Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.086022

ARTICLE

# FedE: Protecting Training Data of Federated Learning Based on Multi-Precision Functional Encryption

**Weijia Liu[1], Junwen Deng[2], Hao Li[3] and Zhenyong Zhang[3,*]**

[1]Intelligent and Digital Operation Center, Guizhou Power Grid Co., Ltd., Guiyang, China
[2]Guizhou Power Trading Center, Guiyang, China
[3]State Key Laboratory of Public Big Data, College of Computer Science and Technology, Guizhou University, Guiyang, China
*Corresponding Author: Zhenyong Zhang. Email: zhangzy@gzu.edu.cn

Received: 22 May 2026; Accepted: 27 July 2026; Published: 15 September 2026

**ABSTRACT:** With the rapid development of artificial intelligence technologies in machine learning-as-a-service (MLaaS), deep learning-based intelligent models have demonstrated high value in applications such as trend prediction and risk assessment. However, MLaaS data are typically highly sensitive and contain private information. In cross-institutional collaborative modeling scenarios, different departments and local centers often hold partial, heterogeneous data resources on MLaaS platforms. Given data security, privacy, and compliance requirements, raw data are difficult to share directly, which makes it challenging to apply centralized model training methods. This paper proposes FedE, a multi-precision, multi-source, heterogeneous privacy-preserving federated learning training method based on functional encryption. By introducing a multi-precision joint-computation mechanism, this method enhances numerical adaptation during ciphertext computation. Besides, combined with differential privacy techniques, it prevents model parameter updates from easily compromising privacy in cross-institutional federated learning. Based on the federated learning training process, a secure training framework for sensitive MLaaS data is constructed. Finally, experiments demonstrate the security and efficiency of the proposed approach.

**KEYWORDS:** Privacy protection; functional encryption; federated learning; MLaaS

## 1 Introduction

With the rapid development of information technology, big data and artificial intelligence technologies are continuously merging and integrating [1,2], and the application of deep learning in the machine learning-as-a-service (MLaaS) field is constantly expanding, showcasing broad prospects in areas such as assisted image analysis, infrastructure surveillance, disease diagnosis, and load forecasting [3]. Leveraging its strong ability to extract features and recognize patterns, deep learning can uncover hidden patterns from complex MLaaS data.

As deep learning applications in MLaaS continue to deepen, issues surrounding data privacy and security are becoming increasingly prominent [4]. MLaaS data is typically stored in a fragmented manner across different departments, regional centers, and management platforms. It contains vast amounts of sensitive information, including identification details, personal records, and imaging data, characterized by high levels of privacy and stringent compliance requirements. In the centralized training model, obtaining high-quality models often requires aggregating and processing multi-source MLaaS data in a unified manner. However, this approach can easily lead to risks such as data breaches, unauthorized access, and misuse. It

also struggles to meet the real-world demands for privacy protection and data security [5]. Especially in cross-institutional collaborative modeling scenarios, although various stakeholders possess data resources that offer complementary value, they are often constrained by privacy protections and legal and regulatory restrictions, making it difficult to share raw data directly. Consequently, how to jointly train multi-source MLaaS data while ensuring that the data remains "available but invisible" has become a critical issue that urgently needs to be addressed.

To address these challenges, functional encryption [6] offers a viable technical pathway for privacy-preserving machine learning. This technology allows computing parties to obtain only the computed results of specified functions without directly accessing the original plaintext data. Compared with technologies such as homomorphic encryption, functional encryption is well-suited to handling linear transformations, parameter aggregation, and vector operations during neural network training, making it particularly suitable for secure training tasks in multi-client scenarios. Based on this, the paper focuses on privacy protection needs in federated learning model training, emphasizing the application of inner-product functional encryption for secure training with heterogeneous, multi-source data.

Directly applying inner-product functional encryption to federated learning training still faces several challenges. On the one hand, traditional inner-product functional encryption is primarily suited for integer-domain computations, whereas federated learning model training often involves floating-point numbers, negative values, and different precision parameter representations. This makes it difficult to directly adapt encryption mechanisms to the training tasks [7]. On the other hand, in multi-client collaborative environments, the data sources, feature structures, and numerical precisions of different clients often differ significantly, leading to inconsistencies in representation, precision loss, and excessive computational overhead during unified encrypted computations. Meanwhile, in federated learning scenarios, even when clients do not directly upload raw data, local model parameters, gradient updates, and aggregated results may still reveal sensitive information over multiple rounds of interaction, posing security threats such as membership inference and gradient inversion [8]. Therefore, balancing the adaptability of ciphertext computation, aggregation efficiency, and the strength of privacy protection remains a critical issue that urgently needs to be addressed in current research.

To address the above issues, this paper utilizes inner-product functional encryption to develop a secure federated learning method (i.e., FedE) for multi-precision, multi-source, heterogeneous privacy protection. During training, the proposed approach introduces a unified representation of encoding precision and a multi-precision inner-product encryption mechanism, thereby enhancing numerical adaptability across different data types and parameter representations in ciphertext form. Besides, by incorporating differential privacy principles, the model weights obtained through local client training are perturbed to provide protection. This creates a complete privacy-preserving training pipeline that covers local training, parameter protection, and ciphertext aggregation toward global model updates. This study not only ensures effective model training but also further meets the data security requirements and system implementation needs in multi-client collaborative scenarios. Specifically, the multi-precision joint computation mechanism enhances the numerical representation support provided by functional encryption, improving its applicability in heterogeneous multi-source environments. Furthermore, the coordinated design of differential privacy and functional encryption effectively reduces the risk of model parameter leakage during federated training. Experimental validation across multiple public datasets demonstrates that the proposed method achieves strong model training performance and computational efficiency while maintaining privacy protection.

## 2 Related Work

With the widespread application of MLaaS in medical diagnosis, risk prediction, and decision support, efficiently training models while ensuring data privacy and security has become a critical research issue in privacy-preserving machine learning.

Homomorphic encryption (HE), an important cryptographic technique in privacy computing, enables computation parties to perform addition, multiplication, and other operations on ciphertexts without decrypting the original data. The decrypted result is identical to the result obtained by performing the same operations on the plaintext. This feature enables privacy-preserving computations while safeguarding data confidentiality, and has therefore been widely adopted in scenarios such as privacy-preserving machine learning, data aggregation, and secure collaborative computing [9,10]. Lee et al. [11] implemented ciphertext training and inference for the standard ResNet-20 model on the CIFAR-10 dataset based on the RNS-CKKS (Residue Number System variant of the Cheon-Kim-Kim-Song) fully homomorphic encryption scheme, and employed MINIMAX polynomial approximation for nonlinear activation functions, effectively mitigating precision loss in ciphertext computations. However, this scheme still faces limitations, including high computational overhead and significant system latency. Park et al. [12] proposed a lightweight homomorphic encryption training protocol for support vector machines; by redefining the objective function and introducing approximate methods, they significantly reduced both computational and communication overhead while ensuring data and model privacy. Nevertheless, this approach is primarily suited for classical machine learning models and lacks sufficient support for complex deep learning tasks. Carpov et al. [13] designed a decentralized privacy-preserving logistic regression scheme that leverages the CKKS encryption scheme for multi-party collaborative training, and improved system feasibility and scalability through graph-based computation optimization and batch processing techniques. In particular, in scenarios involving sensitive high-dimensional data, Castro et al. [14] introduced an efficient PPFL-HE method that effectively balances privacy, efficiency, and model utility.

Differential privacy (DP) is a privacy protection framework grounded in rigorous mathematical definitions. By introducing random noise into raw data, model parameters, or statistical results, DP ensures that even if an attacker obtains the system's output, they cannot determine whether a specific individual participated in the training dataset. As a result, differential privacy has garnered widespread attention in multi-client collaborative training scenarios [15,16]. Lian et al. [17] proposed a scheme that achieves synergistic improvements in both communication efficiency and privacy protection by applying local differential privacy perturbations to selected parameters while reducing the overhead associated with client-side data uploads. Hu et al. [18] introduced the Fed-SMP scheme, which sparsifies local model updates and then adds Gaussian noise to mitigate the adverse impact of client-level differential privacy mechanisms on model accuracy. Wei et al. [19] further developed the DP-PFL framework for personalized federated learning, analyzing the relationship between privacy protection strength and model convergence performance by allocating privacy budgets across a two-tiered update process.

Functional encryption (FE) is an important extension of traditional public-key cryptographic systems. Its core idea is to enable authorized users to obtain only the computed results of a specific function on plaintext data, without decrypting the original data, thereby preventing them from learning any other information about it. Consequently, it has significant application value in privacy-preserving computation and constrained data-sharing scenarios [20,21]. In multi-client collaborative training environments, functional encryption enables computation of specific functions while avoiding direct exposure of raw samples, providing crucial cryptographic support for multi-party joint modeling. Qian et al. [22] proposed CryptoFE, a privacy-preserving federated learning scheme based on functional encryption. This scheme demonstrates good computational efficiency in high-precision model parameter aggregation scenarios while also offering

formal guarantees for user gradient privacy; however, its adaptability to complex deep learning training tasks in multi-source, heterogeneous data environments still requires further improvement. Chang et al. [23] introduced a privacy-preserving federated learning framework based on dual-mode decentralized multi-client functional encryption (2DMCFE), which enhanced the security of functional encryption-based federated learning by identifying the risk of hybrid matching attacks in existing multi-input functional encryption (MIFE) schemes and by hiding intermediate models. However, this approach necessitates establishing new encryption instances for different user subsets, making system implementation relatively complex. Zhang et al. [24] proposed PIM-MCFE, a lattice-based functional encryption scheme, and extended it to privacy-preserving federated learning, offering advantages in protecting intermediate aggregated results and resisting quantum attacks. Yu et al. [25] introduced PrivLDFL, a lightweight, dynamic, privacy-preserving federated learning framework that enables dynamic participation and outage tolerance through decentralized multi-client functional encryption, vector compression funnels, and client partitioning strategies.

## 3 Background Knowledge

The method proposed in this paper is primarily based on inner-product functional encryption and federated learning. Below, we will introduce the relevant concepts and fundamental knowledge.

