Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.082755

ARTICLE

A Privacy-Preserving Aggregation Mechanism with Multi-Key Support and Short Ciphertexts for Federated Learning

Hongzhen Liu[1], Liang Xie[1], Zhiqiang Ru[2,*], Yuan Wan[1], Zhe Zhang[1] and Xi Fang[1,*]

[1]School of Mathematics and Statistics, Wuhan University of Technology, Wuhan, China
[2]Xi'an Jiaotong University-China Mobile Communications Group Co., Ltd. Digital Government Joint Research Institute, Xi'an, China
*Corresponding Authors: Zhiqiang Ru. Email: ruzhiqiang@chinamobile.com; Xi Fang. Email: fangxi@whut.edu.cn

Received: 22 March 2026; Accepted: 08 May 2026; Published: 23 July 2026

ABSTRACT: Federated learning is a privacy-preserving machine learning framework that facilitates model training directly on decentralized data that, due to privacy concerns or transmission costs, cannot be centralized on a server for traditional model training. To prevent adversaries from reconstructing the original data via parameters transmitted during the process, homomorphic encryption is a commonly adopted method. However, it introduces significant communication and computation costs and risks total security failure if any secret key is compromised. This paper proposes a privacy-preserving aggregation mechanism that enables each client to independently generate partial keys for encryption while allowing decryption after homomorphic operations using an aggregated key. Key aggregation for the proposed algorithm is realized through secret sharing. Incorporating these components into a standard federated learning framework yields a novel method that enhances communication efficiency and offers robustness against privacy breaches from internal collusion. The algorithm's resistance to linear and differential attacks is formally demonstrated by algebraically modeling the encryption procedure. Based on this analysis, the overall security of the method is likewise established. Experiments on the privacy-preserving aggregation mechanism demonstrate that the generated ciphertext exhibits favorable statistical properties and sensitivity. Simulation results of the federated learning method further indicate that, compared to existing encryption schemes, our proposed encryption method reduces communication cost by 77%~93% with acceptable computational cost, thereby enabling lightweight encryption in federated learning.

KEYWORDS: Federated learning; homomorphic encryption; lightweight encryption; secret sharing; communication efficiency; privacy preservation

1 Introduction

Federated learning, represented by Federated Averaging (FedAvg) [1], has been extensively studied due to its straightforward framework and practical functionality. By transmitting and aggregating model parameters among different clients, this approach enables collaborative model training across multiple data owners. Although uploading only model parameters or gradients rather than the original local datasets can significantly reduce the risk of privacy leakage and communication cost during training, Subsequent studies have shown that these parameters and gradients may still expose sensitive information. Techniques such as Deep Leakage from Gradients (DLG) [2] have demonstrated that a semi-honest server or eavesdroppers from either internal or external sources can reconstruct original data from the transmitted parameters.

To better protect clients' original data, various privacy-preserving techniques and corresponding federated learning methods have been developed. From the perspective of protection level, existing efforts can be categorized into participant anonymization [3], raw data protection [4], and protection of transmitted gradients. In terms of privacy-preserving methods, participant privacy can be safeguarded through approaches such as perturbation-based differential privacy [5], cryptography-based homomorphic encryption [6], and mask [7] based on secure multi-party computation.

In this paper, we construct a privacy-preserving aggregation scheme based on masking and homomorphic encryption. Homomorphic encryption is a special type of encryption algorithm that allows computations to be performed directly on ciphertexts without decrypting them. The result of these ciphertext operations corresponds to the result of operations on the original plaintexts. Existing homomorphic encryption schemes support operations such as addition between ciphertexts, multiplication between ciphertexts and plaintexts, and even multiplication between ciphertexts [8]. In federated learning, this property can be leveraged by encrypting raw model parameters into ciphertexts, performing aggregation in the cloud, and then decrypting the aggregated result locally. Through this process, collaborative training can be conducted without compromising privacy. Owing to the ease of deployment and strong security guarantees of this method, homomorphic encryption has become one of the key technologies for privacy preservation in federated learning [9]. In traditional homomorphic encryption schemes, the same set of public and private keys is used for all encryption and decryption operations. Under the common federated learning environment, if all parties hold the same key pair, the security of the entire system would be compromised if one client's key is leaked. Moreover, the use of a shared key set introduces the risk that semi-honest clients could compromise each other's data privacy. Currently, multi-key homomorphic encryption, which allows each client to use different keys for encryption and the server to use an aggregated key for decryption, is widely used in federated learning [10].

Although multi-key homomorphic encryption can alleviate the issue of key leakage inherent in conventional homomorphic encryption schemes within federated learning, its underlying encryption process and ciphertext structure remain fundamentally based on traditional designs [11]. Without targeted optimization, it still incurs high computational and communication costs. Reducing the high computational and communication costs caused by the use of homomorphic encryption in federated learning is still an urgent issue that needs to be addressed.

To reduce ciphertext size and the computational cost of encryption and decryption, this study constructs and a privacy-preserving method based on a symmetric encryption algorithm. In this work, to reduce the overall communication cost of federated learning and to prevent privacy leakage from internal participants, we propose a multi-key, short-ciphertext privacy-preserving aggregation mechanism for federated learning (MKPHE-FL). This algorithm is constructed based on a multi-party privacy-preserving mechanism, in which traditional encryption operators are extended to rings, thereby enabling homomorphism and supporting a multi-key setting. A secret sharing-based key negotiation protocol (SS-KNP) allows the aggregated key to be delivered to the server without revealing individual private keys. On this basis, a communication-efficient federated learning method is developed. The security and effectiveness of the proposed algorithm are validated through both theoretical analysis and experimental evaluation. The overall method preserves user privacy with negligible impact on the final model accuracy, while maintaining moderate computational cost and low communication overhead. The main contributions of this paper are as follows:

- In the context of federated learning, a multi-key, short-ciphertext privacy-preserving aggregation mechanism is proposed. The short-ciphertext property stems from the use of a symmetric encryption framework, while the multi-key capability is achieved by incorporating an initialization vector mask during the encryption process. The main advantage of the mechanism lies in its ability to

preserve homomorphism while maintaining a relatively short ciphertext length, all with an acceptable computational cost.

- To aggregate the individual keys held by each client in the privacy-preserving aggregation mechanism into a unified decryption key accessible to the server, a key agreement protocol is designed based on the homomorphic properties of secret sharing. This design provides robustness against collusion, preventing any subset of clients and the server up to a certain threshold from reconstructing the private keys of others.
- By integrating the proposed algorithm into the conventional federated learning framework, a novel method is developed that prevents collusion-based privacy leakage during both the aggregation and key agreement phases. This method is not only communication-efficient but also reduces the overall communication overhead of the system by delegating the decryption process to the server.
- The security of the privacy-preserving aggregation mechanism is first demonstrated through linear algebraic modeling, as well as formal proofs regarding differential attacks, linear relationships, and the overall security of the mechanism. Subsequently, the overall security of the federated learning method is assessed based on the security properties of the privacy-preserving aggregation mechanism and the secret sharing scheme. Finally, experimental results further demonstrate the algorithm's security and the communication efficiency of the proposed federated learning approach.

The rest of this paper is organized as follows: Section 2 presents recent related work on communication efficiency optimization in federated learning and the use of homomorphic encryption for privacy preservation. Section 3 introduces the proposed encryption algorithm, the key agreement protocol, and the federated learning method, and provides the analysis of the communication cost and security of the method. Section 4 provides experiments and analysis on the security of the privacy-preserving aggregation mechanism as well as the computational and communication overhead of the proposed federated learning framework. Section 5 concludes the paper.

2 Related Work

2.1 Communication Efficiency Optimization in Federated Learning

The most straightforward approach to reducing the overall communication and computational costs in federated learning is to minimize the volume of data exchanged between the server and each client. This concern has evolved into a distinct line of research within federated learning, commonly referred to as efficiency optimization. Current studies on communication efficiency primarily employ techniques such as lazy aggregation, sparsification, and quantization. Despite their differences in implementation, these methods share a common objective: to discard redundant information and transmit only the most critical data required for model updates. Sun et al. [12] proposed the LAQ algorithm, which computes the gradient update (i.e., the difference between the current gradient and the previously quantized one). If a worker's gradient update falls below a predefined threshold, communication of that worker's gradient is skipped. In FetchSGD introduced by Rothchild et al. [13], dimensionality reduction is first performed locally using the sketch algorithm. The resulting parameters are then transmitted to the server, where a top-k operation eliminates the least significant ones. Subsequently, the reduced global model is further compressed and sent back to the local clients. Quantization aims to reduce communication cost by converting floating-point values typically responsible for high transmission costs into lower-bit integer representations. Jhunjhunwala et al. [14] argue that applying low-precision quantization during the early and intermediate training stages does not hinder model convergence. As long as precision is gradually increased during the later stages, the final model performance remains unaffected. Based on this insight, an adaptive quantization mechanism is proposed. Hu and Li [15] analyzes the importance of individual parameters in participants' uploaded models

to inference attacks through local model gradient, and achieves effective defense against inference attacks by encrypting only a very small subset of parameters.

Singh et al. [16] proposed a relay-chain-driven architecture that integrates context-aware smart contracts with an enhanced ciphertext-policy attribute-based encryption (CP-ABE) scheme. For highly sensitive events, the model enforces robust multi-attribute access policies, whereas lightweight policies are adopted for routine data updates to reduce computational overhead. In addition, the cryptographic framework provides traceability and low-latency privilege revocation. This distributed and scalable model achieves a sound balance between real-time responsiveness and security.

In addition to directly reducing the number of parameters transmitted, a range of lightweight strategies designed for specific encryption algorithms has been employed to alleviate the communication burden in federated learning systems. Zhang et al. [17] significantly reduces the size and quantity of ciphertexts generated during encryption by optimizing the encoding scheme. Specifically, multiple quantized gradients are packed into a single long integer, allowing for one-time encryption and thus minimizing communication cost for an encryption algorithm based on long integers. Zhang et al. [18] introduce a hybrid approach that leverages both homomorphic encryption and masking techniques. In this method, local data quality is used to weight model aggregation, accelerating convergence. The quality metrics are protected using homomorphic encryption, which, although computationally intensive, is negligible compared to the complete set of model parameters. Meanwhile, the model parameters are safeguarded employing a more lightweight masking technique, thereby balancing security, convergence speed, and computational efficiency. Burlachenko et al. [19] adopt the symmetric Advanced Encryption Standard (AES) encryption algorithm to secure parameter transmission, significantly reducing both computational and communication costs. In this scheme, the server merely collects and redistributes the encrypted gradient updates. Additionally, the stateless compressed sensing algorithm is employed to prevent data leakage to the host and further mitigate communication costs. Pan et al. [20] identify an optimal parameter configuration for the Cheon–Kim–Kim–Song (CKKS) homomorphic encryption scheme based on empirical analysis within a conventional federated learning framework, thereby improving computational efficiency and reducing communication cost.

