wu F, Lyu L, Huang Y, Xie X (2022) Communication-efficient federated learning via knowledge distillation. *Nat Commun* 13(1):2032. <https://doi.org/10.1038/s41467-022-29763-x>
  11. Sattler F, Wiedemann S, Müller KR, Samek W (2020) Robust and communication-efficient federated learning from non-iid data. *IEEE Trans Neural Netw Learn Syst* 31(9):3400–3413. <https://doi.org/10.1109/TNNLS.2019.2944481>
  12. Reiszadeh A, Mokhtari A, Hassani H, Jadbabaie A, Pedarsani R (2020) Fedpaq: a communication-efficient federated learning method with periodic averaging and quantization. In: *Proceedings of the 23th international conference on artificial intelligence and statistics*. PMLR, pp 2021–2031. <https://proceedings.mlr.press/v108/reiszadeh20a.html>
  13. Hao M, Li H, Luo X, Xu G, Yang H, Liu S (2020) Efficient and privacy-enhanced federated learning for industrial artificial intelligence. *IEEE Trans Industr Inf* 16(10):6532–6542. <https://doi.org/10.1109/TII.2019.2945367>
  14. Melis L, Song C, De Cristofaro E, Shmatikov V (2019) Exploiting unintended feature leakage in collaborative learning. In: *2019 IEEE symposium on security and privacy (SP)*. IEEE, pp 691–706. <https://doi.org/10.1109/SP.2019.00029>
  15. Li T, Sahu AK, Talwalkar A, Smith V (2020) Federated learning: challenges, methods, and future directions. *IEEE Signal Process Mag* 37(3):50–60. <https://doi.org/10.1109/MSP.2020.2975749>
  16. Wu X, Zhang Y, Shi M, Li P, Li R, Xiong NN (2022) An adaptive federated learning scheme with differential privacy preserving. *Futur Gener Comput Syst* 127:362–372. <https://doi.org/10.1016/j.future.2021.09.015>
  17. Li C, Li G, Varshney PK (2021) Communication-efficient federated learning based on compressed sensing. *IEEE Internet Things J* 8(20):15,531–15,541. <https://doi.org/10.1109/JIOT.2021.3073112>
  18. Xu Y, Liao Y, Xu H, Ma Z, Wang L, Liu J (2022) Adaptive control of local updating and model compression for efficient federated learning. *IEEE Trans Mob Comput* 22(10):5675–5689. <https://doi.org/10.1109/TMC.2022.3186936>
  19. Wei K, Li J, Ding M, Ma C, Yang HH, Farokhi F, Jin S, Quek TQ, Poor HV (2020) Federated learning with differential privacy: algorithms and performance analysis. *IEEE Trans Inf Forensics Secur* 15:3454–3469. <https://doi.org/10.1109/TIFS.2020.2988575>
  20. Huang Z, Hu R, Guo Y, Chan-Tin E, Gong Y (2019) DP-ADMM: ADMM-based distributed learning with differential privacy. *IEEE Trans Inf Forensics Secur* 15:1002–1012. <https://doi.org/10.1109/TIFS.2019.2931068>
  21. Li D, Liao X, Xiang T, Wu J, Le J (2020) Privacy-preserving self-serviced medical diagnosis scheme based on secure multi-party computation. *Computers & Security* 90:101,701. <https://doi.org/10.1016/j.cose.2019.101701>
  22. Aono Y, Hayashi T, Wang L et al (2018) Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Trans Inf Forensics Secur* 13(5):1333–1345. <https://doi.org/10.1109/TIFS.2017.2787987>
  23. Zhu H, Wang R, Jin Y, Liang K, Ning J (2021) Distributed additive encryption and quantization for privacy preserving federated deep learning. *Neurocomputing* 463:309–327. <https://doi.org/10.1016/j.neucom.2021.08.062>
  24. Alistarh D, Hoefler T, Johansson M, Konstantinov N, Khirirat S, Renggli C (2018) The convergence of sparsified gradient methods. *Adv Neural Inf Process Syst* 31:5973–5983. [http://amazon.jobs-public-documents.s3.amazonaws.com/strom\\_interspeech2015.pdf](http://amazon.jobs-public-documents.s3.amazonaws.com/strom_interspeech2015.pdf)
  25. Fang C, Guo Y, Hu Y, Ma B, Feng L, Yin A (2021) Privacy-preserving and communication-efficient federated learning in internet of things. *Computers & Security* 103:102,199. <https://doi.org/10.1016/j.cose.2021.102199>