### 3.1 Single-Input Functional Encryption

The description of the Single-Input Functional Encryption (SIFE) scheme is given in Eq. (1), where $|\mathbf{x}| = |\mathbf{y}| = \eta$, and $\mathbf{x}$ and $\mathbf{y}$ are two vectors of length $\eta$.

$$f_{SIFE}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{\eta} (\mathbf{x}_i \mathbf{y}_i) \tag{1}$$

Specifically, SIFE comprises four polynomial algorithms: $\varepsilon_{SIFE} = (\varepsilon_{SIFE}.Setup, \varepsilon_{SIFE}.DKGen, \varepsilon_{SIFE}.Enc, \varepsilon_{SIFE}.Dec)$:

- $\varepsilon_{SIFE}.Setup(1^{\lambda}, \eta) \rightarrow (mpk, msk)$: Given security parameters $\lambda$ and vector length $\eta$, the functional encryption public and private key pairs $(mpk, msk)$ are generated using these parameters. First, the algorithm initializes by generating a multiplicative triple $(\mathbb{G}, p, g)$, where $\mathbb{G}$ is a multiplicative cyclic group of order $p$, with generator $g$. A private key $s = (s_1, s_2, ..., s_\eta) \leftarrow \mathbb{Z}_p^n$ is randomly generated, and the public key $mpk = (g, h_i = g^{s_i})_{i \in [1,...,\eta]} \leftarrow \mathbb{Z}_p^n$ is defined, while the private key is $msk = s$.
- $\varepsilon_{SIFE}.DKGen(msk, \mathbf{y}) \rightarrow dk_{\mathbf{y}}$: Given the private key msk and the computation vector $\mathbf{y}$, the key generation algorithm outputs the functional key $dk_{\mathbf{y}}$. The function that corresponds to the vector $\mathbf{y}$ is defined as $dk_{\mathbf{y}} =< \mathbf{y}, \mathbf{s} >$.
- $\varepsilon_{SIFE}.Enc(mpk, \mathbf{x}) \rightarrow Ct$: Given the public key $mpk$ and the plaintext vector $\mathbf{x}$, the encryption algorithm produces the corresponding ciphertext $Ct$. $Ct = (ct_0, (ct_i)_{i \in [1,...,\eta]})$, where $ct_0 = g^r$ $(r \leftarrow \mathbb{Z}_p)$, and $ct_i = h_i^r g^{\mathbf{x}_i}$.
- $\varepsilon_{SIFE}.Dec(Ct, dk_{\mathbf{y}}) \rightarrow C$: Given the ciphertext $Ct$ and the functional key $dk_{\mathbf{y}}$, the discrete logarithm with base $g$ is computed, and the inner product result is finally calculated. The calculation is performed as follows: $C = \prod_{i \in [1,...,\eta]} ct_i^{\mathbf{y}_i} / ct_o^{dk_{\mathbf{y}}}$. The final inner product result is $f_{SIFE}(\mathbf{x}, \mathbf{y}) = \log_g^C$.

### 3.2 Multi-Input Functional Encryption

Multi-Input Functional Encryption (MIFE) is an extension of SIFE that operates in a computational setting with multiple input vectors. Its description is given in Eq. (2), where, for $n$ clients, $|\mathbf{x}_i| = |\mathbf{y}_i| = \eta_i$, with $i \in [1, \ldots, n]$, $\mathbf{x}_i$ and $\mathbf{y}_i$ being two vectors each of length $\eta_i$.

$$f_{MIFE}((\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n), \mathbf{y}) = \sum_{i=1}^{n} \sum_{j=1}^{\eta_i} (\mathbf{x}_{ij} \mathbf{y}_{\sum_{k=1}^{i-1} \eta_{k+j}}) \tag{2}$$

Specifically, Multi-Input Functional Encryption primarily comprises five polynomial algorithms: $\varepsilon_{MIFE} = (\varepsilon_{MIFE}.Setup, \varepsilon_{MIFE}.SKDist, \varepsilon_{MIFE}.DKGen, \varepsilon_{MIFE}.Enc, \varepsilon_{MIFE}.Dec)$.

- $\varepsilon_{MIFE}$.Setup$(\mathbf{1}^{\lambda}, \eta, n) \rightarrow (mpk, msk)$: Input the security parameter $\lambda$, the vector length $\eta$, and the number of clients $n$, and use these parameters along with the vector length to generate the public and private keys for cryptographic purposes $(mpk, msk)$. First, the algorithm initializes by generating a multiplicative triple $(\mathbb{G}, p, g)$, where $\mathbb{G}$ is a multiplicative cyclic group of order $p$, with generator $g$. A random value $a \leftarrow \mathbb{Z}_p$ is chosen, $\mathbf{a} = (1, a)^T$, $\mathbf{W}_i \leftarrow \mathbb{Z}_p^{2\eta_i}$, $\mathbf{u}_i \leftarrow \mathbb{Z}_p^{\eta_i}$, and then the public key $mpk = (g, g^a, g^{w_a})$ is generated, while the private key $msk = (\mathbf{W}, (\mathbf{u}_i)_{i \in [1,\ldots,n]})$ is created.
- $\varepsilon_{MIFE}$.SKDist($mpk$,$msk$,id$_i$) $\rightarrow sk_i$: Given the public key $mpk$, the private key $msk$, and the user's identifier id$_i$, the partial private key $sk_i$ is output. Here, $sk_i = (g, g^a, (\mathbf{W})_i, \mathbf{u}_i)$.
- $\varepsilon_{MIFE}$.DKGen($mpk$,$msk$,$\mathbf{y}$) $\rightarrow dk_{\mathbf{y}}$: Given the private key $msk$ and the computation vector $\mathbf{y}$, the key generation algorithm derives the functional key $dk_{\mathbf{y}}$. In the first step of the algorithm, the computation vector $\mathbf{y}$ is decomposed into $(\mathbf{y}_1 || \mathbf{y}_2 || \ldots || \mathbf{y}_n)$, where the length of each vector $\mathbf{y}_i$ equals $\eta_i$. The algorithm then generates $dk_{\mathbf{y}} = ((\mathbf{d}_i^T \leftarrow \mathbf{y}_i^T \mathbf{W}_i), z \leftarrow \Sigma \mathbf{y}_i^T \mathbf{u}_i)$.
- $\varepsilon_{MIFE}$.Enc$(sk_i, \mathbf{x}_i) \rightarrow Ct_i$: Given the private key $sk_i$ and the plaintext vector $\mathbf{x}_i$, the encryption algorithm outputs the corresponding ciphertext $Ct_i$. The algorithm first generates a random number $r_i \leftarrow \mathbb{Z}_p$ and computes the ciphertext $Ct_i = (t_i \leftarrow g^{ar_i}, ct_i \leftarrow g^{x_i} g^{u_i} g^{(w_a)_i r_i})$.
- $\varepsilon_{MIFE}$. Dec$(Ct, dk_{\mathbf{y}}) \rightarrow C$: Given the ciphertext $Ct$ and the functional key $dk_{\mathbf{y}}$, the algorithm solves the discrete logarithm problem with base $g$, ultimately computing the inner product result. Compute $C = \frac{\prod_{i \in [1,\ldots,n]} ([\mathbf{y}_i^T ct_i] / (d_i^T t_i))}{z}$. The final inner product result is $f_{MIFE}((\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n), \mathbf{y}) = log_g^C$.

### *3.3 Detailed Specification of the Differential Privacy Component*

**Privacy Mechanism and Noise Distribution.** We will formally define the Gaussian mechanism as follows:

**Definition 1.** *Let $f : \mathcal{D} \rightarrow \mathbb{R}^d$ be a deterministic function. The Gaussian mechanism $\mathcal{M}$ adds noise sampled from a Gaussian distribution to the output of $f$: $\mathcal{M}(D) = f(D) + \mathcal{N}(0, \sigma^2 I_d)$, where $\mathcal{N}(0, \sigma^2 I_d)$ denotes the $d$-dimensional isotropic Gaussian distribution with mean zero and variance $\sigma^2$ per coordinate.*

In FedE, each client applies this mechanism to its local weight vector $\mathbf{w}_i$ before encryption: $\mathbf{w}_i^{DP} = \mathbf{w}_i + \eta_i, \quad \eta_i \sim \mathcal{N}(0, \sigma^2 I)$, where the noise is independently sampled per client per round to prevent correlation attacks.

**Sensitivity.** We will define the $\ell_2$-sensitivity of the client's local training function.

**Definition 2.** *For a function $f : \mathcal{D} \rightarrow \mathbb{R}^d$, the $\ell_2$-sensitivity is: $\Delta_2 f = \max_{D, D' \text{ neighbouring}} \|f(D) - f(D')\|_2$, where neighboring datasets differ by at most one training sample. In FedE, to bound the sensitivity, we apply gradient clipping before noise addition, ensuring that $\|\mathbf{w}_i\|_2 \leq C$ for a pre-selected clipping bound $C$. Consequently, the sensitivity is bounded by $\Delta_2 = 2C$ under the standard neighboring relation.*

**Privacy Budget and $(\varepsilon, \delta)$ -DP Guarantee.** The DP guarantee of our local perturbation mechanism is:

**Theorem 1.** *Let $\Delta_2$ be the $\ell_2$-sensitivity of the client's weight vector. For a given noise scale $\sigma > 0$, the Gaussian mechanism $\mathcal{M}(D) = f(D) + \mathcal{N}(0, \sigma^2 I)$ satisfies $(\epsilon, \delta)$-differential privacy, where for any $\delta \in (0, 1)$: $\varepsilon = \frac{\sqrt{2 \ln(1.25/\delta)} \cdot \Delta_2}{\sigma}$.*

*This follows directly from the standard Gaussian mechanism theorem. In our implementation, we set* $\delta = 10^{-5}$ *and calibrate* $\sigma$ *according to the clipping bound* $C$.

### *3.4 Federated Learning*

Federated learning is a distributed machine learning paradigm designed to address the challenges of data privacy and security. In a federated learning system, participating nodes follow a predefined collaborative process: they use their local data to train local models and then submit the updated parameters to a central aggregator for integration. The process can be broken down into the following four steps:

(1) Model Initialization: The central aggregator first initializes the model parameters and then distributes these initialized model parameters to the clients participating in federated learning.

(2) Local Training: Each client uses its own local dataset to perform localized training on the global model parameters issued by the central aggregator.

(3) Model Parameter Aggregation: After several rounds of local training, each client uploads its respective model parameters to the central aggregator. Once the aggregator receives parameter updates from all participants, it performs weighted aggregation using a predefined algorithm and optimizes the global model parameters based on the aggregated results. Subsequently, the aggregator redistributes the updated global model parameters to all clients, initiating a new round of collaborative training.

(4) Iterative Optimization: After the global model parameters are distributed to the clients, steps (2) and (3) are repeated until the global model converges.

## 4 FedE: Privacy-Preserving Federated Learning With a Collaboration Framework

In the collaborative training process of cross-regional, multi-source federated learning, the local data held by each participating party constitutes their core assets and sensitive privacy information. To address privacy risks encountered during joint training, this paper designs and implements a privacy-protectionmework (i.e., FedE) tailored for cross-regional, multi-source federated learning.

### *4.1 Privacy-Preservation Collaborative Framework*

This section describes the multi-precision, multi-source, heterogeneous federated learning collaboration training architecture. FedE involves three entities: clients, a central server, and a key generation authority (KGA).

**Key Generation Authority (KGA):** The KGA is a trusted third-party entity that can be regarded as an authoritative key distribution center or any other credible entity. The KGA's primary task is to generate system-wide public parameters and distribute all keys. Specifically, the KGA generates the master public key $mpk$ and the master private key $msk$. It also generates the functional private key $dk_f$ associated with the weight vector provided by the central server.

**Clients:** Clients train models based on their local private datasets and generate initial model parameters. To prevent private information from being leaked during subsequent transmission and aggregation processes, clients first add differential privacy noise to the model parameters obtained from local training. By introducing rigorously defined mathematical perturbations, they ensure that the contribution of individual data points cannot be accurately inferred, thereby providing theoretically provable privacy protection. Subsequently, the data protection encryption (DPE) method is employed to perform structured decomposition and encryption on the noisy parameters, ultimately uploading the resulting ciphertext-formatted parameters to the central server. The server performs aggregation operations on the ciphertexts, completing the update of the global model without decrypting each client's parameters.