Methods for reducing communication costs can be broadly categorized into general approaches that do not account for privacy considerations and specialized techniques designed to efficiently apply specific encryption algorithms. However, there has been relatively little research focused on directly improving encryption algorithms themselves to reduce communication cost. Chait et al. [21] proposed a symmetric homomorphic encryption algorithm whose ciphertext size is comparable to that of the plaintext and whose computational complexity is O(n). Ming et al. [22] applied an efficient symmetric homomorphic encryption scheme to federated learning, achieving reduced computational and communication overhead. Hariss and Noura [23] previously proposed a lightweight encryption algorithm tailored for resource-constrained IoT devices. However, due to differences in privacy requirements and hardware environments, this scheme is not fully suitable for federated learning applications. Building upon the structure of the algorithm, this paper introduces a homomorphic encryption algorithm adapted to federated learning by improving its parameter settings, encryption mode, and key negotiation protocol.

Azzouzi et al. [24] proposed fine-grained field-programmable gate array (FPGA) architectures for the MixColumns and InvMixColumns transformations in the Advanced Encryption Standard (AES) to improve performance and optimize resource utilization. The proposed designs exhibit competitive performance compared with state-of-the-art FPGA implementation schemes in the current literature and industry.

2.2 Federated Learning Combined with Encryption

In federated learning, secret-sharing primitives are commonly employed to achieve privacy preservation or tamper-resistance. However, encryption methods relying on a single key may be vulnerable to key leakage.An intuitive approach to mitigating key leakage is to enable each client to independently encrypt data using distinct keys, a concept known as multi-key encryption. Early studies in this area primarily focused on multi-key extensions of lattice-based cryptographic algorithms. Over time, multi-key homomorphic encryption algorithms based on various cryptographic frameworks have been increasingly adopted in federated learning.

Ma et al. [10] address the risks of data leakage and collusion associated with using a shared key by introducing a multi-key homomorphic encryption approach. This method permits participants to encrypt data using individual keys and ensures that the aggregated result can only be decrypted if all participants provide their respective decryption shares. Similarly, Park and Lim [25] propose a distributed homomorphic encryption method based on secure multi-party computation to mitigate the challenges introduced by a single-key setup, such as increased key distribution cost and heightened risk of key exposure. In this work, parameter aggregation is performed collaboratively by two non-colluding servers. Jiang et al. [26] construct a multi-key homomorphic encryption system utilizing a double trapdoor function and design a federated learning method involving two non-colluding servers. The privacy of the ciphertext is further protected by incorporating ciphertext perturbation between the servers. Cai et al. [27] replace the trusted third party traditionally required in homomorphic encryption-based federated learning methods with a trusted execution environment (TEE) deployed on the server. By enhancing the existing multi-key homomorphic encryption algorithms, this method improves both efficiency and robustness against mid-training dropouts, thus achieving a fault-tolerant, privacy-preserving federated learning system. Kumbhar and Rao [4] employs the PAOA optimization algorithm to generate cryptographic keys that perform reasonably well and applies multi-key homomorphic encryption directly to the datasets of each participant. Pham et al. [28] utilize multi-key homomorphic encryption to secure each participant's model, while additionally employing random masking with xor operations to protect local data. Li et al. [29] implement model aggregation via re-encryption, allowing participants to encrypt with distinct public keys. A federated learning method is built upon this technique using two non-colluding servers. In terms of application, Walskaar et al. [30] provide a practical implementation of Ma's [10] scheme using numpy and the Flower framework, and demonstrate its applicability in medical data scenarios.

In addition to preventing single-point key leakage, other cryptographic tools can also be employed in encryption algorithms to meet diverse functional requirements in federated learning. Wu et al. [31] utilized blockchain and multi-authority attribute-based encryption (MA-ABE) to construct a highly secure encryption system, supporting dynamic revocation under partial policy hiding, verifiable lightweight operations, and resistance to offline dictionary attacks and multiple collusion attacks within the system. Tian et al. [32] proposed a dynamic privilege attribute-based encryption scheme based on a multi-authority architecture, enabling changes in decryption privileges and auditable records of access behaviors. Finally, after the training of federated learning is completed, homomorphic encryption can also be used to perform model inference in the ciphertext state while protecting the privacy of participants. Kong et al. [33] implemented a fully privacy-preserving machine-learning-based model predictive control method. They constructed a homomorphic encryption recurrent neural network (HE-RNN), which supports training and inference on ciphertext inputs.

In current privacy-preserving federated learning methods based on multi-key homomorphic encryption, some require multiple rounds of fixed communication, which reduces the overall robustness of the process. Others rely on third parties, such as non-colluding dual servers, to perform privacy operations,

resulting in a relatively complex system. In this work, the above issue is mitigated by designing a federated learning method that includes only semi-honest, potentially colluding participants and a server with the same characteristics. Furthermore, the privacy-preserving components employed in our federated learning approach consist solely of the proposed encryption algorithm and key negotiation protocol, along with an eavesdropping-resistant secure channel for key transmission and negotiation.

The specific innovations of this study, as compared with the aforementioned studies, are presented in Table 1.

Table 1: Comparison with related work.

Ref.	No Dual-Server Coordination	Resistant to Single-Point Key Leakage	Communication Efficient	Privacy Preserving	No Accuracy Loss	Cryptographic Tool
Sun et al. [12]	✓	×	✓	×	×	×
Jhunjhunwala et al. [14]	✓	×	✓	×	×	×
Zhang et al. [18]	✓	×	✓	✓	✓	HE
Burlachenko et al. [19]	✓	×	✓	✓	✓	NHE
Pan et al. [20]	✓	×	✓	✓	✓	HE
Hariss and Noura [23]	✓	×	✓	✓	×	SHE
Park and Lim [25]	×	×	×	✓	×	HE
Jiang et al. [26]	×	×	×	✓	×	MKHE+SA
Cai et al. [27]	✓	✓	×	✓	×	MK-HE
Kumbhar and Rao [4]	✓	×	×	✓	×	MK-HE
Pham et al. [28]	✓	×	×	✓	×	×
Li et al. [29]	×	✓	✓	✓	×	HE
Walskaar et al. [30]	✓	✓	×	✓	×	MK-HE
Fang and Qian [6]	✓	×	×	✓	✓	PHE
Our work	✓	✓	✓	✓	✓	HE

Notes: HE, PHE, SHE, and MK-HE denote Homomorphic Encryption, Privacy-Preserving Aggregation Mechanism, Somewhat Homomorphic Encryption, and Multi-Key Homomorphic Encryption, respectively. MKHE+SA represents a hybrid mechanism combining Multi-Key Homomorphic Encryption with Secure Aggregation. NHE refers to Non-Homomorphic Encryption, which provides ciphertext-level protection but does not support direct computation over encrypted data. The symbol × indicates that no explicit cryptographic primitive is used or specified.

In summary, based on the work of Hariss and Noura [23], we have developed a short-ciphertext lightweight multi-key privacy-preserving aggregation mechanism suitable for federated learning. Compared with the aforementioned work, this algorithm features a lightweight ciphertext length and a moderate computational cost, and can be easily combined with other federated learning lightweight methods to further reduce communication overhead. Meanwhile, by designing a key agreement algorithm, our system avoids introducing additional trusted parties, can operate under the classic FedAvg architecture, and maintains its simplicity.

3 Proposed Method

This section first provides an overview of the proposed federated learning method. Subsequently, we introduce two modules proposed in this work that deviate from the traditional federated learning framework, including the proposed privacy-preserving aggregation mechanism and the key agreement protocol designed to integrate the encryption scheme into the federated setting. In this process, we analyze the communication cost and security of the proposed method. As a relatively large number of symbols are used in this paper, the

tables listing the symbols employed in this and subsequent sections are provided in Appendix B, organized into several tables according to the respective modules to which the symbols belong.

3.1 Threat Model

During the training process of federated learning, we assume that the system operates under the Honest-but-Curious threat model, also known as the Semi-honest model, and we take potential internal collusion attacks into account. Specifically, the assumptions regarding the entities and the adversarial capabilities are defined as follows. We assume that the server is semi-honest. This means that the server will strictly follow the protocol specification and honestly execute all prescribed aggregation computations and communication procedures, and will not intentionally discard messages, tamper with computation results, or launch denial-of-service attacks. However, the server is curious in that it may attempt to infer the original private data of individual participants by exploiting all underlying data collected during the execution of the protocol, including the received ciphertext streams, public parameters, and intermediate computational views in the aggregation process. We assume that the participants in the system consist of honest, curious nodes. Curious also follows the apparent protocol flow, but they may collude with each other. Specifically, some malicious participants, the exact number of which will be discussed later in the security analysis, may secretly collude by sharing their private inputs, internal states, local masks, and key information, in an attempt to recover the private data of other honest participants through algebraic solving or statistical analysis. We assume that the number of Curious is bounded by the system security threshold. Finally, the system assumes that the underlying network connections used for transmitting highly sensitive information among all entities, such as key negotiation and initial parameter configuration during the initialization phase, are established over standard secure channels, for example, mutually authenticated links based on TLS/SSL (Transport Layer Security/Secure Sockets Layer).

3.2 Federated Learning Method with Multi-Key Encryption

This subsection presents the overall method of our proposed efficient federated learning algorithm. Our overall federated learning system and the corresponding information transmission steps are illustrated in Fig. 1. The entire training process consists of four main components: initialization, local training, key agreement, and aggregation. The arrows in the figure illustrate the communication steps involved throughout the entire process. Each component is described in detail as follows:

Initialisation: The server firstly generates the initial global model and secret key, and then distributes them to clients as indicated by line ①. Each client performs key expansion locally.

Local training: Each participant conducts a certain number of rounds of training using the local dataset. After training, the gradients are extracted, quantized, and encrypted to produce ciphertexts. These ciphertexts, along with the aggregated key shares, are then transmitted to the server, as illustrated in line ②.

Key negotiation: Each client refreshes a subset of its subkeys using a local random state and packages its subkeys into shares based on the share generation mechanism. As illustrated in line ③, the clients then exchange the corresponding shares with each other. After the exchange, the shares are aggregated accordingly.

If the number of participants in federated learning exceeds the threshold n_{agg}, the server randomly divides the participants into multiple groups, ensuring that each group contains at least two participants. Key negotiation is then conducted independently within each group.