  26. Strom N (2015) Scalable distributed DNN training using commodity GPU cloud computing. In: *Sixteenth annual conference of the international speech communication association*, pp 1488–1492. [http://amazon.jobs-public-documents.s3.amazonaws.com/strom\\_interspeech2015.pdf](http://amazon.jobs-public-documents.s3.amazonaws.com/strom_interspeech2015.pdf)
  27. Luo P, Yu FR, Chen J, Li J, Leung VC (2021) A novel adaptive gradient compression scheme: reducing the communication overhead for distributed deep learning in the internet of things. *IEEE Internet Things J* 8(14):11,476–11,486. <https://doi.org/10.1109/JIOT.2021.3051611>
  28. Nori MK, Yun S, Kim IM (2021) Fast federated learning by balancing communication trade-offs. *IEEE Trans Commun* 69(8):5168–5182. <https://doi.org/10.1109/TCOMM.2021.3083316>
  29. Mao Y, Zhao Z, Yan G, Liu Y, Lan T, Song L, Ding W (2022) Communication-efficient federated learning with adaptive quantization. *ACM Trans Intell Syst Technol (TIST)* 13(4):1–26. <https://doi.org/10.1145/3510587>
  30. Truex S, Baracaldo N, Anwar A, Steinke T, Ludwig H, Zhang R, Zhou Y (2019) A hybrid approach to privacy-preserving federated learning. In: *Proceedings of the 12th ACM workshop on artificial intelligence and security*, pp 1–11. <https://doi.org/10.1145/3338501.3357370>
  31. Xu G, Li H, Zhang Y, Xu S, Ning J, Deng R (2022) Privacy-preserving federated deep learning with irregular users. *IEEE Trans Dependable Secure Comput*, pp 1364–1381. <https://doi.org/10.1109/TDSC.2020.3005909>
  32. Bonawitz K, Ivanov V, Kreuter B, Marcedone A, McMahan HB, Patel S, Ramage D, Segal A, Seth K (2017) Practical secure aggregation for privacy-preserving machine learning. In: *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pp 1175–1191. <https://doi.org/10.1145/3133956.3133982>
  33. Ma J, Naas SA, Sigg S, Lyu X (2022) Privacy-preserving federated learning based on multi-key homomorphic encryption. *Int J Intell Syst* 37(9):5880–5901. <https://doi.org/10.1002/int.22818>
  34. Boulemtafes A, Derhab A, Challal Y (2020) A review of privacy-preserving techniques for deep learning. *Neurocomputing* 384:21–45. <https://doi.org/10.1016/j.neucom.2019.11.041>
  35. Ganesan I, Balasubramanian AAA, Muthusamy R (2018) An efficient implementation of novel paillier encryption with polar encoder for 5g systems in vlsi. *Comput Electr Eng* 65:153–164. <https://doi.org/10.1016/j.compeleceng.2017.04.026>
  36. Wu HT, Ym Cheung, Huang J (2016) Reversible data hiding in paillier cryptosystem. *J Vis Commun Image Represent* 40:765–771. <https://doi.org/10.1016/j.jvcir.2016.08.021>
  37. Dwork C (2008) Differential privacy: a survey of results. In: *Proceedings of the 5th international conference on theory and applications of models of computation*. Springer, pp 1–19. [https://link.springer.com/chapter/10.1007/978-3-540-79228-4\\_1](https://link.springer.com/chapter/10.1007/978-3-540-79228-4_1)
  38. Shokri R, Stronati M, Song C, Shmatikov V (2017) Membership inference attacks against machine learning models. In: *2017 IEEE symposium on security and privacy (SP)*. IEEE, pp 3–18. <https://doi.org/10.1109/SP.2017.41>
  39. Dwork C, McSherry F, Nissim K, Smith A (2006) Calibrating noise to sensitivity in private data analysis. In: *Third theory of cryptography conference*. Springer, pp 265–284. [https://doi.org/10.1007/11681878\\_14](https://doi.org/10.1007/11681878_14)
  40. Goldreich O (2009) *Foundations of cryptography: vol 2, basic applications*. Cambridge university press. <https://www.wisdom.weizmann.ac.il/~oded/PSBookFrag/v2.pdf>
  41. Deng L (2012) The mnist database of handwritten digit images for machine learning research. *IEEE Signal Process Mag* 29(6):141–142. <https://doi.org/10.1109/MSP.2012.2211477>
  42. Abouelnaga Y, Ali OS, Rady H, Moustafa M (2016) Cifar-10: Knn-based ensemble of classifiers. In: *2016 International conference*