**Central Server:** During the initialization phase of collaborative learning, the central server presets global model parameters and is responsible for coordinating model updates and system scheduling throughout the entire federated learning process. Specifically, after receiving encrypted weight parameters uploaded by clients, the central server requests functional decryption keys from the KGA, enabling it to perform secure inner-product computations on ciphertexts and thus obtain aggregated results for the encrypted model weights. Once these aggregated results are verified through decryption, they are distributed as new global model parameters to all participating clients for the next round of local training.

The DPE is described as follows. FedE uses $W_k^i$ to represent the weight vector of client k's local model. Clients must perform the following weight processing and weight encryption steps:

- Data Processing: On client k, the Euclidean division method is employed to perform precision decomposition on each data point $w_{kj}^i$ in the local model's weight vector $W_k^i$. Specifically, this decomposition algorithm divides each $w_{kj}^i$ into t distinct precision sub-datasets $W_{kl}^i$ based on precision requirements, thereby forming a multi-precision data representation.
- Data Encryption: During the data encryption phase, the client first submits an encryption key request to the KGA. After identity verification, it obtains the corresponding functional encryption key. Subsequently, the system employs a hierarchical encryption mechanism to encrypt the weights obtained through Euclidean precision decomposition. The t-weight sub-datasets $W_{k,l}^i$ are each encrypted separately, yielding the corresponding ciphertexts $ct_{k,l}^i$. Ultimately, all ciphertexts together form the complete set of encrypted weight vectors $CTs = \{ct_{k,1}^i, ct_{k,2}^i, ..., ct_{k,t}^i\}$.

### *4.2 Security Threats*

We assume that the central server is an honest yet curious entity, ensuring its behavior adheres to the protocol specifications and does not actively deviate or engage in malicious tampering. It may attempt to infer the private data of other participants by collecting and analyzing information exchanged during protocol execution (such as ciphertexts, intermediate computation results, or other metadata). To ensure the security and functional correctness of the cryptographic system, the KGA is introduced as the system's core trust authority. The KGA is responsible for generating system parameters, producing public and private keys, and securely distributing them. Moreover, it serves as a completely trusted third party. This institution does not participate in the actual computation or transmission of data, but the key materials it generates provide the cryptographic foundation for the entire privacy-preserving computation. The KGA's complete trustworthiness means it will not disclose any key information nor collude with either the server or clients. This hypothesis aligns with the trust models used in most inner-product cryptography and advanced cryptographic protocols. Based on the above, consider the following threat model:

- Honest Yet Curious Central Server: The central server strictly adheres to the Federated Learning protocol specifications and is capable of honestly and correctly performing aggregation operations on the encrypted model weights collected from each client, ensuring computational integrity and result accuracy throughout the aggregation process, and reliably distributing the resulting global aggregated results to the participating parties. However, while its behavior complies with protocol requirements and does not actively deviate from established procedures or introduce malicious actions, it does record and store all weight parameters uploaded by clients (or their encrypted forms), and may, based on historical interaction data, attempt to conduct inference and analysis in order to deduce private information contained within a specific client's local training data.
- Malicious Attacker: A malicious attacker possesses the ability to eavesdrop on various communication channels and may also launch attacks against the server to steal sensitive data.

### *4.3 Inner-Product Functional Encryption Method Based on Multi-Precision Bit Decomposition*

To address the high decryption overhead of inner-product functional encryption, this paper proposes a scheme that leverages multi-precision bit decomposition.

#### *4.3.1 Double-Precision Bit Decomposition Inner Product Functional Encryption*

**Definition 3.** *(Vector Euclidean Division) Let $\boldsymbol{z} \in \boldsymbol{Z}^n$ be a vector. Through Euclidean division, $\boldsymbol{z}$ can be expressed as: $\boldsymbol{Z} = \frac{\boldsymbol{Z} \oplus \check{\boldsymbol{Z}}_{p_{div}}}{p_{div}} \cdot p_{div} \oplus \check{\boldsymbol{z}}_{p_{div}}$ Here, $\oplus$ and $\ominus$ denote element-wise addition and subtraction, respectively.*

According to Definition 3, the vector $\mathbf{z}$ is decomposed into: $DCP_{p_{div}}(\mathbf{z}) = (\hat{\mathbf{z}}, \check{\mathbf{z}}) \in \mathbf{Z}^{2\times n}$. Conversely, the restructured representation of $(\hat{\mathbf{z}}, \bar{\mathbf{z}}) \in \mathbb{Z}^{2\times n}$ is given by: $RCB_{p_{div}}(\hat{\mathbf{z}}, \check{\mathbf{z}}) = \hat{\mathbf{z}} \cdot p_{div} \oplus \check{\mathbf{z}} \in \mathbf{Z}^n$. Furthermore, for $\mathbf{z} \in \mathbf{Z}^n$, it is possible to extend this approach to high-precision vector decomposition, for example: $DCP^t_{p_{div}}(\mathbf{z}) = (\hat{\mathbf{z}}_{t-1}, \ldots, \hat{\mathbf{z}}_0) \in \mathbf{Z}^{t\times n}$. And we also performs high-precision decomposition and recombination on the set of vectors $\{\hat{\mathbf{z}}_{t-1} \in \mathbf{Z}^n, \ldots, \check{\mathbf{z}}_0\}$: $RCB^t_{p_{div}}(\hat{\mathbf{z}}_{t-1}, \ldots, \hat{\mathbf{z}}_0) = \hat{\mathbf{z}}_{t-1} \cdot p^{t-1}_{div} \oplus \ldots \oplus \hat{\mathbf{z}}_0 \cdot p^0_{div}$

**Definition 4.** *(Double-Tuple Ciphertext Based on Euclidean Division) Let $ct = (\widehat{ct}_1, \widehat{ct}_0) \in \mathbb{Z}_q^{2\times n}$ be a double-tuple ciphertext, the ciphertext $ct$ is defined as follows: $\widehat{ct}_j \leftarrow [\widehat{z}_{i,j}]$, $j \in \{0,1\}$, where $\boldsymbol{z}_i = \hat{\boldsymbol{z}}_{i,1} \cdot p^1_{div} \oplus \hat{\boldsymbol{z}}_{i,0} \cdot p^0_{div}$, and $[x]$ denotes the ciphertext form of the vector $x$.*

**Lemma 1.** *(Double-Precision Inner Product Functional Encryption) Let $\boldsymbol{ct} = (\hat{\boldsymbol{z}}_{1,1}, \tilde{\boldsymbol{z}}_{1,0}) \in \mathbb{Z}_q^{2\times n}$ be a double-tuple ciphertext, with a functional private key $dk = (\widehat{\boldsymbol{y}}_2, \tilde{\boldsymbol{y}}_2) = (<\widehat{\boldsymbol{z}}_2, \boldsymbol{s}>, (\tilde{\boldsymbol{z}}_2, \boldsymbol{s}) >\in \mathbb{Z}_q^2$. $RCB^3_{p_{div}}(b_2, b_1, b_0) =< \boldsymbol{Z}_1, \boldsymbol{Z}_2 >$, where $b_2 = Dec([\hat{\boldsymbol{z}}_1], \hat{\boldsymbol{y}}_2)$, $b_1 = Dec([\hat{\boldsymbol{z}}_1], \hat{\boldsymbol{y}}_2) \oplus Dec([\check{\boldsymbol{z}}_1], \hat{\boldsymbol{y}}_2)$, and $b_0 = Dec([\check{\boldsymbol{z}}_1], \check{\boldsymbol{y}}_2)$. Here, $Dec$ represents the decryption result when $\eta = 1$, i.e., $\varepsilon_{SIFE}.Dec$, or when $\sum \eta_i = 1$ and $n = 1$, i.e., $\varepsilon_{MIFE}.Dec$.*

**Proof:** It is clear that $b_2 =< \hat{\mathbf{z}}_1 \cdot \hat{\mathbf{z}}_2 >$, $b_1 =< \hat{\mathbf{z}}_1 \cdot \check{\mathbf{z}}_2 > + < \check{\mathbf{z}}_1 \cdot \hat{\mathbf{z}}_2 >$, and $b_0 =< \check{\mathbf{z}}_1 \cdot \check{\mathbf{z}}_2 >$. Therefore:

$$\begin{aligned} &RCB^3_{p_{div}}(b_2, b_1, b_0) = b_2 \cdot p^2_{div} \oplus b_1 \cdot p^1_{div} \oplus b_0 \cdot p^0_{div} \\ =&< \hat{\mathbf{z}}_1 \cdot \hat{\mathbf{z}}_2 > \cdot p^2_{div} \oplus < \check{\mathbf{z}}_1 \cdot \check{\mathbf{z}}_2 > \cdot p^0_{div} \oplus (< \hat{\mathbf{z}}_1 \cdot \check{\mathbf{z}}_2 > + < \check{\mathbf{z}}_1 \cdot \hat{\mathbf{z}}_2) \cdot p^1_{div} \end{aligned}$$

Thus, we obtain $RCB^3_{p_{div}}(b_2, b_1, b_0) =< \mathbf{z}_1, \mathbf{z}_2 >$. □

#### *4.3.2 Multi-Precision Bit Decomposition Inner Product Functional Encryption*

**Definition 5.** *(T-Tuple Ciphertext Based on Euclidean Division) Let $ct = (\widehat{ct}_{t-1}, \ldots, \widehat{ct}_0) \in \mathbb{Z}_q^{t\times n}$ be a t-tuple ciphertext. The Euclidean ciphertext $ct$ is defined as follows: $\hat{ct}_j \leftarrow [\hat{z}_{i,j}]$, $j \in \{0, \ldots, t-1\}$. Among them, $\boldsymbol{z}_i = \hat{\boldsymbol{z}}_{i,t-1} \cdot p^{t-1}_{div} \oplus \ldots \oplus \hat{\boldsymbol{z}}_{i,0} \cdot p^0_{div}$.*

**Lemma 2.** *(Multi-precision Inner Product Functional Encryption) – Let $ct = (\hat{z}_{1,t-1}, \ldots, \hat{z}_{1,0}) \in Z_q^{t\times\eta}$ be a t-tuple ciphertext, and let the functional private key $dk = (\hat{y}_{2,t-1}, \ldots, \hat{y}_{2,0}) = (< (\hat{z}_{2,t-1}, s), (\hat{z}_{2,0}, s) >\in Z_q^t)$. $RCB^{2t-1}_{p_{div}}(b_{2t-2}, \ldots, b_0) =< \mathbf{Z}_1, \mathbf{Z}_2 >$ Where $b_i = \sum_{k+l=i} Dec([\hat{z}_{1,k}], \hat{y}_{2,l})$, with Dec representing the decryption result of either $\varepsilon_{SIFE}.Dec$ or $\varepsilon_{MIFE}.Dec$.*

**Proof:** It is clear that $b_i = \sum_{k+l=i} Dec([\hat{z}_{1,k}], \hat{y}_{2,l})$. From this, we can conclude:

$$RCB^{2t-1}_{p_{div}}(b_{2t-2}, \ldots, b_1, b_0) = \sum_{0\le i<2t-1} p^i_{div} \cdot b_i = \sum_{0\le i<2t-1} p^i_{div} \sum_{k+l=i} < \hat{z}_{1,k}, \hat{z}_{2,l} > .$$

Therefore, by similar reasoning, we can derive $RCB^{2t-1}_{p_{div}}(b_{2t-2}, \ldots, b_1, b_0) =< \mathbf{z}_1, \mathbf{z}_2 >$. □

### *4.4 Design Details of FedE*

#### *4.4.1 Local Privacy Enhancement Training Based on Differential Privacy*

First, the system is initialized as shown in Algorithm 1, where a cryptographic system is constructed via a key-generation mechanism. Algorithms $\varepsilon_{MIFE}.Setup(1^{\lambda}, \eta, n)$ and $\varepsilon_{MIFE}.SKDist(mlpk, msk, C_i)$ are executed to generate system parameters, and encryption keys are distributed to each client $C_i$. Simultaneously, the parameters required for the central server's functional private key generation stage are prepared in advance, and the cryptographic book necessary for inner product function decryption is constructed. In this phase, KGA generates a corresponding number of keys to support the addition of more participants, and no modifications are required for participants throughout the process.