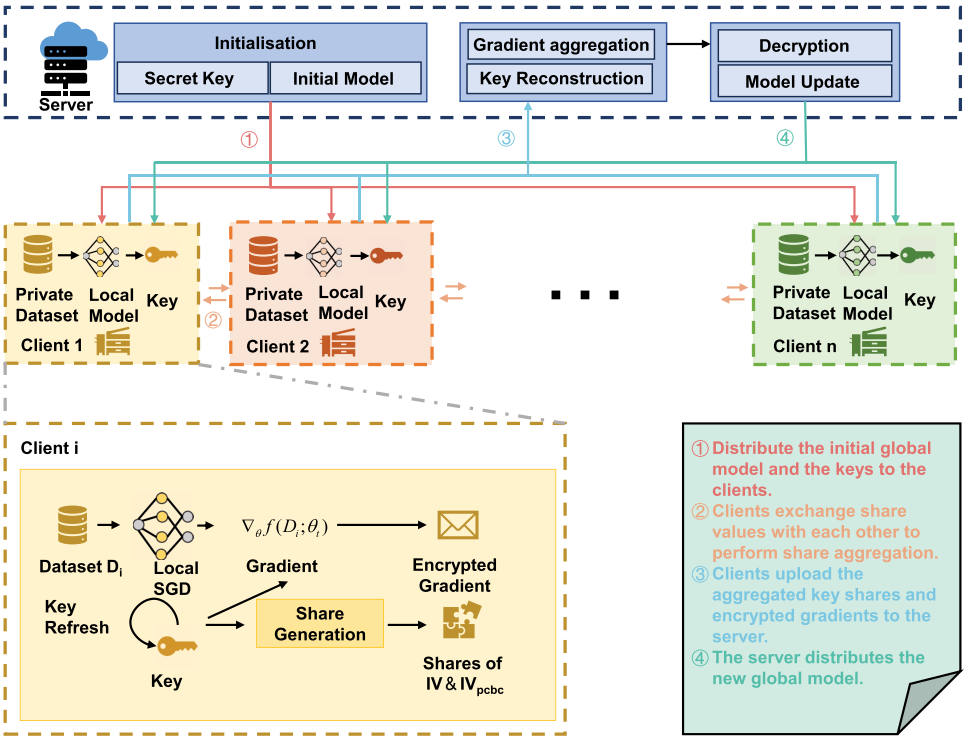


Figure 1: The functional modules of system components and the overall training workflow.

Aggregation: The server first reconstructs the decryption key from the aggregated key shares, aggregates the encrypted gradients, and then performs decryption. Finally, the model is updated, and the new global model is sent to the clients selected for the next round, as shown in ④.

If there are multiple groups in the current training process, decryption is first performed within each group, and the ciphertexts are then aggregated again.

Initialisation is only performed once, while the other three steps are repeated until the training is complete. Since the gradients are already encrypted, they can be transmitted directly. However, to prevent potential eavesdropping, the initial key distribution and negotiation are supposed to be conducted over secure communication channels.

The following subsections will provide a detailed description of our encryption algorithm and key negotiation protocol, along with an analysis of the method's security and communication efficiency.

3.3 Multi-Key Privacy-Preserving Mechanism

This subsection presents the proposed MKPHE-FL algorithm. Its parameters can be tuned according to the requirements of federated learning to reduce ciphertext length. By incorporating a randomization mechanism and encryption modes into the privacy-preserving aggregation mechanism, the security of the mechanism is significantly enhanced. Moreover, the algorithm allows each client to encrypt data using a distinct initialization vector, provided that these vectors are properly aggregated during decryption.

3.3.1 Key Expansion and Refresh

This subsubsection describes how to generate the cryptographic keys required by the privacy-preserving aggregation mechanism based on the parameters and keys in federated learning.

First, define the following parameters: the number of clients in the federated learning system, n; the lower and upper bounds of the interval of integers to which the model's floating-point parameters are mapped, $[-\alpha, \alpha]$; the bound of the injected noise, $[-\beta, \beta]$; the block size of the ciphertext matrix, $size_{block}$; and the number of encryption rounds, n_r.

Set a single-group aggregation threshold n_{group} for the privacy-preserving aggregation mechanism, and set a minimal ciphertext space of the privacy-preserving aggregation mechanism as $N_{min} = 2n_{group}\alpha \times 2n_{group}\beta$. To reduce computational complexity and ciphertext length, choose a prime modulus N that is marginally greater than N_{min}. A key sk of at least 128 bits is generated and used as the input to a key derivation function (KDF), which produces a pseudorandom byte string of sufficient length. The required length of the byte string, denoted as l_{string}, is determined by the encryption parameters and is computed as shown in Eq. (1):

$$l_{string} = ((3n_r + 1) \times size_{block}^2) \times \lceil \log_2 N \rceil + l_\pi \tag{1}$$

where the number of matrices needed is determined by the encryption round count n_r, and the required key length l_π for generating the permutation matrix π depends on the specific permutation generation algorithm adopted. Specifically, for n_r rounds of encryption, the scheme requires n_r diffusion matrices (DM), n_r permutation matrices (π), and $n_r + 1$ round key matrices (RK). Each element of RK is selected from the nonzero elements of the ring.

In this work, the KSA algorithm proposed by Noura et al. [34], an enhanced variant of RC4, is employed for generating permutation boxes. The method by Hariss and Noura [35] is employed to construct permutation matrices that can be used to perform reversible permutation operations. The bit string is divided into substrings, which are then used to generate the individual subkeys. Once the matrices are generated, their corresponding inverse keys are also derived, and all cryptographic keys are stored in their respective lists for subsequent use.

After the completion of local training in each training round, just before the encryption phase begins, each client refreshes their initial vectors. They independently generate a sufficiently long pseudorandom bit string using the key sk and a locally maintained random state, denoted as $nonce$. This bit string is then parsed into $n_r + 1$ matrices for IV_{round}, one matrix for IV_{pcbc}, and two matrices for IV_{cbc}. These three components are respectively used for round keys, plaintext cipher block chaining (PCBC), and cipher block chaining (CBC). Collectively, these are referred to as the IV_s set. The complete key generation process is illustrated in Fig. 2. It consists of two parts: a one-time key expansion (upper part) and a key refresh (lower part) that is repeated in each round.

An example of the derived key and corresponding subkeys for the case where $N = 127$ and $size_{block} = 4$ is provided in Fig. 3. They are eventually encoded into subkey matrices. Each element of RK is multiplicatively invertible, DM is an invertible matrix over a ring, each element of π represents a non-repeating coordinate value, and each element of IV_s is additively invertible.

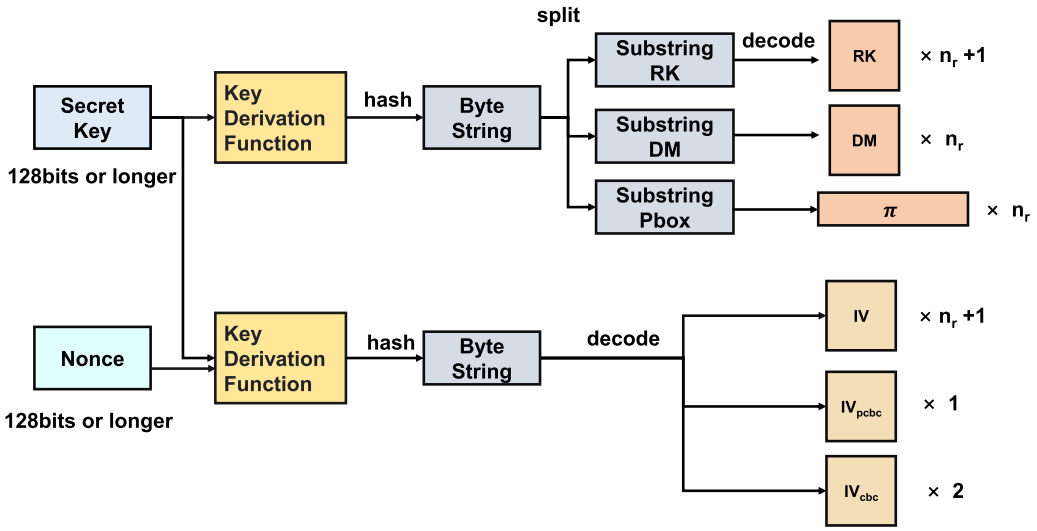


Figure 2: The process of expanding secret keys and nonce into subkeys.

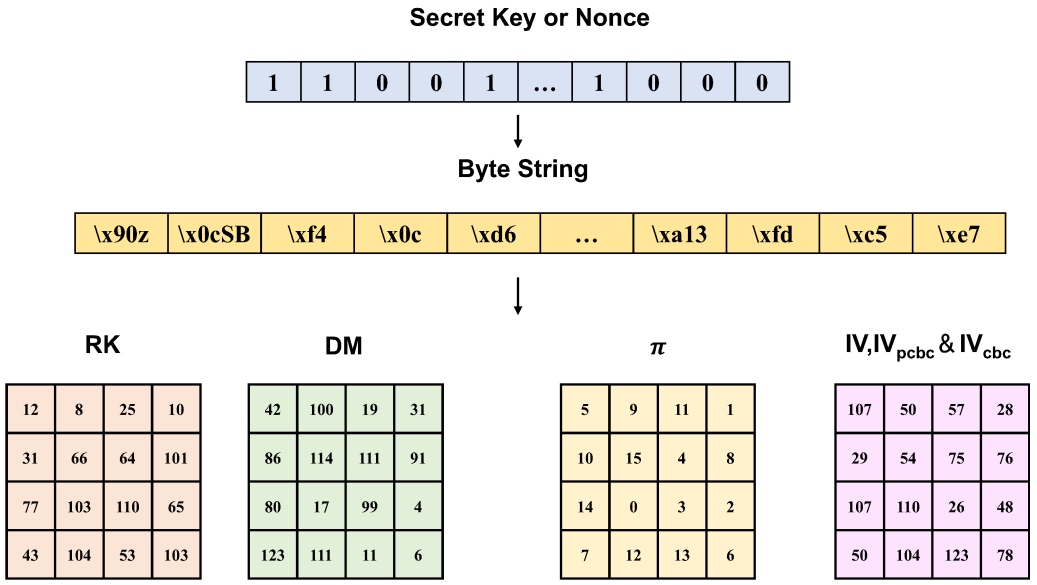


Figure 3: A numerical example of keys, nonce, bit strings, and subkeys for a given parameter.