on computational science and computational intelligence (CSCI). IEEE, pp 1192–1195. <https://doi.org/10.1109/CSCI.2016.0225>

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Wei Yang** received the B.S. degree in communication engineering from Nanyang Normal University, Nanyang, China, in 2014, and the M.S. degree in signal and information processing from the School of Computer Science, Zhengzhou University of Light Industry, Zhengzhou, China, in 2018. He is currently working toward the Ph.D. degree in electronic science and technology with the School of Automation and Information Engineering, Xi'an University

of Technology, Xi'an, China. His research interests include federated learning, optimization, and wireless networks.



**Yuan Yang** received the B.S. degree in applied electronic technology, and the M.S., and Ph.D. degrees in microelectronics and solid state electronics from the Xi'an University of Technology, Xi'an, China, in 1997, 2000, and 2004, respectively. From 2000 to 2004, she was a Lecturer, and from 2004 to 2009, she was an Assistant Professor. Since 2009, she has been a Professor with the Department of Electronics, Xi'an University of Technology. In 2005,

she was a Visiting Scholar with Kyushu University, Fukuoka, Japan. Her research interests include digital-analog mixed integrated circuit designs, driver and protection technology of power semiconductor device, deep learning algorithm, and hardware acceleration.



**Yingjie Xi** a senior student majoring in Electronic Science and Technology at the International College of Engineering, Xi'an University of Technology. His research interests include circuit systems and wireless communication technology.



**Hailong Zhang** a senior student majoring in Electronic Science and Technology at the International College of Engineering, Xi'an University of Technology. He has a strong interest in digital integrated circuits and telecommunications.



**Wei Xiang** received the B.Eng. and M.Eng. degrees, both in electronic engineering, from the University of Electronic Science and Technology of China, Chengdu, China, in 1997 and 2000, respectively, and the Ph.D. degree in telecommunications engineering from the University of South Australia, Adelaide, Australia, in 2004. He is currently a Cisco Research Chair of AI and IoT and the Director of the Cisco-La Trobe Centre for AI and IoT, La Trobe University, Melbourne, Australia. Previously, he was Foundation Chair and Head of Discipline of IoT Engineering, James Cook University, Cairns, Australia. He has authored or coauthored more than 250 peer-reviewed papers including three books and 200 journal articles. He has served in a large number of international conferences in the capacity of General Co-Chair, TPC Co-Chair, Symposium Chair, etc. His research interest includes the Internet of Things, wireless communications, machine learning for IoT data analytics, and computer vision.

Prof. Xiang, due to his instrumental leadership in establishing Australia's first accredited Internet of Things Engineering degree program, was inducted into Pearcey Foundation's Hall of Fame in October 2018. He is an elected Fellow of the IET in U.K. and Engineers Australia. He was the recipient of the TNQ Innovation Award in 2016, and Pearcey Entrepreneurship Award in 2017, and Engineers Australia Cairns Engineer of the Year in 2017. He was a corecipient of four Best Paper Awards at *WiSATS'2019*, *WCSP'2015*, *IEEE WCNC'2011*, and *ICWMC'2009*. He has been awarded several prestigious fellowship titles. He was named a Queensland International Fellow (2010–2011) by the Queensland Government of Australia, an Endeavour Research Fellow (2012–2013) by the Commonwealth Government of Australia, a Smart Futures Fellow (2012–2015) by the Queensland Government of Australia, and a JSPS Invitational Fellow jointly by the Australian Academy of Science and Japanese Society for Promotion of Science (2014–2015). He was the Vice Chair of the IEEE Northern Australia Section from 2016 to 2020. He was an Editor for IEEE Communications Letters from 2015 to 2017 and is currently an Associate Editor for IEEE Communications Surveys and Tutorials, IEEE Internet of Things Journal, IEEE Access, and Nature journal of Scientific Reports.

Prof. Xiang, due to his instrumental leadership in establishing Australia's first accredited Internet of Things Engineering degree program, was inducted into Pearcey Foundation's Hall of Fame in October 2018. He is an elected Fellow of the IET in U.K. and Engineers Australia. He was the recipient of the TNQ Innovation Award in 2016, and Pearcey Entrepreneurship Award in 2017, and Engineers Australia Cairns Engineer of the Year in 2017. He was a corecipient of four Best Paper Awards at *WiSATS'2019*, *WCSP'2015*, *IEEE WCNC'2011*, and *ICWMC'2009*. He has been awarded several prestigious fellowship titles. He was named a Queensland International Fellow (2010–2011) by the Queensland Government of Australia, an Endeavour Research Fellow (2012–2013) by the Commonwealth Government of Australia, a Smart Futures Fellow (2012–2015) by the Queensland Government of Australia, and a JSPS Invitational Fellow jointly by the Australian Academy of Science and Japanese Society for Promotion of Science (2014–2015). He was the Vice Chair of the IEEE Northern Australia Section from 2016 to 2020. He was an Editor for IEEE Communications Letters from 2015 to 2017 and is currently an Associate Editor for IEEE Communications Surveys and Tutorials, IEEE Internet of Things Journal, IEEE Access, and Nature journal of Scientific Reports.