**Algorithm 1:** System initialization

**Require:** Security parameter $\lambda$, parameter $\eta$, parameter $n$, set of functions $C$
**Ensure:** Master public key $mpk$, master secret key $msk$, distribution of functional secret keys to clients
1: $(mpk, msk) \leftarrow \varepsilon_{\text{MIFE}}.\text{Setup}(1^{\lambda}, \eta, n)$
2: **for** each $C_i \in C$ **do**
3: $sk_i \leftarrow \varepsilon_{\text{MIFE}}.\text{SKDist}(mpk, msk, C_i)$
4: Send $sk_i$ to $C_i$ via a secure authenticated channel
5: **end for**
6: KGA securely stores msk and distributes mpk to all parties
7: **return** $mpk, msk$

Before performing local model parameter encryption and uploading them to the central server for subsequent secure aggregation, client $C_i$ first obtains the locally deployed federated learning model and sets relevant parameters, such as differential privacy settings. As shown in Algorithm 2, before performing local model parameter encryption and uploading them to the central server to participate in subsequent secure aggregation, client $C_i$ first completes model training on the local dataset according to the federated learning algorithm and generates the corresponding model weight parameters. Subsequently, to meet privacy protection requirements, the client uses the differential privacy algorithm (i.e., $DP(\cdot)$) to process the model parameters obtained from local training, applying noise perturbations with a given privacy strength $\sigma$. In particular, the local model update is first clipped with clipping bound $C$, ensuring $\|\mathbf{w}_i\|_2 \leq C$, from which the sensitivity is bounded by $\Delta_2 = 2C$ under the standard neighboring dataset definition. This establishes a clear relationship between the clipping bound and the sensitivity used in the Gaussian mechanism. The Gaussian noise variance is determined according to the standard Gaussian mechanism: $\mathcal{M}(D) = f(D) + \mathcal{N}(0, \sigma^2 I)$, where the privacy guarantee satisfies $\epsilon = \frac{\sqrt{2\ln(1.25/\delta)}\,\Delta_2}{\sigma}$. The noise scale $\sigma$ is calibrated according to the clipping bound $C$ and the target privacy parameters $(\varepsilon, \delta)$.

After completing the perturbation, the client sets the appropriate type-identification parameters based on the data type and performs unified-precision encoding of the noisy model weights, thereby mapping parameters in different precision formats into a unified numerical space. Following the parameter decomposition method, multiple-precision parameters $t$ and $p_{div}$ are selected to represent the encoded results in a vectorized manner, ensuring they meet the input requirements for subsequent inner-product functional encryption operations. Finally, the client encrypts the decomposed weight parameters using the functional private key $sk_i$ and sends the resulting ciphertext, along with the data type parameters, to the central server.

**Algorithm 2:** Client-side local training

**function** CLINET-TRAIN($\sigma, sk_i, C_i, \mathcal{L}_{FL}, type$)
  $w_i \leftarrow \mathcal{L}_{FL}(C_i)$
  $w_i^{DP} \leftarrow DP(w_i, \sigma)$
  $w_i^{Scaled} \leftarrow round(w_i^{DP} \cdot p_{div})$
  $(\hat{w}_{t-1}, \ldots, \hat{w}_0) \in W^{t\times n} \leftarrow DCP_{pdiv}^{t}(w_i^{Scaled})$
  **for** $k = 0$ to $t-1$ **do**
    $ct_{i,k} \leftarrow \varepsilon_{MIFE}.Enc(sk_i, \hat{w}_k)$
  **end for**
  $ct_i^{t\times n} \leftarrow (ct_{i,t-1}, ..., ct_{i,0})$
  **return** send $ct_i^{t\times n}$ along with $type$ to the central server
**end function**

Inner-product encryption algorithms typically support only operations over finite integer domains; however, in practice, sample data and model parameters often contain both negative numbers and floating-point values. This makes it difficult to deploy this encryption scheme directly. To address the aforementioned issue, this chapter extends the application scenarios of inner-product encryption in vector computation by employing a generator inversion strategy, thereby enabling support for vector operations that include negative elements. To address the challenges posed by floating-point numbers, this chapter performs pre-processing by scaling the data and rounding it down, as follows:

$$\tilde{w} \leftarrow \left[\tilde{w} \cdot 2^p\right], \overline{w}_1 + \overline{w}_2 \leftarrow \left[\widetilde{w_1} + \widetilde{w_2}\right], \widehat{w_1 \cdot w_2} \leftarrow \left[\frac{\widetilde{w_1} \cdot \widetilde{w_2}}{2^p}\right]. \tag{3}$$

In this process, the scaling factor $p$ is a predefined parameter primarily used to control computational precision. In practical deployment scenarios, this coefficient can be adjusted to achieve the desired prediction accuracy, thereby reducing the impact of floating-point preprocessing on the final prediction performance. The reason why the multiplication of $\overline{w}_1$ and $\overline{w}_2$ needs to be divided by $2^p$ is that both $\overline{w}_1$ and $\overline{w}_2$ were scaled up by a factor of $2^p$ during the pre-processing stage, resulting in their product being amplified to a total factor of $2^{2p}$. However, to maintain appropriate precision, it suffices to scale the result by a factor of $2^p$.

**Algorithm 3:** Central server secure aggregation

**function** SERVER($\epsilon, sk_i, C_i, \mathcal{L}_{FL}, type$)
  **while** $C_i \in C$ **do**
    Request the client to obtain the corresponding $Ct_q^{t\times n}$ and $type$
  **end while**
  **while** $|Ct_{recv}| \geq t$ **do**
   $Ct_{recv} \leftarrow Ct_q^{t\times n}$
   **if** $type == 1$ **then**
    $w_i \leftarrow \frac{1}{n}$
   **else if** $type == 2$ **then**
    $w_i \leftarrow 1$
   **end if**
   $W_{server} \leftarrow w_i$
  **end while**

(Continued)

**Algorithm 3 (continued)**

**if** $|Ct_{recv}| \geq t$ **then**
  Request KGA to obtain the functional decryption private key $dk_{w_c}$ based on $W_{server}$
  $W_{global} \leftarrow \mathrm{RCB}_{p_{div}}^{2t-1}(b_{2t-2}, \ldots, b_0) = \langle W_{server}, W \rangle$
  $b_i = \sum_{k+l=i} \mathrm{Dec}(Ct_{recv}, dk_{w_c})$
  **return** Distribute the secure aggregated weight parameter $W_{global}$ to the clients
**end if**
**end function**

The DP and FE mechanisms serve complementary roles in our framework. FE provides cryptographic confidentiality: even with full access to ciphertexts and functional decryption keys, the server cannot recover individual client parameters beyond the prescribed inner product. However, FE does not prevent inference attacks that exploit statistical patterns in the aggregated results (e.g., membership inference or property inference). DP addresses this gap by adding calibrated noise to each client's local update, which mathematically bounds the influence of any single training sample on the final aggregated output. The two protections compose additively: FE protects against direct parameter reconstruction, while DP protects against statistical inference. Even if the server were to bypass the FE (e.g., via KGA compromise), DP would still provide a statistical privacy guarantee (though weakened by the lack of FE). In our threat model, both defences are active simultaneously, providing layered security.

#### *4.4.2 Secure Aggregation Based on Multi-Precision Inner-Product Functional Encryption*

During the secure aggregation phase, the central server first sets the minimum aggregation parameters and other relevant information. As shown in Algorithm 3, the central server first requests the locally trained ciphertext weight vector parameters from each client. Next, it determines whether the collected weight vector parameters meet the minimum aggregation criteria; if so, all parameters uploaded by clients are organized and stored in $Ct_{recv}$ for further processing. At the same time, based on the type parameter uploaded by each client, the system determines whether the data is horizontal or vertical. For horizontal data, $w_i$ is set to $\frac{1}{n}$; for vertical data, $w_i$ is set to 1 (this paper assumes that all vertical data can be combined into complete horizontal data), with $|W_{server}| = n$, where $n$ represents the number of clients that have sent responses. After setting the server parameters, the system uses the configured $W_{server}$ to request that KGA generate a multi-precision, inner-product-function-based encryption and functional-decryption private key $dk_{w_c}$.

This process ensures effective protection of data privacy during secure aggregation. Finally, using the multi-precision inner-product-based encryption and recombination method, the global aggregation parameters $W_{global}$ are decrypted and distributed to clients to update the global model, thereby laying the foundation for the next round of model training and secure aggregation.

## 5 Formal Security Proof of the Protocol

### *5.1 Security Model and Adversarial Capabilities*

We formalize the security model under the honest-but-curious setting. Let the adversary $\mathcal{A}$ be a probabilistic polynomial-time (PPT) entity that controls the central server. $\mathcal{A}$ has access to:

- All ciphertexts (CTs) uploaded by clients;
- The functional decryption key $dk_{\mathbf{y}}$ for the aggregation weight vector $\mathbf{y}$;
- All intermediate aggregated results $W_{global}^{(t)}$ over multiple communication rounds.

$\mathcal{A}$ does not have access to the master secret key $msk$, nor does it collude with the Key Generation Authority (KGA). The goal of $\mathcal{A}$ is to infer any additional information about a client's local weight vector $W_i$ beyond the intended aggregated inner product $\langle W_i, \mathbf{y} \rangle$.

### *5.2 IND-CPA Security of the Multi-Precision Inner-Product Functional Encryption*

We prove that our multi-precision FE scheme is selectively Indistinguishability under Chosen-Plaintext Attack (IND-CPA) secure under the standard Decisional Diffie-Hellman (DDH) assumption in the underlying cyclic group $\mathbb{G}$.

**Theorem 2.** *(**Confidentiality of Ciphertexts**). Let $\Pi = (Setup, SKDist, DKGen, Enc, Dec)$ be the multi-precision inner-product functional encryption scheme described in Section 4.3. Under the DDH assumption in $\mathbb{G}$, $\Pi$ achieves selective IND-CPA security against any PPT adversary $\mathcal{A}$.*

**Proof:** We construct a sequence of hybrid games $Game_0$ to $Game_3$:

- $Game_0$: The real selective IND-CPA game, where the adversary commits to two challenge vectors $x^{(0)}$ and $x^{(1)}$ of equal length before receiving the public key.
- $Game_1$: The challenger replaces the group elements $h_i = g^{s_i}$ in the public key with random elements from $\mathbb{G}$. By the DDH assumption, $Game_1$ is computationally indistinguishable from $Game_0$.
- $Game_2$: The challenge ciphertext components $ct_i = h_i^r g^{x_i}$ are replaced with uniformly random group elements. This step relies on the fact that, without knowing $s_i$, the adversary cannot distinguish $g^{rs_i + x_i}$ from random, given the random masking of $x_i$.
- $Game_3$: The ciphertext is completely independent of the challenge messages, making the adversary's advantage negligible.