3.3.2 Block Encryption with Multiple Encryption Modes

This subsubsection details the encryption procedure of the proposed algorithm. The extracted gradient data are first mapped onto the integers in the interval $[-\alpha, \alpha]$ to form the plaintext. In this work, the mapping is performed using the dACIQ [17] mechanism. Given a plaintext vector of length l_{array}, it is first partitioned into $l_{matrix} = \left\lceil l_{array}/size_{block}^2 \right\rceil$ matrices of size $size_{block} \times size_{block}$. Any remaining entries are padded with random elements uniformly sampled from the ring $\mathbb{Z}_N$ to complete the final matrix. After preprocessing, the input is represented as a list of l_{matrix} matrices: $[M_0, M_1, \ldots, M_{l_{matrix}-1}]$. Each matrix in the list is then encrypted one by one. The preceding matrices influence the subsequent ciphertexts through the encryption modes. To enhance the randomness of the encryption process and improve resistance against linear cryptanalysis, a randomization mechanism is introduced, as shown in Eq. (2).

$$C_{disturb,i} = M_i * scale + R_i \tag{2}$$

where each matrix M_i is masked with a randomly generated matrix R_i, $scale$ is chosen such that the homomorphic operation does not result in overflow. This constraint is satisfied by setting $scale = 2n\beta$. In order to improve the diffusion effect of the plaintext on the ciphertext and avoid specific patterns, the PCBC (Plaintext Cipher Block Chaining) mode like the AES algorithm is employed. With the IV_{pcbc} generated in the previous subsubsection,the plaintext M_i and ciphertext C_i from the previous phase are used to influence the ciphertext of the next phase. The concrete operator is shown in Eq. (3)

$$C_{pcbc,i} = \begin{cases} C_{disturb,i} + IV_{pcbc}, & i = 0 \\ C_{disturb,i} + C_{disturb,i-1} + C_{i-1}, & i > 0 \end{cases} \tag{3}$$

where $C_{disturb,i}$ denotes the ciphertext corresponding to the i-th plaintext matrix M_i after applying the disturbance mechanism, while C_{i-1} refers to the ciphertext of the $(i-1)$-th matrix M_{i-1} obtained through encryption. In the encryption of the first plaintext matrix, since there is no previous ciphertext available, the previously generated subkey IV_{pcbc} is used.

Following this, the initial round key addition As shown in Eq. (4) is performed using the round key (RK) and the initial vector (IV_{round}) indexed by $order_{init} = i \bmod (n_r + 1)$ from the key list. The formal encryption process consists of n_r rounds of encryption blocks then begins. In each round, three operations are sequentially executed: P-box permutation, column diffusion, and round key addition. These operations are formally defined in Eqs. (4)–(6).

$$C_{add,i} = M_i \odot RK + IV_{round} \tag{4}$$

$$C_{dm,i} = DM^{-1} \times M_i \times DM \tag{5}$$

$$C_{pi,i} = \pi(M_i) \tag{6}$$

where M_i denotes the plaintext input to the operator, $C_{add,i}$, $C_{dm,i}$, and $C_{pi,i}$ represent the corresponding outputs of the round key addition, diffusion matrix multiplication, and permutation operations, respectively. The operator $\odot$ denotes element-wise (Hadamard) multiplication between matrices, whereas $\times$ indicates standard matrix multiplication. The function $\pi(\cdot)$ denotes a permutation operation applied to the flattened version of M_i, where the element at position i is moved to the position specified by the permutation vector $\pi[i]$. When the number of encryption rounds reaches $\lceil \frac{n_r}{2} \rceil$, an additional layer of protection against differential attack on ciphertexts generated from a single encryption round is introduced by incorporating a second invocation of the CBC (Cipher Block Chaining) mechanism, inspired by the PCBC structure. This mid-stage CBC operation is formalized in Eq. (7)

$$C_{cbc,i} = \begin{cases} C'_i + IV_{cbc,1} + IV_{cbc,0} & i = 0 \\ C'_i + C'_{i-1} + IV_{cbc,1} & i = 1 \\ C'_i + C'_{i-1} + C^{t\prime}_{i-2} & i > 1 \end{cases} \tag{7}$$

where C'_{i-1} and C'_{i-2} represent the ciphertexts obtained from the immediately preceding and the second preceding encryption rounds, respectively. When the encryption process begins and no previous ciphertext is available, the previously generated key IV_{cbc} is used.

Upon completion of the CBC mechanism, the remaining $n_r - \lceil \frac{n_r}{2} \rceil$ encryption rounds are executed sequentially. At the conclusion of this process, the final ciphertext C_i corresponding to the current plaintext matrix M_i is produced. The encryption procedure then proceeds to the next plaintext matrix M_{i+1}. To

enhance the key sensitivity of the encryption scheme—thereby obfuscating the functional roles of individual subkeys and mitigating the risk of key inference—key rotation is employed throughout the encryption process. Specifically, for each plaintext matrix M_i, the index of the round key used for additive operations is computed as $order_a = (1 + i + round_{enc}) \mod (n_r + 1)$, where $round_e = 0, 1, \ldots, n_r - 1$ denotes the current encryption round. Similarly, the index of the key used for permutation and diffusion is given by $order_{dm} = order_\pi = (i + round_{enc}) \mod n_r$. This dynamic indexing strategy ensures that key reuse follows a non-trivial pattern, contributing to the cryptographic robustness of the algorithm. The complete encryption procedure is detailed in Algorithm 1.

Algorithm 1: Block encryption with multiple encryption modes

Require: Plaintext vector *Message* to be encrypted, key *sk* used, current system random state *nonce*
Ensure: The ciphertext vector *Ciphertext*, the initialisation vector IV_s used to encrypt this ciphertext

$IV_{round}, IV_{pcbc}, IV_{cbc} \leftarrow keyGen(sk, nonce)$
$[M_0, M_1, \ldots, M_{l_{matrix}-1}] \leftarrow Message$
for $i \leftarrow 0$ to l_{matrix} **do**
  $C_{disturb,i} \leftarrow disturb(M_i)$; $C_{pcbc,i} \leftarrow pcbc(C_{disturb,i}, IV_{pcbc}, C_{disturb,i-1}, C_{i-1})$
  $order_{init} \leftarrow i \mod (n_r + 1)$
  $C_{round,i} \leftarrow addRoundKey(C_{pcbc,i}, order_{init})$
  for $round_{enc} \leftarrow 0$ to $n_r - 1$ **do**
    $order_{dm} \leftarrow order_\pi \leftarrow (i + round_{enc}) \mod n_r$
    $C_{pi,i} \leftarrow permutate(C_{add,i}, order_{dm})$;
    $C_{dm,i} \leftarrow diffuse(C_{pi,i}, order_\pi)$
    $order_a \leftarrow (1 + i + round_{enc}) \mod (n_r + 1)$
    $C_{add,i} \leftarrow addRoundKey(C_{dm,i}, order_a)$
    if $round_{enc} = \lceil n_r/2 \rceil$ **then**
      $C_{round,i} \leftarrow cbc(C_{add,i}, IV_{cbc,0}, IV_{cbc,1}, C_{add,i-1}, C_{add,i-2})$
    else
      $C_{round,i} \leftarrow C_{add,i}$
    end if
  end for
end for
$Ciphertext \leftarrow C_0 \parallel C_1 \parallel \cdots \parallel C_{L_{matrix}}$

The decryption process strictly follows the inverse procedure of encryption. Upon receiving the ciphertext bitstream, the decrypting party first decodes the bitstream into vectors according to the encryption parameters determined during key initialization, and subsequently reconstructs it into a list of ciphertext matrices. The ciphertext list is then traversed in the same order as the encryption phase for decryption. During the decryption of each ciphertext block, multiple rounds of inverse operations are performed sequentially, including round key addition, substitution, and permutation. When the process reaches the $\left\lfloor \frac{n_r}{2} \right\rfloor$ round, the inverse CBC operation is executed, after which decryption continues with the same inverse sequence of round key addition, substitution, and permutation. Upon completing all rounds, a final round key addition is performed, followed by inverse PCBC and scaling operations to eliminate random perturbations, thereby recovering the original plaintext.

3.3.3 Correctness and Homomorphic Properties of the Proposed Algorithm

This subsubsection introduces the source of correctness and homomorphism in the proposed encryption algorithm.

In Section 3.3.1, we choose the modulus N of the ciphertext domain to be a prime number slightly greater than $2n\alpha \times 2n\beta$, which means that the entire ciphertext space is a finite field $\mathbb{F}_p$. This space supports addition, subtraction, multiplication, and division of its elements. All elements in this space possess additive inverses (the additive inverse of an element a is $-a \mod N$); therefore, the addition of IV, the applied random mask, and the CBC and PCBC mechanisms can all be reversed by adding the corresponding additive inverse. Each nonzero element in this space also has a multiplicative inverse, which means that the elements of RK all possess multiplicative inverses, and thus the adopted round-key addition operator is reversible. By employing the algorithm of Hariss and Noura [35], the invertibility of the matrix multiplication of the confusion matrix DM is also guaranteed. The invertibility of the permutation operation is not affected by the algebraic space and always holds. In addition, by properly configuring the parameters of random obfuscation, the impact of random bit overflow on ciphertext bits during decryption or intermediate computation can be effectively avoided. Ultimately, the correctness of the overall encryption and decryption process of the algorithm is guaranteed. Specifically, the invertibility of each encryption operator and the corresponding decryption methods are shown in Eqs. (8)–(12). Among them, Eq. (10) is performed over the real field, while the remaining operations are performed over the modular ring.

$$M_i \equiv (M_i \odot RK + IV_{round} - IV_{round}) \odot RK^{inv} \mod N \tag{8}$$

$$M_i \equiv DM \times DM^{-1} \times M_i \times DM \times DM^{-1} \mod N \tag{9}$$

$$M_i \equiv \lfloor (M_i * scale + R_i)/scale \rfloor \tag{10}$$

$$C_{disturb,i} = \begin{cases} (C_{disturb,i} + IV_{pcbc}) - IV_{pcbc} \mod N, & i = 0 \\ (C_{disturb,i} + C_{disturb,i-1} + C_{i-1}) - C_{disturb,i-1} - C_{i-1} \mod N, & i > 0 \end{cases} \tag{11}$$

$$C'_i = \begin{cases} (C'_i + IV_{cbc,1} + IV_{cbc,0}) - IV_{cbc,1} - IV_{cbc,0} \mod N, & i = 0 \\ (C'_i + C'_{i-1} + IV_{cbc,1}) - C'_{i-1} + IV_{cbc,1} \mod N, & i = 1 \\ (C'_i + C'_{i-1} + C'_{i-2}) - C'_{i-1} + C'_{i-2} \mod N, & i > 1 \end{cases} \tag{12}$$

Leveraging the associativity of addition over a ring and the distributive properties of both scalar and matrix multiplication, the round key addition and column mixing steps in the encryption process inherently preserve additive homomorphism. Moreover, the permutation operation, being a position-based transformation that does not alter the actual values of ciphertext elements, also maintains homomorphic properties, provided that an identical subkey is used across ciphertexts. Consequently, it becomes feasible to perform plaintext addition within the encrypted domain simply by executing additive operations on the corresponding ciphertexts. Following encryption, as long as the sum of the original plaintexts remains within the bounds of the minimal plaintext space $2n\alpha$, the decryption procedure can accurately recover the plaintext sum. During homomorphic addition, the round key (RK) and diffusion matrix (DM) components naturally combine without additional processing. In contrast, the subkey IV_s does not directly merge; rather, it accumulates incrementally across successive homomorphic additions, as illustrated in Eqs. (13)–(15).

$$\sum C_i = \sum M_i \odot RK + \sum IV_i \tag{13}$$

$$\sum C_{pcbc,i} = \begin{cases} \sum C_{disturb,i} + \sum IV_{pcbc}, & i = 0 \\ \sum (C_{disturb,i} + C_{disturb,i-1} + C_{i-1}), & i > 0 \end{cases} \tag{14}$$

$$\sum C_{cbc,i} = \begin{cases} \sum C'_i + \sum IV_{cbc,0} + \sum IV_{cbc,1}, & i = 0 \\ \sum C'_i + \sum IV_{cbc,1} + \sum C'_{i-1}, & i = 1 \\ \sum (C'_i + C'_{i-1} + C'_{i-2}), & i > 1 \end{cases} \tag{15}$$

where the symbols is consistent with their previous definitions.