A formal reduction shows that any PPT distinguisher between $Game_0$ and $Game_3$ can be transformed into a DDH distinguisher with non-negligible advantage, contradicting the DDH assumption. Therefore, we have $Adv_{\Pi,\mathcal{A}}^{IND-CPA}(\lambda) \leq negl(\lambda)$.

Additionally, because our multi-precision decomposition (Section 4.3) linearly recombines ciphertext components without introducing non-linear operations, the security reduction seamlessly extends to the $t$-tuple ciphertext case. Each decomposed component $\hat{z}_{i,k}$ is independently encrypted under the same master key, and the recombination only occurs in the decryption phase via linear summation. Thus, the IND-CPA guarantee holds for each component, and by composition, for the entire multi-precision representation. □

### *5.3 Compositional Security of the FedE Protocol*

**Theorem 3.** *(**Overall Protocol Security**). Under the combined FE and DP protections, the FedE protocol ensures that the honest-butcurious server, having access to all ciphertexts and all functional decryption keys $dk_{\mathbf{y}^{(t)}}$ for rounds $t = 1, \ldots, T$, cannot recover any client's raw weight vector $W_i^{(t)}$ or infer the presence of any individual training sample, except with probability bounded by the DP guarantee.*

**Proof:** From Theorem 1, the FE scheme ensures that ciphertexts reveal no information about $W_i$ to the server beyond the intended inner product $\langle W_i, \mathbf{y} \rangle$ even across multiple rounds, provided that each round uses fresh randomness and the master secret key remains uncompromised. From Theorem 2, the DP perturbation ensures that even if the server could infer some aggregate statistic, the contribution of any single data point is masked by calibrated noise. Since the server only obtains aggregated results from at least $n \geq 2$ clients (with $n$ the minimum aggregation threshold), and the DP noise is independently sampled per client per round, the combined effect ensures that the server's view is differentially private with respect to the union of all clients' datasets. □

Thus, the overall protocol satisfies compositional privacy. The FE protects against direct parameter reconstruction, while the DP protects against statistical inference attacks, including membership and property inference.

Under the DDH assumption and assuming the KGA is fully trusted and does not collude with the server, the FedE protocol ensures that (1) individual client ciphertexts are IND-CPA secure, revealing nothing about the plaintext except the prescribed inner product with the aggregation vector; (2) the server's view over multiple rounds reveals only the noisy aggregated inner products, which are differentially private; (3) the combined FE + DP protection prevents both direct parameter reconstruction and statistical inference attacks (membership and gradient inversion) within the bounds of the given privacy budget. These guarantees hold as long as the master secret key remains uncompromised and the randomisation of each round is fresh. If the KGA is compromised or colludes, all cryptographic privacy guarantees are void.

## 6 Performance Evaluation

### *6.1 Experimental Setup*

To evaluate the performance of the FedE framework, this section selected four publicly available datasets. The detailed information about these datasets is as follows:

- Diabetes Dataset[1]: The Diabetes Dataset contains 768 medical records related to diabetes, featuring eight variables including the number of pregnancies, blood glucose levels, blood pressure, age, and other indicators, with labels indicating whether the patient had diabetes. This dataset reflects real-world clinical prediction scenarios and can be used to test the framework's ability to learn from imbalanced data and to predict trends using critical physiological indicators.
- Heart Disease Classification Dataset[2]: The Heart Disease Dataset consists of 303 patient medical records, each comprising 13 clinical features such as age, blood pressure, cholesterol levels, and others, with labels indicating whether the patient had heart disease. This dataset is relatively small in scale and has moderate feature dimensionality, making it suitable for validating the framework's classification performance and privacy-protection effectiveness on structured sample data in MLaaS.
- Breast Cancer Diagnostic Dataset[3]: The Breast Cancer Dataset contains 569 breast cancer samples, each represented by 30 features extracted from digital images—such as radius, texture, perimeter, and other characteristics. The task is to determine whether each tumor is benign or malignant based on these features. With its high-dimensional features and clear clinical significance, this dataset is often used to assess the accuracy and reliability of machine learning models in medical diagnosis, particularly suitable for verifying the security and balance between privacy protection and utility in sensitive medical data.
- MNIST[4]: The MNIST dataset is a widely used standard benchmark dataset for image classification tasks, primarily employed for handwritten digit recognition. This dataset comprises 10 categories (digits 0–9) and contains 70,000 grayscale images, each 28 × 28 pixels, with 60,000 for training and 10,000 for testing.

All experiments were conducted on a computing platform equipped with an AMD Ryzen 7 server with 8 cores at 3.8 GHz and 32 GB of memory. Throughout the model training process, the system used only the Central Processing Unit (CPU) for computation. The security parameters were set to 256 bits to meet the security requirements of current cryptographic standards. The runtime measurements were all performed using Python's built-in "time" module. The baseline method (without encryption) and BatchCrypt [26]

[1] https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database.
[2] https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset.
[3] https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data.
[4] https://www.kaggle.com/datasets/hojjatk/mnist-dataset.

are used for comparison experiments. All experiments were conducted on a computing platform with the specifications given in Table 1. The security parameters for all experiments were set as given in Table 2. The key parameters are defined in Table 3.

**Table 1:** The computing platform.

| Component | Specification |
|---|---|
| CPU | AMD Ryzen 7 (8 cores, 3.8 GHz) |
| Memory | 32 GB Random Access Memory (RAM) |
| Operating System | Ubuntu 20.04 Long Term Support (LTS) |
| Programming Language | Python 3.8 |
| Operating System | Ubuntu 20.04 LTS |

**Table 2:** Cryptographic parameters.

| Parameter | Value | Description |
|---|---|---|
| $\lambda$ | 256 bits | Security parameter |
| $\mathbb{G}$ | 2048-bit multiplicative cyclic group | Based on the decisional Diffie–Hellman (DDH) assumption |
| $p$ | 2048-bit prime | Order of the cyclic group |
| $\eta$ | Variable | Vector length for inner-product |

**Table 3:** Precision parameters.

| Parameter | Symbol | Description |
|---|---|---|
| Precision Level | $\Delta t = 2^m$ | The numerical precision of the computation. $m$ denotes the number of bits representing the fractional part |
| Decomposition Precision | $t$ | The number of sub-vectors after precision decomposition. $t = 1$ corresponds to the traditional FE scheme. $t = 2$ and $t = 3$ correspond to our multi-precision FE scheme |
| Precision Decomposition Base | $p_{div} = 2^p$ | The base used for Euclidean decomposition, where $p$ is the bit-length of each decomposed component |
| $\eta$ | Variable | Vector length for inner-product computation |

### *6.2 Evaluation of the Computational Performance*

Table 4 compares the performance between the multi-precision inner-product functional encryption schemes across various computational precisions. Experimental results show that as computational precision increases ($2^6$ to $2^{18}$), the computational space required by traditional encryption schemes expands dramatically. At $2^{18}$ and parameter t = 1, the space overhead reaches 10.88 TB, indicating that traditional methods face significant scalability bottlenecks when handling high-precision computations. In contrast, the scheme proposed in this paper significantly reduces the computational space by introducing a ciphertext decomposition-and-combination mechanism based on balanced decomposition. Particularly

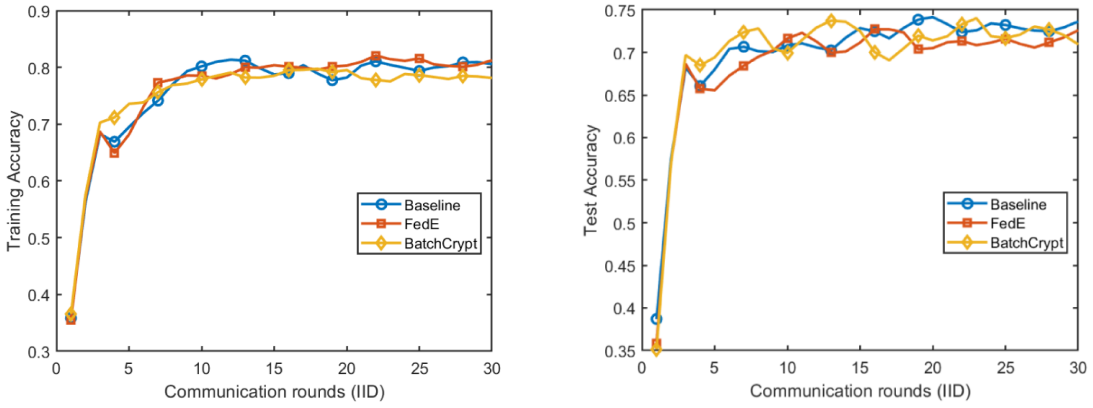
in high-precision scenarios, when parameter t = 3, the proposed scheme requires only 696 KB of storage, representing a reduction of approximately $1.0 \times 2^{24}$ times the overhead of traditional methods at t = 1. These results validate the effectiveness of the balanced decomposition strategy in controlling ciphertext expansion and improving computational storage efficiency, providing a viable technical pathway for the practical deployment of high-precision ciphertext computation.

**Table 4:** Comparison of computational cost under the same computational precision.

| Δ | $t = 1$ (Traditional Functional Encryption) | $t = 2$ | $t = 3$ (FedE) |
|---|---|---|---|
| $2^6$ | 696 KB | 10.9 KB | 2.7 KB |
| $2^{10}$ | 174 MB | 174 KB | 43.5 KB |
| $2^{14}$ | 43.5 GB | 2.74 MB | 174 KB |
| $2^{18}$ | 10.88 TB | 43.52 MB | 696 KB |

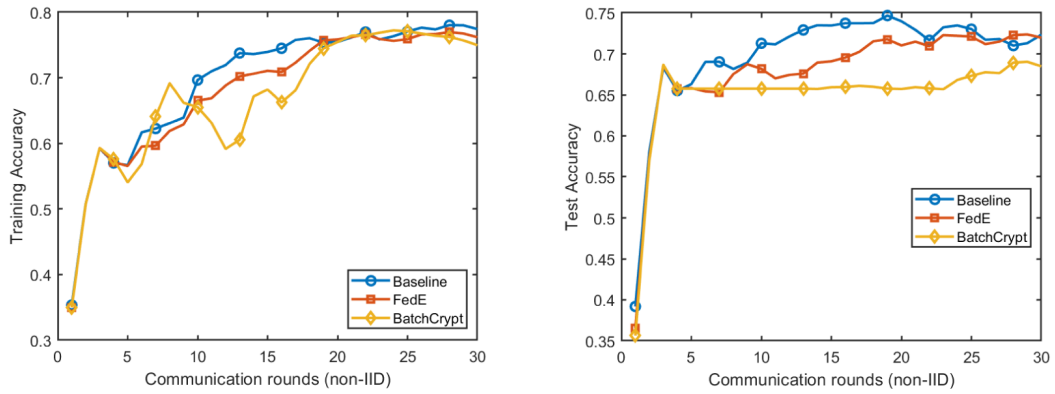
### *6.3 The Impact of Global Communication Rounds on Model Accuracy*

To analyze the impact of global communication rounds on model accuracy, this section conducted experiments with a fixed client count of 6 in federated learning and a differential privacy noise strength parameter of 0.02. The following figures (From Figs. 1–8) respectively illustrate how, under Independent and Identically Distributed (IID) and Non-IID data splits, the accuracy of various methods on the training set and test set changes as the number of global communication rounds increases for diabetes, heart disease classification, breast cancer, and the MNIST dataset. Overall, the three methods exhibit relatively consistent convergence patterns across most datasets: as the number of global communication rounds increases, both model training accuracy and test accuracy generally trend upward, eventually stabilizing after reaching a certain number of rounds. This indicates that the constructed federated learning training framework can continuously integrate local knowledge from each client through multiple rounds of parameter interaction, thereby gradually enhancing the overall classification capability of the global model.



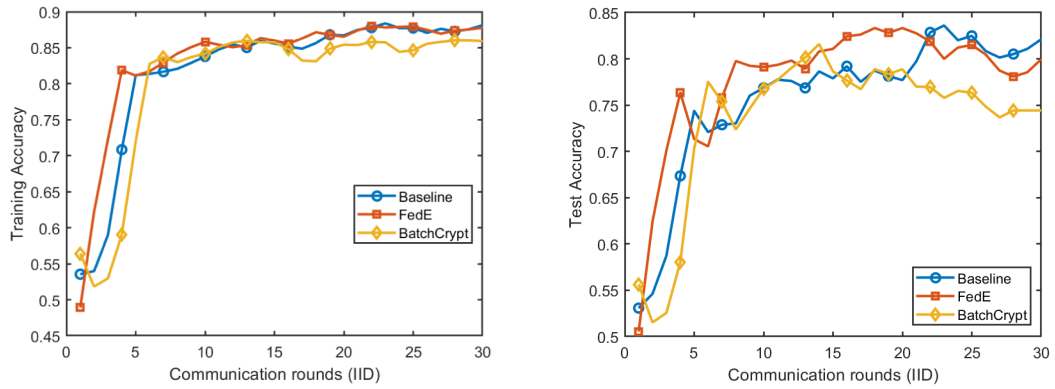

**a** Training **b** Testing

**Figure 1:** The impact of global communication rounds on model accuracy on the diabetes dataset (IID) during the (**a**) training and (**b**) testing phases, respectively.



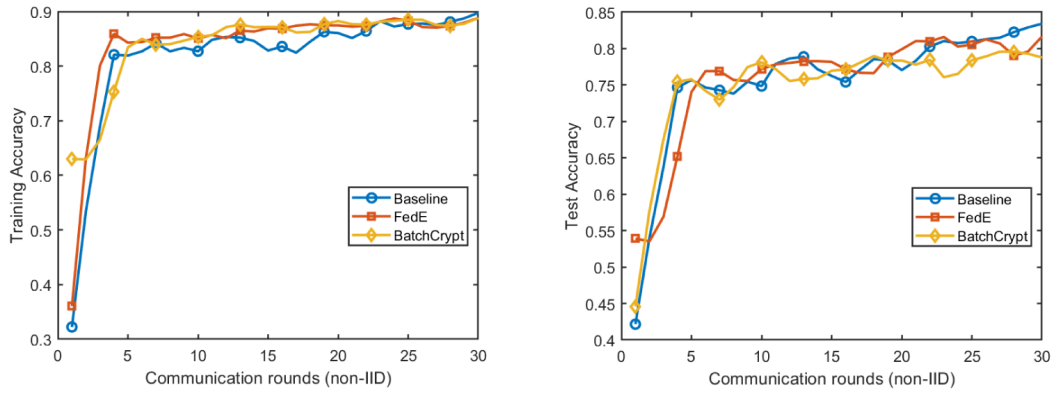

**a** Training **b** Testing

**Figure 2:** The impact of global communication rounds on model accuracy on the diabetes dataset (non-IID) during the (**a**) training and (**b**) testing phases, respectively.



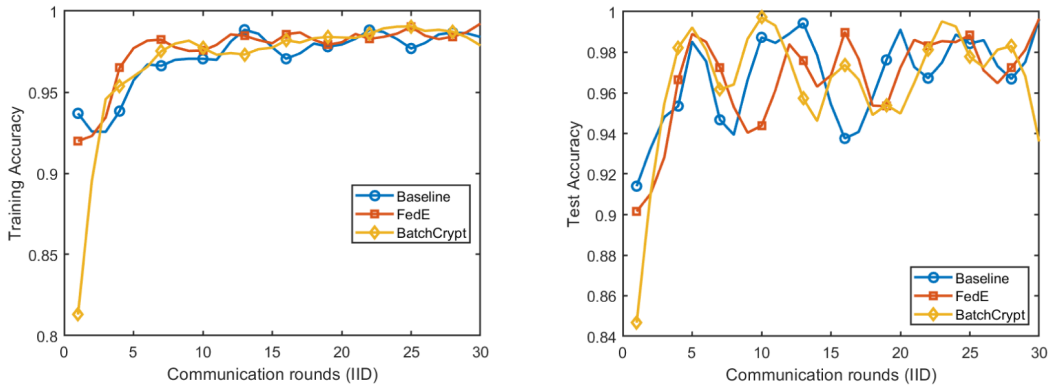

**a** Training **b** Testing

**Figure 3:** The impact of global communication rounds on model accuracy on the heart disease dataset (IID) during the (**a**) training and (**b**) testing phases, respectively.



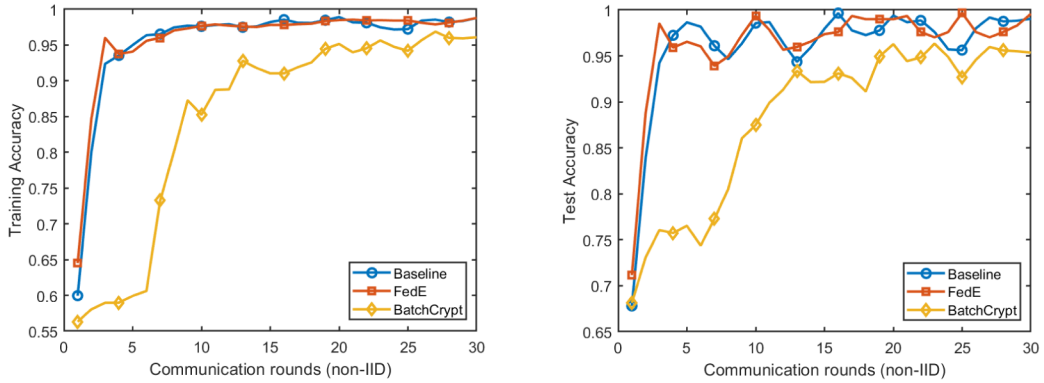

**a** Training **b** Testing

**Figure 4:** The impact of global communication rounds on model accuracy on the heart disease dataset (non-IID) during the (**a**) training and (**b**) testing phases, respectively.



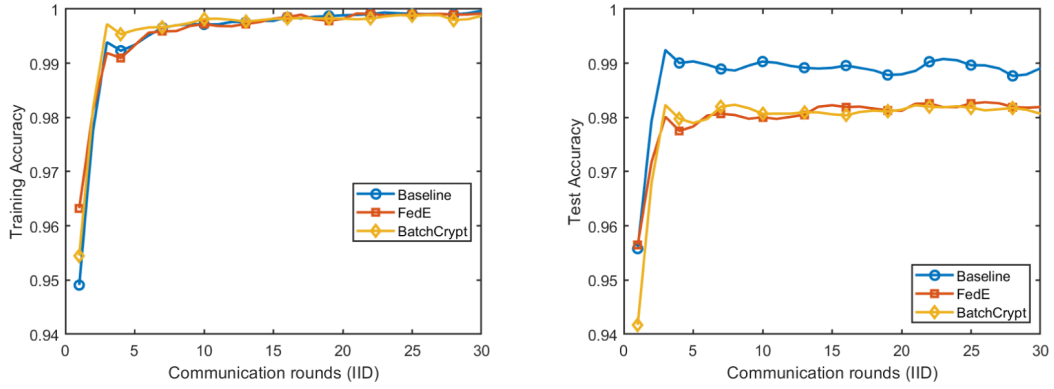

**a** Training **b** Testing

**Figure 5:** The impact of global communication rounds on model accuracy on the breast cancer diagnostic dataset (IID) during the (**a**) training and (**b**) testing phases, respectively.



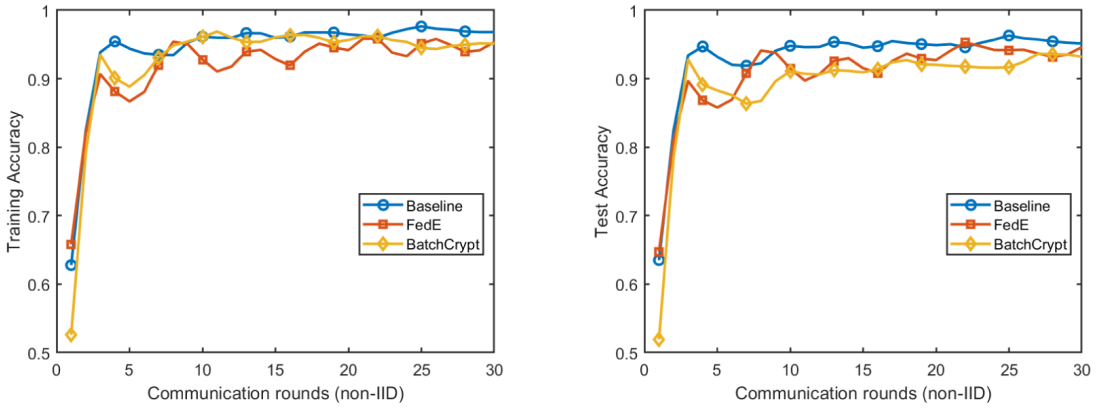

**a** Training **b** Testing

**Figure 6:** The impact of global communication rounds on model accuracy on the breast cancer diagnostic dataset (non-IID) during the (**a**) training and (**b**) testing phases, respectively.




**a** Training **b** Testing

**Figure 7:** The impact of global communication rounds on model accuracy on the MNIST dataset (IID) during the (**a**) training and (**b**) testing phases, respectively.




**a** Training **b** Testing

**Figure 8:** The impact of global communication rounds on model accuracy on the MNIST dataset (non-IID) during the (**a**) training and (**b**) testing phases, respectively.

In IID scenarios, the model convergence process is typically more stable, and the rate of improvement in training and test accuracy is also faster. The main reason is that the local data distributions across clients are relatively similar; after each round of aggregation, the global model can easily maintain a consistent direction of optimization. As a result, the training process exhibits less oscillation, and the model converges more quickly. Whether on structured disease datasets such as diabetes, heart disease classification, or breast cancer, or on the MNIST image dataset, it can be observed that as the number of communication rounds increases, the accuracy of all methods continues to improve, eventually reaching relatively high levels. This indicates that when data distributions are relatively balanced, federated learning is more effective at extracting global discriminative features and achieving stable classification.