For the entire encryption algorithm, its homomorphic property can be referred to in the equation in the Appendix A.1.2. For multiple participants using the same public key, the subkeys derived from the public key are identical, and the generated A_i^M and A_i^R are also the same. Therefore, the overall homomorphic addition of the ciphertext can be transformed into the following form.

$$\sum \vec{C}_i = A_i^M \begin{bmatrix} \sum \vec{M}_{scale,i} \\ \sum \vec{M}_{scale,i-1} \\ \sum \vec{M}_{scale,i-2} \end{bmatrix} + A_i^R \begin{bmatrix} \sum (\vec{R}_i + \vec{R}_{i-1}) \\ \sum (\vec{R}_{i-1} + \vec{R}_{i-2}) \\ \sum (\vec{R}_{i-2} + \vec{R}_{i-3}) \end{bmatrix} + A_i^{IV} \begin{bmatrix} \sum \vec{IV}_{i,0} \\ \sum \vec{IV}_{i-1,0} \\ \sum \vec{IV}_{i-2,0} \end{bmatrix} \tag{16}$$

Therefore, a key negotiation step is required following homomorphic operations to ensure the correct decryption of the aggregated result. On the other hand, this characteristic also enables each client to utilize a distinct subkey IV_s for encryption. As long as the server ultimately receives the correctly aggregated version of the subkeys, it can successfully decrypt the final ciphertext.

3.4 Key Negotiation of Initialization Vectors

The previous subsection introduced the improved encryption algorithm proposed in this paper. However, as observed in the decryption phase, it is necessary to aggregate the different IV_s values after homomorphic encryption in order to correctly decrypt the ciphertext. Given the presence of curious servers and potential external eavesdroppers, directly transmitting each participant's initialization vector for aggregation is not viable within the federated learning method. In this subsection, a key negotiation protocol based on secret sharing (SS-KNP) is constructed to address this issue.

Secret sharing allows a secret to be divided into n parts and distributed to n different entities. The original secret can only be reconstructed when at least t of these parts are combined. This implies that an adversary must collude with more than t clients in the federated learning system to successfully recover another client's secret.

The security of Shamir's secret sharing scheme relies on the uniqueness of the solution to a system of linear equations or equivalently, the uniqueness of the coefficients in polynomial interpolation. In this scheme, the secret S is embedded as a coefficient in a randomly generated polynomial $f(x)$ of degree $t-1$. The values of this polynomial at distinct points $X = \{x_1, x_2, \ldots, x_n\}$ are computed and distributed as secret shares to individual entities. The secret can later be reconstructed by solving for the coefficients using Lagrange interpolation.

When applying secret sharing for key agreement, the entire protocol consists of three phases: share generation, aggregation, and reconstruction of the secret.

Generation the Sharing of Subkeys: For the aforementioned privacy-preserving aggregation mechanism, the secret that needs to be shared and aggregated is the initial vector IV_s in the subkeys. Flatten and concatenate the IV_s to form a vector $S = [s_0, s_1, \ldots, s_{l_s}]$ containing $l_{secret} = (n_r + 4) \times size_{block}^2$ secret values. During each sharing, t coefficients can be packed into a single polynomial value as the coefficients of a $t-1$ degree polynomial, e.g., $f(x, s) = s_0 + s_1 x + s_2 x^2 + \ldots + s_{t-1} x^{t-1}$. When the polynomial degree is high, there is a considerable risk that the secret generation and reconstruction operations will overflow, so all operations are performed in the ring $\mathbb{Z}_N$.

Key Aggregation: Once all clients have shared their secrets, the j-th client's corresponding share $shares_j = f(x_j, s)$ is sent to the corresponding client. Finally, each client uses the received polynomial value to complete the aggregation. The theoretical basis for this operation is the homomorphism property of Shamir secret sharing, i.e., when different clients use the same set of random number coordinates $X = \{x_1, x_2, \ldots, x_n\}$. For client i, all the polynomials received by it follow Eq. (17)

$$\sum_{j=0}^{n-1} f(x_i, s^j) = \sum_{j=0}^{n-1} \sum_{k=0}^{t-1} s_{j,k} x_i^k = \sum_{k=0}^{t-1} \Big(\sum_{j=0}^{n-1} s_{j,k} \Big) x_i^k = f\Big(x_i, \sum_{j=0}^{n-1} s^j \Big) \tag{17}$$

where i and j represent the designated recipient and sender of the share, respectively. Therefore, for multiple clients using the same set of X, each client i can add up all the shared values it receives to get the aggregated shared value through $shares_{agged,i} = \sum_{j=0}^{n-1} share_{j,i}$.

Reconstruction of the Aggregated Key: After the sharing and aggregation are complete, the aggregated and sufficient sharing values are sent to the required entity (server) for secret reconstruction. This process is typically performed using the Lagrange interpolation polynomial. By collecting at least t share values, constructing the Lagrange interpolation polynomial with them, and combining the coefficients of terms of the same degree, the coefficients can be extracted to reconstruct the aggregated IV_s. Since the threshold t can be smaller than the total number of shares n, our method is theoretically capable of flexibly adjusting the threshold according to practical scenarios, such as handling potential participant dropouts. For example, by discarding offline participants and retaining only those with stable communication to allow the negotiation to proceed normally, rather than aborting the entire training round due to a few participants dropping out. However, this is not the main focus of this work and is therefore not explicitly reflected in the training process. This step is performed after each client sends the aggregated shares to the server, which then executes the reconstruction operation to complete the process.

3.5 Security Analysis

In this subsection, the security of the proposed method based on the security properties of the privacy-preserving aggregation mechanism and the secret sharing protocol is analysed. By examining the information generated and exchanged by each participant throughout the federated learning process, it has been demonstrated that the proposed method effectively preserves the privacy of individual clients.

Lemma 1: *For the proposed privacy-preserving aggregation mechanism, even if an entity has access to a set of plaintext-ciphertext pairs, as well as knowledge of the key sk and its derived keys RK, DM, and π, it remains computationally infeasible to infer the initial vectors IV_s used by other participants or to recover any unknown plaintexts.*

Proof: See Appendix A.3. □

Lemma 2: *For any subset of fewer than t participants in a Shamir secret sharing scheme, the shares they obtain provide no statistical advantage in inferring the secret.*

Proof: This property has been formally proven in Shamir's original work [36]. □

First, we present the views observable to the server and the colluding participants: Let Π denote the aggregation protocol proposed by us. During the execution of Π, all the information observable to the privacy-stealing party constitutes its "view." Below, by analyzing the execution process of the framework, we present the views of the server and the colluding participants.

Initialization: During the initialization phase, the server generates a system-wide shared key sk and securely distributes it to each client. The global model θ_0 is generated and distributed to each client. Since both the public key and the global model are randomly generated, no private information is leaked.

Local training: In the local training phase, no information is transmitted, so naturally, no information leakage occurs. During the encryption and uploading process, each client's local random state and the generated initial vector are both randomly instantiated. According to Lin et al. [37], the gradients generated during deep learning training tend to be sparse and contain a large number of zero entries. This sparsity may leak additional information to an adversary within the system regarding the plaintext corresponding to a given ciphertext, which could potentially be exploited to construct a known-plaintext attack.

Key negotiation: In this phase, each client packages its own IV_s (initial vector) information as a secret and generates shares to be sent to other clients. Without collusion, each client holds its own shares and the secret share values received from other participants corresponding to is identity.

Aggregation: The server can acquire the local encrypted weighted gradients $[[w_i g_i]]$ from each participant, their sum $[[\sum_{i=1}^{n_{select}} w_i g_i]]$, and the final decrypted gradient. As demonstrated in the multi-key encrypted federated learning framework proposed by Ma et al. [10], the server's acquisition of aggregated parameters does not result in the privacy leakage of participants. Since the privacy-preserving aggregation mechanism itself is key-sensitive, the server cannot obtain gradient information through incorrect decryption without knowledge of the IV_s.