In contrast, in Non-IID scenarios, the data distributions across clients vary significantly, making it difficult for the model to converge quickly to a unified global optimization direction in the early stages of training. As a result, the overall convergence speed is noticeably slower than in the IID scenario. As shown in Figs. 2, 4, 6, and 8, even though the training accuracy and test accuracy of all three methods continue to rise gradually with increasing communication rounds under Non-IID conditions, the rate of improvement is relatively slower, and the curves exhibit more pronounced fluctuations. Ultimately, the final accuracy is usually slightly lower than in the IID case. This phenomenon indicates that data heterogeneity weakens the effectiveness of federated aggregation, causing local updates from different clients to partially counteract one another, thereby affecting the convergence efficiency and the final performance of the global model.

Compared with the Baseline and BatchCrypt methods, FedE consistently maintains accuracy levels comparable to those of the baseline methods across datasets and data distributions. This indicates that introducing differential privacy and multi-precision inner-product-based encryption does not significantly compromise the convergence performance of federated learning models. In particular, during multiple rounds of communication, the accuracy curve of FedE generally follows the same trend as the plaintext or the benchmark approach, demonstrating that while protecting client-side local updates and ensuring the privacy of aggregation weights, it still effectively preserves the essential statistical features required for model training. This suggests that the proposed secure aggregation mechanism is highly feasible for balancing enhanced training security with minimal adverse impact on model classification performance.

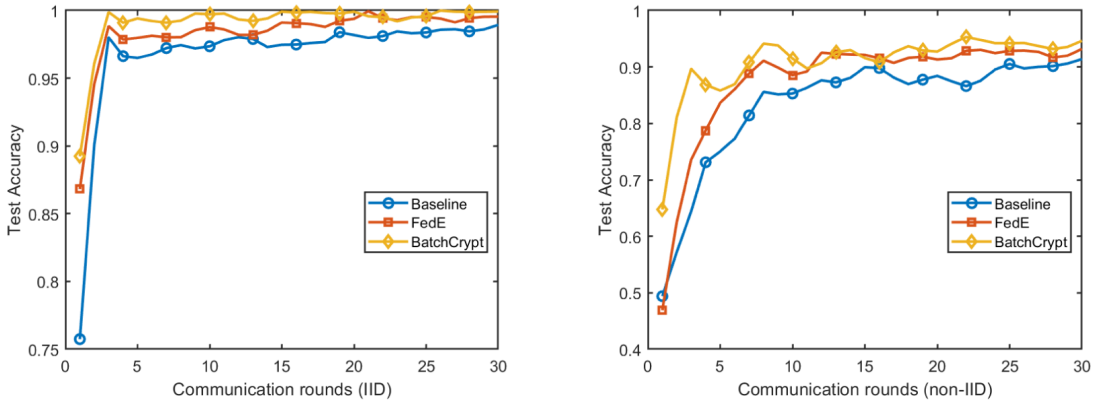
Based on the above analysis, increasing the number of global communication rounds can effectively improve the overall accuracy of federated learning models. However, the extent of this improvement is

influenced by factors such as data distribution consistency, task complexity, and the integration of security mechanisms. The proposed FedE framework exhibits relatively stable convergence in both IID and Non-IID scenarios, and it maintains accuracy levels close to those of the baseline methods across multiple disease prediction and image classification datasets, indicating that this approach enables effective multi-client collaborative training while safeguarding privacy and security.

### *6.4 The Impact of Multi-Precision on Model Accuracy*

To further analyze the impact of multi-precision mechanisms on the classification performance of federated learning models, this study compared model training results under different precision settings within a federated learning environment consistent with the aforementioned experiments. The experiment primarily examined whether, after introducing multi-precision inner-product-based encryption, the model's final classification accuracy would be significantly affected by changes in numerical representation methods, while still maintaining its ability to protect privacy.

As shown in Fig. 9, different precision settings affect model accuracy. Overall, as precision representation capabilities increase, model training outcomes tend to become more stable, and test set accuracy is more likely to remain high. This indicates that multi-precision mechanisms in federated learning scenarios can better adapt to varying ranges of data characteristics and parameter update magnitudes, thereby reducing information loss from numerical truncation, rounding, or low-precision representations. Particularly when multiple rounds of global communication and local updates from multiple clients are continuously layered together, higher or more flexible precision settings help preserve gradient changes and parameter details more accurately, thereby enhancing the model's ability to fit the global data distribution.



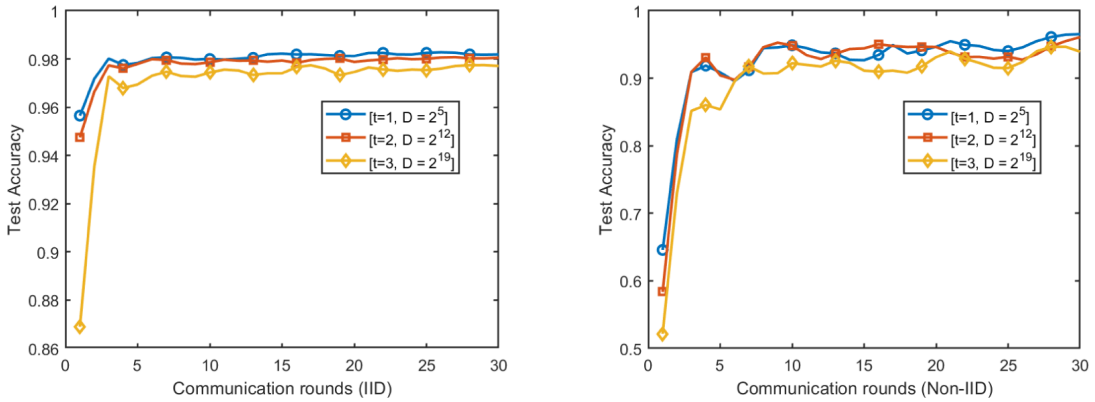

**a** Training **b** Testing

**Figure 9:** The impact of multi-precision on model accuracy for the MNIST dataset under IID and Non-IID settings during the (**a**) training and (**b**) testing phases, respectively.

### *6.5 The Impact of Noise Disturbance Intensity on Model Accuracy*

As shown in Fig. 10, the experimental results reveal that as the strength of differential privacy noise increases, the model's overall accuracy typically declines. This is because while stronger noise perturbations can effectively enhance privacy protection, they also weaken the ability of client-side local parameter updates to accurately capture the true data distribution. As a result, the local model information uploaded to the server contains more random components, which in turn affects the global aggregation process. In particular, in federated learning scenarios, local updates from multiple clients must be accumulated and integrated over multiple iterations. When each client introduces strong noise, random perturbations are gradually

transmitted and accumulated across the global training process, leading to slower model convergence and a decline in test accuracy.




**a** Training **b** Testing

**Figure 10:** The impact of noise perturbation intensity on model accuracy for the MNIST dataset under IID and Non-IID settings during the (**a**) training and (**b**) testing phases, respectively.

### *6.6 The Impact of Training Iterations on Time Efficiency*

Here, we compare the execution times of the FedE, centralized training, and BatchCrypt schemes on the MNIST dataset, both locally and on the cloud server. By comparing the results, we observed how the time overhead of the three algorithms varied across different communication rounds. As shown in Fig. 11, as the number of communication rounds increased from 1 to 10, the cumulative time consumption for all methods exhibited nearly linear growth; however, the rate of increase differed significantly. Among them, the baseline method had the lowest time overhead—since the entire process was completed in plaintext, the total time consumed after all 10 rounds of communication was approximately 260 s. In contrast, the FedE method proposed in this paper, by introducing multi-precision inner-product functional encryption and differential privacy mechanisms, saw its cumulative time consumption increase to about 900 s over the same 10 communication rounds, which is roughly 3.46 times that of the baseline scheme. This increase was primarily due to the additional computational burden imposed by encryption and decryption operations. Meanwhile, the comparison method, BatchCrypt, which employed a batch-based fully homomorphic encryption strategy, experienced the highest time overhead, with 10 rounds of communication. It reached approximately 2500 seconds, which was about 2.78 times that of FedE and 9.62 times that of the baseline scheme. Notably, during the initial phase (the first 3 rounds), each. The time overhead differences between the methods were relatively small; as the number of rounds increased, the cumulative computational burden from encryption operations became pronounced, and the time gap between the methods gradually widened. The experimental results confirmed that, while ensuring security, FedE offers a significant computational efficiency advantage over traditional fully homomorphic encryption schemes.

Based on the above experimental results, it is evident that each of the three key technologies in this scheme plays a distinct role during model training. Differential privacy primarily enhances privacy protection during the client-side local update phase, with its impact mainly reflected in changes in noise intensity on model accuracy. The multi-precision mechanism is designed to mitigate numerical representation errors in scenarios involving heterogeneous data and ciphertext-based computations; its advantage lies in the differences between model accuracy and convergence stability across various precision settings. Functional encryption, on the other hand, enables secure aggregation by servers, with its value primarily stemming

from the feasibility of ciphertext-based computations, reduced communication overhead, and the ability to protect privacy during aggregation. Experimental results demonstrate that these three techniques are not independent of one another—they work together within the FedE framework, ensuring both privacy security and, as much as possible, maintaining the performance of federated learning models.

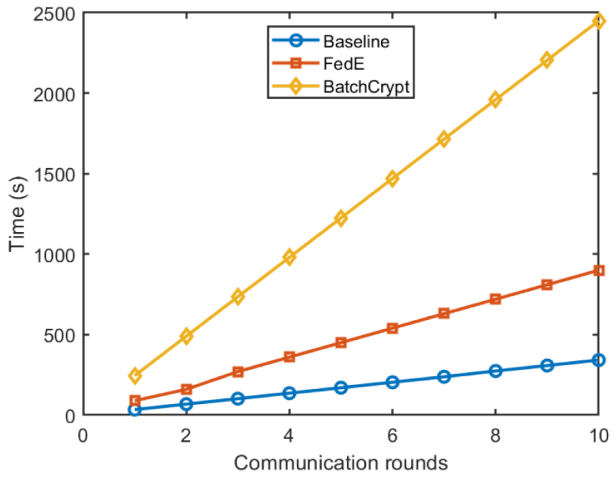



**Figure 11:** Time consumption of the MNIST dataset across different communication rounds.

### *6.7 Comparison with more State-of-the-Arts*

We compare FedE with TAPFed (2024) [27], EncCluster (2025) [28], PEFed (2025) [29], and BatchCrypt (2020) [26]. To ensure a fair and transparent comparison, we categorize the baselines into two groups. (i) Reproduced methods: BatchCrypt [29] and PEFed were re-implemented and evaluated under the exact same hardware environment (AMD Ryzen 7, 32 GB RAM, Ubuntu 20.04), dataset splits (Breast Cancer and MNIST, Non-IID with $\alpha = 0.5$), number of clients ($n = 6$), communication rounds ($R = 50$), model architectures, and privacy settings as our FedE. All reported results are averages over 10 independent runs. (ii) Literature-cited methods: TAPFed and EncCluster are cited directly from their original papers, as their specialised protocols (threshold secret sharing for TAPFed and weight clustering for EncCluster) lack public implementations and their core mechanisms fundamentally alter the aggregation pipeline, making fair re-implementation prone to biases. We selected results from those papers that use the same Breast Cancer dataset under Non-IID settings with 6–10 clients. For efficiency, although hardware varies, FedE's 5× speedup over TAPFed and 4× over EncCluster is substantial enough to remain valid across typical hardware discrepancies. Privacy configurations for each method follow the default settings recommended in their respective original works.