In summary, we can characterize the views (leakage assumptions) of an honest-but-curious server and a subset of participants who are also honest-but-curious but may engage in collusion among themselves, when the number of participants has not yet reached the level required for group aggregation. The server $\mathcal{S}$ strictly follows the execution of protocol Π, but attempts to infer the unknown true gradients from all the data it collects. The view of the server, $View_{\mathcal{S}}^{\Pi}(\vec{g})$, is defined as the complete set of its "internal state" and "observed information" during the execution of the protocol:

$$View_{\mathcal{S}}^{\Pi}(\vec{g}) = (\mathcal{R}_{\mathcal{S}}, Params_{pub}, \mathcal{M}_{\mathcal{S}}^{recv}, \mathcal{S}^{comp}) \tag{18}$$

where each component is specified as follows:

$\mathcal{R}_{\mathcal{S}}$: the internal randomness generated by the server during the execution of the protocol. $Params_{pub}$: the aforementioned globally public training context and key information. $\mathcal{M}_{\mathcal{S}}^{recv}$: the set of messages received by the server from all clients $i \in \{1, 2, \ldots, n\}$, including the uploaded encrypted gradients and aggregated key shares:

$$\mathcal{M}_{\mathcal{S}}^{recv} = \{([[w_i g_i]], shares_{agg,i})\}_{i=1}^{n} \tag{19}$$

$\mathcal{S}^{comp}$: the intermediate states produced by the server during the aggregation computation phase and the final decryption results:

$$\mathcal{S}^{comp} = \left(\sum_{i=1}^{n}[[w_i g_i]], \sum_{i=1}^{n} IV_i, \sum_{i=1}^{n} g_i\right) \tag{20}$$

Let $G_{coll} \subset \{1, 2, \ldots, n\}$ denote the set of Curious that collude among the participants (with $|G_{coll}| < t$, that is, the number of colluding participants does not reach the reconstruction threshold of secret sharing). The situation is different when the number of participants exceeds n_agg and grouping is required; this case will be discussed separately in the proof. It is assumed that this set does not collude with the server. Moreover, the protocol stipulates that all clients will receive the globally decrypted gradient $\sum_{i=1}^{n} g_i$ after aggregation is completed. The overall view of the colluding set $View_{G_{coll}}^{\Pi}(\vec{g})$ is defined as the union of the individual views of all colluding members:

$$View^{\Pi}_{G_{coll}}(\vec{g}) = \bigcup_{k \in G_{coll}} View^{\Pi}_{k}(\vec{g}) = (\mathcal{R}_{coll}, Params_{pub}, \mathcal{D}^{self}_{coll}, \mathcal{M}^{recv}_{coll}, Out_{agg}) \tag{21}$$

where each component is specifically expanded as follows:

$\mathcal{R}_{coll}$: the set of internal randomness of all colluding participants, $\bigcup_{k \in G_{coll}} \mathcal{R}_k$. $Params_{pub}$: the globally public parameters. $\mathcal{D}^{self}_{coll}$: all private data generated by the colluding subset itself and shared internally:

$$\mathcal{D}^{self}_{coll} = \{g_k, [[w_k g_k]], IV_{s,k}, share_{k,i} | k \in G_{coll}, i \in \{1, 2, \ldots, n\}\}. \tag{22}$$

Here, $IV_{s,k}$ and $share_{k,i}$ are the keys and shares generated by the colluding participants themselves.

$\mathcal{M}^{recv}_{coll}$: all key share fragments received by the colluding parties during the key negotiation and sharing phase:

$$\mathcal{M}^{recv}_{coll} = \{share_{i,j} \mid i \in \{1, \ldots, n\},\ j \in G_{coll}\}. \tag{23}$$

Out_{agg}: the global plaintext aggregation result received by the colluding parties from the server in the final stage of the protocol, namely $\sum_{i=1}^{n} g_i$.

Definition 1 (Privacy against Honest-but-Curious Server): *Let Π be the aggregation protocol proposed in this paper, n be the total number of participants, and $\vec{g} = (g_1, g_2, \ldots, g_n)$ be the true local gradients of all participants. We say that the protocol Π is computationally secure against an honest-but-curious server $\mathcal{S}$ if there exists a probabilistic polynomial-time (PPT) simulator $\mathcal{SIM}_{\mathcal{S}}$ such that, given the globally public parameters $Params_{pub}$ and the final legitimate output (i.e., the aggregated gradient $\sum_{i=1}^{n} g_i$), the simulated view generated by $\mathcal{SIM}_{\mathcal{S}}$ is computationally indistinguishable from the server's view in the real execution of the protocol, denoted by $View^{\Pi}_{\mathcal{S}}(\vec{g})$. That is, for any security parameter λ, it holds that*

$$\mathcal{SIM}_{\mathcal{S}}(1^{\lambda}, Params_{pub}, \sum_{i=1}^{n} g_i) \approx_s View^{\Pi}_{\mathcal{S}}(\vec{g}). \tag{24}$$

Definition 2 (Privacy against < t Colluding Participants): *Let $G_{coll} \subset \{1, 2, \ldots, n\}$ be the set of colluding participants, satisfying $|G_{coll}| < t$. Assume that this set does not collude with the server, and that all participants receive the final aggregation result. We say that the protocol Π can securely resist internal collusion attacks by fewer than t participants if there exists a probabilistic polynomial-time (PPT) simulator $\mathcal{SIM}_{coll}$ such that, given only the globally public parameters $Params_{pub}$, the set of true private data of the colluding parties $\mathcal{D}^{self}_{coll}$, and the final legitimate aggregation result $\sum_{i=1}^{n} g_i$, the simulated view generated by $\mathcal{SIM}_{coll}$ is computationally indistinguishable from the view of the colluding set in the real world, denoted by $View^{\Pi}_{G_{coll}}(\vec{g})$. That is, for any security parameter λ, it holds that*

$$\mathcal{SIM}_{coll}(1^{\lambda}, Params_{pub}, \mathcal{D}^{self}_{coll}, \sum_{i=1}^{n} g_i) \approx_c View^{\Pi}_{G_{coll}}(\vec{g}). \tag{25}$$

Theorem 1: *Under the assumption that the proposed privacy-preserving aggregation mechanism satisfies IND-CPA security, the protocol Π is computationally secure against the honest-but-curious server $\mathcal{S}$.*

Proof: To prove Definition 1, we construct a probabilistic polynomial-time (PPT) simulator $\mathcal{SIM}_{\mathcal{S}}$. This simulator takes only the public parameters $Params_{pub}$ and the final aggregation result $G_{sum} = \sum_{i=1}^{n} g_i$ as input, and forges a simulated server view $View^{sim}_{\mathcal{S}}$ as follows.

Construction of the forged view by $\mathcal{SIM}_{\mathcal{S}}$.

Internal state and public parameters: $\mathcal{SIM}_{\mathcal{S}}$ generates completely uniform random coins as the forged $\mathcal{R}'_{\mathcal{S}}$, and directly copies $Params_{pub}$.

Forgery of the received messages $(\mathcal{M}_{\mathcal{S}}^{recv})$ *and the computation state* $(\mathcal{S}^{comp})$*:*

Forged aggregated key shares: $\mathcal{SIM}_{\mathcal{S}}$ randomly generates a sum of IV'_s for the constant term satisfying the threshold requirement, and based on this, uses the corresponding polynomial to generate n random aggregated key shares $shares'_{agg,i}$.

Forged initialization vectors: According to the generated $shares'_{agg,i}$, it computes the corresponding forged sum of initialization vectors $\sum IV'_i$.

Forged encrypted gradients: This is the key step. $\mathcal{SIM}_{\mathcal{S}}$ cannot know the true g_i. Therefore, over the modulus-N ring $\mathbb{F}_N$, it independently and randomly generates the first $n-1$ participants' ciphertexts $C'_1, C'_2, \ldots, C'_{n-1}$ using a uniform random number generator.

To ensure consistency of the final decryption result, $\mathcal{SIM}_{\mathcal{S}}$ uses the known final result G_{sum} and the forged mask (determined by $\sum IV'_i$), and then forcibly computes the last ciphertext by reverse algebraic computation:

$$C'_n = \sum \mathrm{C}_i - \sum_{i=1}^{n-1} C'_i. \tag{26}$$

The ciphertexts C'_i are then used as the forged $[[w_i g_i]]'$, and are assembled into $\mathcal{M}_{\mathcal{S}}^{recv}$ and $\mathcal{S}^{comp}$.

Security analysis.

In the real execution, according to the security of the privacy-preserving aggregation mechanism proved in Appendix A.3, each true ciphertext uploaded by a participant, $[[w_i g_i]]$, is computationally indistinguishable from a random value over $\mathbb{F}_N$. In the simulated execution, $\mathcal{SIM}_{\mathcal{S}}$ also generates C'_i by random sampling from the uniform distribution. Since the sum of N uniformly distributed variables under a fixed-sum constraint still has a uniform marginal distribution, the statistical distance (SD) between the forged ciphertext set and the real ciphertext set is at most a negligible function ϵ.

Meanwhile, the forged key shares satisfy the same algebraic structure as in the real world. Since IV itself is sampled from a uniform random distribution, it is indistinguishable from the randomly generated IV'. Therefore, the simulated view output by $\mathcal{SIM}_{\mathcal{S}}$ is statistically indistinguishable from the real view $View_{\mathcal{S}}^{\Pi}(\vec{g})$, that is,

$$View_{\mathcal{S}}^{sim} \approx View_{\mathcal{S}}^{\Pi}(\vec{g}). \tag{27}$$

Thus, Theorem 1 is proved. □

Theorem 2: *Under the assumption that the underlying secret sharing scheme (e.g., Shamir Secret Sharing) satisfies threshold security and the proposed privacy-preserving aggregation mechanism satisfies IND-CPA security, the protocol* Π *can still securely resist any internal collusion attack among participants with* $|\ G_{coll}\ | < t$*, even when the final aggregated model is broadcast to all clients.*

Proof: Analogously to the above, we construct a probabilistic polynomial-time (PPT) simulator $\mathcal{SIM}_{coll}$. In the ideal world, this simulator acts as a "forger" and takes only the public parameters $Params_{pub}$, the colluders' private data $\mathcal{D}_{coll}^{self}$, and the final correct output obtained from the ideal functionality, namely $G_{sum} = \sum_{i=1}^{n} g_i$, as input. $\mathcal{SIM}_{coll}$ then forges the colluding parties' view $View_{G_{coll}}^{sim}$ step by step as follows.

Construction of the forged view by $\mathcal{SIM}_{coll}$.

Internal state and preservation of local data: $\mathcal{SIM}_{coll}$ honestly generates the colluders' internal randomness $\mathcal{R}'_{coll}$, the global parameters $Params_{pub}$, and their own true private data $\mathcal{D}^{self}_{coll}$.

Embedding the final output (Out_{agg})*:* $\mathcal{SIM}_{coll}$ directly uses the final aggregation result G_{sum} returned by the ideal functionality as the forged Out'_{agg} in the simulated view, namely

$$Out'_{agg} = \sum_{i=1}^{n} g_i. \tag{28}$$

Forging the received key shares ($\mathcal{M}^{recv}_{coll}$)*:*

In the real protocol, the colluding parties receive key shares $share_{i,j}(j \in G_{coll})$ sent by the honest participants $i \notin G_{coll}$. Since $\mathcal{SIM}_{coll}$ does not know the honest participants' true keys or local blinding factors, it cannot generate these shares according to the real algorithm. Therefore, $\mathcal{SIM}_{coll}$ directly performs uniform random sampling over the corresponding finite field $\mathbb{F}_q$, generates purely random values $r_{i,j} \leftarrow U(\mathbb{F}_q)$, and inserts them into $\mathcal{M}^{recv\prime}_{coll}$ as forged key shares $share'_{i,j}$ (For the shares exchanged internally among the colluding parties, $\mathcal{SIM}_{coll}$ computes them honestly according to the protocol using the known $\mathcal{D}^{self}_{coll}$ and $\mathcal{R}'_{coll}$).

Security analysis:

In the real execution of the protocol, each honest participant $i \notin G_{coll}$ constructs a polynomial $f_i(x)$ using a Shamir secret sharing scheme with reconstruction threshold t. According to the perfect information-theoretic secrecy of this scheme, for any polynomial of degree $t-1$, when the number of given evaluation points is strictly less than t (i.e., $\mid G_{coll} \mid < t$), the resulting sequence of fewer than t function values is completely equivalent to a mutually independent uniformly random vector over the field $\mathbb{F}_q$. In scenarios with a large number of participants that require grouping, as long as the threshold is set consistently with the upper bound on the number of members in a single group, colluding participants cannot reconstruct the key from the secret sharing scheme. The key can only be reconstructed if the entire group consists of colluding participants, in which case such reconstruction is meaningless.

This implies that:

The real $\mathcal{M}^{recv}_{coll}$ in the view is perfectly uniformly distributed over the finite field, and the forged $\mathcal{M}^{recv\prime}_{coll}$ in the simulated view is also generated by uniform sampling. Hence, the two are not only computationally indistinguishable, but in fact identically distributed. Moreover, Out_{agg} in the real view and in the simulated view are exactly the same final aggregation result G_{sum}. The real key shares (which determine the blinding masks) are algebraically independent of the final plaintext output G_{sum}, since the blinding masks are canceled out during aggregation through summation. Therefore, even if G_{sum} is included in the view, the independent randomness of the key shares is not affected.

Finally, even if the colluding participants eavesdrop on the ciphertexts uploaded by other participants, the security in this case can still be established through a simulation argument similar to that for the server, due to the computational indistinguishability of the privacy-preserving aggregation mechanism.

In summary, the joint distribution generated by the simulator,

$$\mathcal{SIM}_{coll}\Big(\mathcal{R}'_{coll}, Params_{pub}, \mathcal{D}^{self}_{coll}, \mathcal{M}^{recv\prime}_{coll}, Out'_{agg}\Big), \tag{29}$$

is perfectly indistinguishable from the colluding set's view $View^{\Pi}_{G_{coll}}(\vec{g})$ in the real world. This proves that when $\mid G_{coll} \mid < t$, or when participants perform group aggregation with and $t = n_{agg}$ even if the colluding

parties obtain the final model update (i.e., the aggregated gradient) as well as partial key fragments from honest nodes, they still cannot reconstruct any additional information about the gradient of any individual honest participant $g_i (i \notin G_{coll})$. Thus, Theorem 2 is proved. □

Theorem 3: *Assume that the underlying encryption algorithm satisfies IND-CPA computational indistinguishability, and that the secret sharing scheme (e.g., Shamir Secret Sharing) achieves perfect secrecy with fewer than t shares. Then, under the honest-but-curious model, the protocol* Π *not only guarantees the privacy of each individual gradient* g_i *against the centralized server, but also resists any internal collusion attack by participants satisfying* $| G_{coll} | < t$ *or participants performing group aggregation with and* $t = n_{agg}$*. That is, no unauthorized entity can extract any useful information about the local gradient* g_i *of any honest node* $i \notin G_{coll}$ *from its view with non-negligible probability.*

Proof: The conclusion of Theorem 3 follows directly from Theorems 1 and 2.

Specifically, Theorem 1 shows that, under the assumption that the underlying encryption algorithm satisfies IND-CPA security, the view of the honest-but-curious server $\mathcal{S}$ can be simulated from the public parameters and the final aggregation result alone, and is therefore statistically indistinguishable from the real view. Hence, the server cannot obtain any additional information about any individual participant's local gradient g_i beyond the legitimate output.

Meanwhile, Theorem 2 shows that, under the threshold security of the underlying secret sharing scheme, for any colluding set satisfying $| G_{coll} | < t$, or in the case where the number of participants exceeds n_{agg} and $t = n_{agg}$, the joint view of the colluding participants can also be perfectly simulated given only the public parameters, the colluders' own private data, and the final aggregation result. Therefore, such colluding participants cannot recover any additional information about the local gradient g_i of any honest participant $i \notin G_{coll}$.

Combining the above two results, it follows that, under the honest-but-curious model, the protocol Π simultaneously guarantees privacy against the centralized server and security against internal collusion by fewer than t participants. Therefore, no unauthorized entity can extract any useful information about the local gradient of any honest participant from its view with non-negligible probability. Thus, Theorem 3 is proved. □

In the above proof, we presented the security analysis of the federated learning method against an honest-but-curious server and colluding participants, based on the security of the privacy-preserving aggregation mechanism and the security of secret sharing. When the number of participants is large, the proposed federated learning method groups the participants. After grouping, the view observed by the server remains essentially unchanged; therefore, the above conclusion still holds. For colluding participants, since each group conducts key negotiation and aggregation independently, colluding participants from different groups cannot form effective cooperation. Within a single group, as long as the secret-sharing threshold t is set to the number of participants in the group/the maximum number of participants in the group, it can be guaranteed that the key cannot be stolen unless all participants in that group are colluding participants (if all participants in a group are colluding participants, stealing the key is meaningless anyway). Since the colluding participants do not collude with the server, they can only receive the decrypted aggregated gradients. In this case, the tolerance upper bound of the entire system to the number of colluding participants can reach $n - 2$.

Since the participants' original gradients are not leaked, the framework can effectively prevent attacks that aim to steal participants' privacy, such as gradient inversion attacks. However, since no screening mechanism is incorporated into the framework, it may be unable to defend against attacks intended to disrupt training.

3.6 Communication Efficiency Analysis

This subsection analyzes the overall communication cost after applying the proposed algorithm within the federated learning method. The communication efficiency of the proposed algorithm is influenced primarily by the quantisation precision bit_{quant} (corresponding to α), the noise scale β, as well as the configured maximum number of participants per group. Under symmetric quantisation, model parameters are mapped to the integer range $[1-2^{bit_{quant}-1}, 2^{bit_{quant}-1}-1]([-\alpha, \alpha])$. Here, the quantization precision bit_{quant} refers to the number of bits used to store and transmit the mapped integer parameters in the computer and $2^{bit_{quant}-1}-1$ refers to α mentioned earlier. Accordingly, the theoretical minimum ciphertext length required to encrypt a single quantised parameter is given by Eq. (30).

$$\begin{aligned} bit^{cipher} &= \log_2(N) \\ &= \log_2\left(2n_{group} \times \left(2^{bit_{quant}-1}-1\right) \times 2n_{group}\beta\right) \\ &= 2 + 2\log_2(n_{agg}) + \log_2\left(2^{bit_{quant}-1}-1\right) + \log_2(\beta) \end{aligned} \tag{30}$$

where bit^{cipher} represents the minimum ciphertext length required to encrypt a single parameter.

During the key negotiation process, within each group, the total number of times shares are sent is n_{group}^2, and the total number of keys that need to be packed is $n_r + 4$, namely $(n_r+4) \times size_{block}^2$ elements over the ring $\mathbb{Z}_N$. After being packed into sharing polynomials, the total number of generated shares is $\left\lceil \frac{(n_r+4)\times sizeblock^2}{t} \right\rceil$. Since the ciphertext domain used in secret sharing is the same as that used in the privacy-preserving aggregation mechanism, the communication cost generated by a single share is $comm_{share} = \left\lceil \frac{(n_r+4)\times size_{block}^2}{t} \right\rceil \times bit^{cipher}$, which is a constant when the encryption parameters are fixed. For different numbers of participants, the total number of communications among participants, denoted as num_{neg}, is shown in Eq. (31):

$$num_{neg} = \begin{cases} n^2 & \text{if } n \le n_{group} \\ \left\lfloor \frac{n}{n_{group}} \right\rfloor \times n_{group}^2 + (n \mod n_{agg})^2 & \text{if } n > n_{group} \end{cases} \tag{31}$$

The total communication cost during the negotiation process is: $comm_{neg} = (num_{neg} + n) \times comm_{share}$. Therefore, when the number of participants satisfies $n \le n_{group}$, the time complexity of the key negotiation part is $O(n^2)$; when the number of participants satisfies $n \ge n_{group}$, the communication cost generated by key negotiation is approximately $O(n)$.

For the federated learning process, assuming that the number of parameters in the global model is num_{para} and the training process involves a total of $round_{fed}$ communication rounds, the communication cost associated with transmitting the model throughout the training process is: $comm_{model} = (bit_{float} + bit^{cipher}) \times num_{para} \times round_{fed} \times n_{select}$.

Therefore, the total communication cost in federated learning is $comm_{model} + comm_{neg}$. Here, n_{select} denotes the number of clients actually selected in each round of the federated learning process, which is a variable parameter that accounts for a proportion of n. bit_{float} denotes the default number of bits for floating-point numbers in deep learning frameworks, which is typically 32.

Since, compared with the number of parameters in commonly used deep learning models, the number of parameters occupied by IV_s accounts for only a very small proportion. When the model architecture and communication protocol are fixed, the communication cost incurred by ciphertext transmission scales linearly with the quantisation precision and exhibits an $O(n)$ dependence on the number of clients n.

4 Experiment

In this section, we evaluate the performance of the federated learning framework in terms of accuracy, computational cost (time), and communication cost. The performance of the privacy-preserving aggregation mechanism is evaluated using metrics such as entropy and NPCR (Number of Pixels Change Rate). Specifically, we conduct horizontal performance comparisons and parameter sensitivity analyses for both the federated learning framework and the privacy-preserving aggregation mechanism, and further perform ablation experiments on the privacy-preserving aggregation mechanism. The code related to the experiment can be found at https://github.com/ignorance163/LHEFL_algorithm.

4.1 Baselines

In the first part, to illustrate the communication cost of the federated learning method and the computational cost of the privacy-preserving aggregation mechanism, we compare our approach (Our Work) with several baselines: FedAvg with direct plaintext transmission (Plaintext), FedAvg with quantized plaintext transmission (QuantPlaintext), variants of our method that use other homomorphic encryption schemes (BFV and CKKS), and MaskCrypt [15], a privacy-preserving federated learning framework based on homomorphic encryption with optimized communication efficiency. The comparison is conducted in terms of both transmitted data volume and execution time.

In the second part, to provide an intuitive comparison of the computational cost and ciphertext size of the proposed encryption algorithm, we evaluate it outside the federated learning framework against traditional homomorphic encryption schemes (Paillier and Elliptic Curve ElGamal) as well as lattice-based cryptosystems (BFV and CKKS). Under the same security level, we compare their performance in terms of encryption/decryption, homomorphic addition, and ciphertext length.