From Table 5, FedE achieves 98% accuracy, which is the highest among all compared methods, tying with TAPFed. FedE's multi-precision inner-product functional encryption combined with differential privacy does not compromise model utility even under challenging Non-IID data distributions. The accuracy improvement over PEFed (+3.2%), EncCluster (+2%), and BatchCrypt (+1%) confirms the effectiveness of the proposed multi-precision decomposition mechanism in preserving fine-grained parameter information during encrypted aggregation. FedE completes 10 rounds of training in 15 min, which is 5.2× faster than TAPFed (1.3 h), 4.4× faster than EncCluster (1.1 h), and 2.1× faster than PEFed (32 min). While BatchCrypt achieves the fastest runtime at 9 min, FedE provides a very competitive efficiency (15 min) while delivering superior accuracy (98%). This demonstrates a better accuracy-efficiency trade-off compared to BatchCrypt, which sacrifices accuracy for speed.

**Table 5:** Comparison results with SOTA on the breast cancer diagnostic dataset (non-IID).

| Method | Model Accuracy | Computational Efficiency (10 Rounds) |
|---|---|---|
| TAPFed | 98% | 1.3 h |
| EncCluster | 96% | 1.1 h |
| PEFed | 94.8% | 32 min |
| BatchCrypt | 97% | 9 min |
| FedE (Ours) | 98% | 15 min |

## 7 Conclusion

As data resources become increasingly distributed across institutions, terminals, and platforms, collaborative computation among multiple clients has gradually emerged as an important approach for implementation in MLaaS scenarios. This paper conducts a comprehensive study of inner-product functional encryption technology and proposes a multi-precision inner-product functional encryption scheme. It incorporates differential privacy techniques to inject adaptive noise into raw data, and employs multi-input, multi-precision inner-product functions to securely aggregate model parameters during the federated learning process under the FedE strategy. By integrating client data from multiple organizations across multiple training rounds, a federated learning framework with strong privacy protection capabilities is constructed. Through targeted experiments on the diabetes, heart disease classification, breast cancer, and MNIST (classic handwritten digits) datasets, we compared and analyzed the advantages of using differential privacy and multi-precision functional encryption for privacy-preserving federated learning.

**Acknowledgement:** We express our gratitude to Guizhou Power Grid Co., Ltd., for their invaluable support and assistance in this investigation.

**Funding Statement:** This paper is supported in part by the Major Research Project of Guizhou Provincial Laboratory (QKHPT-SSYS [2024]014) and in part by the Natural Science Foundation of China no. 62362008.

**Author Contributions:** Conceptualization, Weijia Liu, Hao Li, Zhenyong Zhang, and Junwen Deng; methodology, Weijia Liu, Junwen Deng, Hao Li, and Zhenyong Zhang; software, Weijia Liu, Hao Li, Junwen Deng, and Zhenyong Zhang; validation, Weijia Liu, Junwen Deng, Hao Li, and Zhenyong Zhang; formal analysis, Weijia Liu, Junwen Deng, Hao Li, and Zhenyong Zhang; investigation, Weijia Liu, Junwen Deng, Hao Li, and Zhenyong Zhang; resources, Weijia Liu, Junwen Deng, Hao Li, and Zhenyong Zhang; data curation, Weijia Liu, Junwen Deng, Hao Li, and Zhenyong Zhang; writing—original draft preparation, Weijia Liu and Zhenyong Zhang; writing—review and editing, Weijia Liu and Zhenyong Zhang; visualization, Weijia Liu and Zhenyong Zhang; supervision, Zhenyong Zhang; project administration, Zhenyong Zhang; funding acquisition, Zhenyong Zhang. All authors reviewed and approved the final version of the manuscript.

**Availability of Data and Materials:** Data openly available in a public repository.

**Ethics Approval:** Not applicable.

**Conflicts of Interest:** Given his role as Editorial Board Member of this journal, Zhenyong Zhang had no involvement in the peer review of this article and had no access to information regarding its peer review. Full responsibility for the editorial process for this article was delegated to another journal editor. The authors declare no other conflicts of interest.

## References


1. Almanasra S. Applications of integrating artificial intelligence and big data: a comprehensive analysis. J Intell Syst. 2024;33(1):20240237. doi:10.1515/jisys-2024-0237.
2. Zhang Z, Liu M, Sun M, Deng R, Cheng P, Niyato D, et al. Vulnerability of machine learning approaches applied in IoT-based smart grid: a review. IEEE Internet Things J. 2024;11(11):18951–75. doi:10.1109/jiot.2024.3349381.
3. Zhang Z, Wang W, Wang M, Wang H, Bi J, Xu G. Physics-constrained adversarial attack generation and active defense for the hybrid deep reinforcement learning-based load frequency control. ACM Trans Auton Adapt Syst. 2026:3805703. doi:10.1145/3805703.
4. Zhang Z, Shao K, Deng R, Wang X, Zhang Y, Wang M. PrivLSTM: a privacy-preserving LSTM inference framework by fusing encryption and network structure for multi-sourced Data. Inf Fusion. 2026;127(7):103711. doi:10.1016/j.inffus.2025.103711.
5. Peng S, Yang Y, Mao M, Park DS. Centralized machine learning versus federated averaging: a comparison using MNIST dataset. KSII Trans Internet Inf Syst. 2022;16(2):742–56. doi:10.3837/tiis.2022.02.020.
6. Abdalla M, Bourse F, De Caro A, Pointcheval D. Simple functional encryption schemes for inner products. In: IACR International Workshop on Public Key Cryptography. Berlin/Heidelberg, Germany: Springer; 2015. p. 733–51.
7. Panzade P, Takabi D, Cai Z. Privacy-preserving machine learning using functional encryption: opportunities and challenges. IEEE Internet Things J. 2024;11(5):7436–46. doi:10.1109/jiot.2023.3338220.
8. Bai L, Hu H, Ye Q, Li H, Wang L, Xu J. Membership inference attacks and defenses in federated learning: a survey. ACM Comput Surv. 2025;57(4):1–35. doi:10.1145/3704633.
9. Zhang Z, Cheng P, Wu J, Chen J. Secure state estimation using hybrid homomorphic encryption scheme. IEEE Trans Control Syst Technol. 2021;29(4):1704–20. doi:10.1109/tcst.2020.3019501.
10. Wu L, Wang XA, Liu J, Su Y, Tu Z, Liu W, et al. Homomorphic encryption for machine learning applications with CKKS algorithms: a survey of developments and applications. Comput Mater Contin. 2025;85(1):89–119. doi:10.32604/cmc.2025.064346.
11. Lee JW, Kang H, Lee Y, Choi W, Eom J, Deryabin M, et al. Privacy-preserving machine learning with fully homomorphic encryption for deep neural network. IEEE Access. 2022;10:30039–54. doi:10.1109/access.2022.3159694.
12. Park S, Byun J, Lee J. Privacy-preserving fair learning of support vector machine with homomorphic encryption. In: Proceedings of the ACM Web Conference 2022; 2022 Apr 25–29; Virtual Event. p. 3572–83.
13. Carpov S, Gama N, Georgieva M, Troncoso-Pastoriza JR. Privacy-preserving semi-parallel logistic regression training with fully homomorphic encryption. BMC Med Genom. 2020;13(7):88.
14. Castro F, Impedovo D, Pirlo G. An efficient and privacy-preserving federated learning approach based on homomorphic encryption. IEEE Open J Comput Soc. 2025;6:336–47. doi:10.1109/ojcs.2025.3536562.
15. Manderna A, Dohare U, Kumar S, Ram B. Intrusion detection in Internet of Things using differential privacy: a hybrid machine learning approach. Ad Hoc Netw. 2025;174(16):103818. doi:10.1016/j.adhoc.2025.103818.
16. Demelius L, Kern R, Trügler A. Recent advances of differential privacy in centralized deep learning: a systematic survey. ACM Comput Surv. 2025;57(6):1–28. doi:10.1145/3712000.
17. Lian Z, Wang W, Su C. COFEL: communication-efficient and optimized federated learning with local differential privacy. In: Proceedings of the ICC 2021-IEEE International Conference on Communications; 2021 June 14–23; Montreal, QC, Canada; p. 1–6.
18. Hu R, Guo Y, Gong Y. Federated learning with sparsified model perturbation: improving accuracy under client-level differential privacy. IEEE Trans Mob Comput. 2024;23(8):8242–55. doi:10.1109/tmc.2023.3343288.
19. Wei K, Li J, Ma C, Ding M, Chen W, Wu J, et al. Personalized federated learning with differential privacy and convergence guarantee. IEEE Trans Inf Forensics Secur. 2023;18:4488–503. doi:10.1109/tifs.2023.3293417.
20. Sorbera E, Zanetti F, Brandi G, Tomasi A, Doriguzzi-Corin R, Ranise S. Adaptive federated learning with functional encryption: a comparison of classical and quantum-safe options. arXiv:2504.00563. 2025.

21. Yu P, Huang W, Zhang R, Qian X, Li H, Chen H. GuardGrid: a queriable and privacy-preserving aggregation scheme for smart grid via function encryption. IEEE Internet Things J. 2025;12(11):17622–33. doi:10.1109/jiot.2025.3539724.
22. Qian X, Li H, Hao M, Yuan S, Zhang X, Guo S. CryptoFE: practical and privacy-preserving federated learning via functional encryption. In: Proceedings of the GLOBECOM 2022–2022 IEEE Global Communications Conference; 2022 Dec 4–8; Rio de Janeiro, Brazil. p. 2999–3004.
23. Chang Y, Zhang K, Gong J, Qian H. Privacy-preserving federated learning via functional encryption, revisited. IEEE Trans Inf Forensics Secur. 2023;18:1855–69. doi:10.1109/tifs.2023.3255171.
24. Zhang R, Li H, Qian X, Jiang W, Zhang X. An efficient and secure privacy-preserving federated learning via lattice-based functional encryption. In: Proceedings of the ICC 2024-IEEE International Conference on Communications; 2024 Jun 9–13; Denver, CO, USA. p. 2185–90.
25. Yu B, Zhao J, Zhang K, Gong J, Qian H. Lightweight and dynamic privacy-preserving federated learning via functional encryption. IEEE Trans Inf Forensics Secur. 2025;20:2496–508. doi:10.1109/tifs.2025.3540312.
26. Zhang C, Li S, Xia J, Wang W, Yan F, Liu Y. BatchCrypt: efficient homomorphic encryption for cross-Silo federated learning. In: Proceedings of the 2020 USENIX Conference on Usenix Annual Technical Conference; 2020 Jul 15–17; Boston, MA, USA. p. 493–506.
27. Xu R, Li B, Li C, Joshi JBD, Ma S, Li J. TAPFed: threshold secure aggregation for privacy-preserving federated learning. IEEE Trans Dependable Secure Comput. 2024;21(5):4309–23. doi:10.1109/tdsc.2024.3350206.
28. Tsouvalas V, Mohammadi S, Balador A, Ozcelebi T, Flammini F, Meratnia N. EncCluster: scalable functional encryption in federated learning through weight clustering and probabilistic filters. Pervasive Mob Comput. 2025;108(5):102021. doi:10.1016/j.pmcj.2025.102021.
29. Guan M, Bao H, Wang J, Xing L, Dai HN. PEFed: enhancing privacy and efficiency in federated learning via removable perturbation and decentralized encryption. Inf Fusion. 2025;122(3):103187. doi:10.1016/j.inffus.2025.103187.