4.2 Settings

This subsection presents the devices and parameters used in our experiments. The proposed federated learning method is simulated on a host equipped with an Intel Core i7-14700KF 20-core processor, 32 GB of RAM, and an NVIDIA 4070Ti GPU with 16 GB of graphics memory, running the Windows operating system. The model was implemented using PyTorch.

General parameters of the federated learning framework were configured as, and $n = 10$, the global training is conducted for 100 rounds, All remaining parameters followed the default configuration in PFLlib [38], whose key settings include a batch size of 10, a local learning rate of 0.05, and 3 local training epochs.

In the second part, in addition to empirically analyzing the security of the algorithm, we also compare its efficiency and ciphertext length with those of existing classical homomorphic encryption algorithms. The specific algorithms and parameter settings used for comparison are as follows: For proposed algorithm, encryption parameters were $bit_{quant} = 5$, $size_{block} = 16$, $n_r = 2$ and $\beta = 4$; The CKKS and BFV encryption algorithms were evaluated using their respective implementations from TenSEAL [39], and the encryption parameters were set according to the default configurations recommended on TenSEAL's official documentation and tutorial pages. For the CKKS scheme, the polynomial modulus degree is set to 8192, the coefficient modulus bit sizes are $[60, 40, 40, 60]$, the global scale is 2^{40}. For the BFV scheme, the polynomial modulus degree is 4096, and the plaintext modulus is 1,032,193. The Paillier and Elliptic Curve ElGamal Cryptosystem (EC ElGamal) algorithms are implemented using LightPHE [40]. To maintain the same 128-bit security level as in the preceding algorithms, the key sizes are set to 3072 bits and 256 bits, respectively. In the

third part, we conducted a comparative analysis of the framework's performance under varying numbers of participating clients.

4.3 Neural Network Models and Datasets

The models and datasets tested in this paper are shown below.

- **Image Classification:** ResNet18 on Cifar-10 [41]. CIFAR-10 is a standard benchmark dataset for image classification, consisting of 60,000 32×32 images across 10 distinct object categories. The dataset was evenly partitioned into ten subsets and distributed among different participants.
- **Text Classification:** Transformer on AGNews [42]. The AGNews dataset contains 31,900 news samples, including headlines and content from four distinct topical categories. The dataset was partitioned into ten equally sized subsets with identical label distributions, following the same approach as in the Cifar-10 setting.
- **Pose Recognition:** HARCNN on PAMAP2 [43]. PAMAP2 provides motion data from nine participants performing 18 different physical activities, which are categorized into 12 fundamental actions such as standing and lying down. As the data originates from nine individual participants, the dataset is naturally divided into nine subsets and allocated to nine clients, differing from the partitioning strategies used for the other datasets.

4.4 Performance Evaluation of the Method

This subsection evaluates the method in terms of accuracy, computational cost, and communication cost during training. The computational cost is assessed by the time consumed in each phase of the method. To evaluate communication cost, the data exchanged in a single round of federated learning is converted into byte streams using the serialization functions provided by NumPy and TenSEAL [39]. The communication cost is then represented as the total volume of data transmitted.

Fig. 4 and Table 2 illustrate the trends in accuracy variation and the final achieved accuracy of the global model under different encryption schemes. It can be observed that, even with the application of quantization, the accuracy and convergence curves of the proposed method remain largely consistent with those obtained from plaintext training and partial encryption, which indicates that the proposed approach does not introduce any adverse impact on model performance in terms of accuracy.

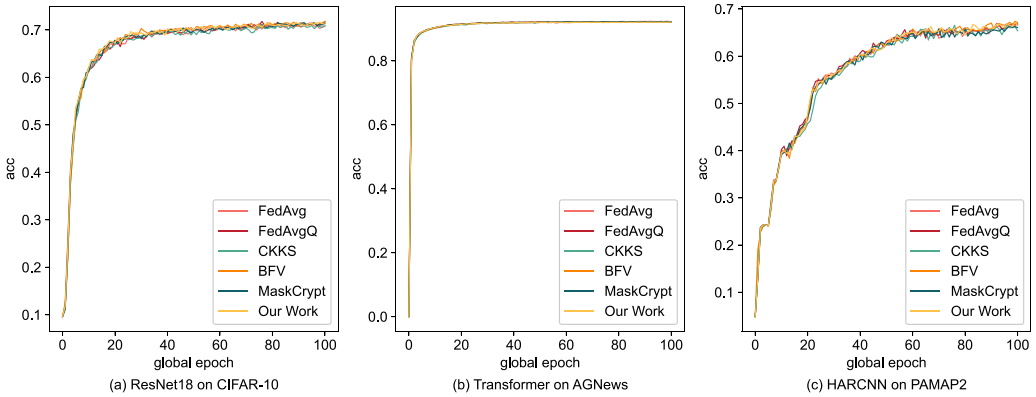


Figure 4: Average accuracy of the global model on each local dataset across different tasks.

Table 2: Final accuracy achieved by various frameworks across different models and datasets.

Model & Dataset	Plaintext	QuantPlaintext	CKKS	BFV	MaskCrypt	Our Work
ResNet18 on CIFAR-10	70.76%	71.69%	70.92%	71.82%	71.39%	71.25%
Transformer on AGNews	92.28%	92.05%	92.18%	92.23%	92.20%	92.11%
HARCNN on PAMAP2	66.68%	66.84%	65.40%	67.13%	66.06%	66.54%

Table 3 summarizes the computational and communication costs of the proposed method by reporting the time consumed at each step as well as the total time per iteration. When he proposed privacy-preserving aggregation mechanism is employed within the proposed framework, both the encryption and aggregation times are significantly reduced compared with those of the CKKS scheme. Although the encryption and decryption times of our algorithm are slightly higher than those of BFV, they remain within an acceptable range. Moreover, our method achieves a shorter aggregation time for homomorphic addition. In addition, unlike conventional encryption schemes that distribute only a single public-private key pair, the multi-key property of our approach allows decryption to be performed solely on the server side, without requiring each client to decrypt locally. This design further reduces the overall computational burden of the system in practical deployments. Although MaskCrypt exhibits the shortest encryption and decryption times among the homomorphic-encryption-based approaches, it requires additional training during the negotiation phase to estimate the gradient sensitivity of the updated model, which results in a longer overall training time per round. Finally, under a network bandwidth setting of 5 Mb/s, we evaluate the total communication time required for each participant to upload and download data to and from the server and other participants.

Table 3: Comparison of the average per-round computational time and communication cost of our work and the baselines.

Task	Method	Agg (s)	Encrypt (s)	Decrypt (s)	Negotiate (s)	Data Transmit (s)	Iteration (s)
ResNet18 on CIFAR-10	FedAvg	0.03	–	–	–	174.52	256.19
	FedAvgQ	0.47	4.95	0.16	–	86.56	170.25
	CKKS	0.95	80.45	2.46	–	1851.90	2013.96
	BFV	0.57	37.67	0.96	–	556.56	674.20
	MaskCrypt	0.28	32.48	1.38	48.60	676.15	838.52
	Our Work	0.30	44.68	3.88	1.95	130.33	259.24
Transformer on AGNews	FedAvg	0.01	–	–	–	351.40	548.05
	FedAvgQ	0.97	10.09	0.34	–	176.44	385.90
	CKKS	3.40	169.19	5.38	–	3752.86	4130.27
	BFV	1.11	75.07	2.07	–	1116.70	1393.62
	MaskCrypt	0.55	63.05	2.64	121.27	1328.90	2001.24
	Our Work	0.61	91.47	7.93	2.12	268.30	581.27
HARCNN on PAMAP2	FedAvg	0.01	–	–	–	60.54	68.36
	FedAvgQ	0.16	1.65	0.05	–	30.42	40.43
	CKKS	0.33	28.02	0.84	–	645.71	683.42
	BFV	0.19	13.08	0.36	–	192.52	214.63
	MaskCrypt	0.09	10.52	0.49	13.01	228.28	260.93
	Our work	0.10	15.47	1.50	1.31	45.53	72.61

The experimental results show that, in a serial simulation environment, our method achieves the shortest per-round execution time among all privacy-preserving federated learning frameworks, thereby realizing a balanced optimization of computational and communication costs.

Under the parameter settings adopted in this subsection, the average communication cost of our encryption algorithm is reduced by 25% compared with plaintext communication, is 50% higher than that of quantized plaintext communication, and is reduced by 93%, 77%, and 81% compared with the CKKS and BFV variants, and MaskCrypt, respectively.

Overall, our method achieves a substantial reduction in communication cost while maintaining the same model accuracy as plaintext training, at the expense of an acceptable computational overhead.

4.5 Security and Performance Evaluation of the Proposed Privacy-Preserving Aggregation Mechanism

This subsection quantitatively evaluates the security of the proposed algorithm based on a set of well-defined metrics. Given that the encryption of parameter tensors functions more similarly to image encryption algorithms than to bit-level or stream encryption schemes, the evaluation methodology proposed by Radwan et al. [44] is referenced in our work. Specifically, the Pixel Change Rate (NPCR) and Unified Average Changing Intensity (UACI) metrics are employed to quantify the differences between plaintext and ciphertext, as well as between different ciphertexts. On the other hand, entropy is used to assess the randomness of the ciphertext. Subsequently, we conducted ablation experiments on each module in the privacy-preserving aggregation mechanism.

Additionally, we conduct a separate comparison, outside the federated learning framework, of the computational time and ciphertext length of the proposed algorithm with several classical homomorphic encryption schemes.

The metrics used in this subsection are described as follows.

- Entropy: Entropy measures the randomness of elements within the encrypted ciphertext. A value closer to the ideal entropy indicates higher unpredictability of the ciphertext. The theoretical maximum entropy is given by $\log_2(cate_{sample})$, which corresponds to the base-2 logarithm of the total number of possible states in the system.

$$Entropy = -\sum_{i=1}^{cate_{sample}} P(sample_i)\log_2 P(sample_i) \tag{32}$$

  where $sample_i$ denotes the event that a ciphertext takes a specific value i in the ring, $cate_{sample}$ denotes the modulus N of the ring.
- UACI (Unified Average Changing Intensity): This metric quantifies the average intensity of differences between two sets of data. The ideal UACI value is approximately $UACI = 33.4635\%$.

$$UACI = \frac{1}{l_{vec}}\sum_{i=1}^{l_{vec}}\frac{|v_{1,i} - v_{2,i}|}{N} \times 100\% \tag{33}$$

  where $v_{1,i}, v_{2,i}$denotes the i-th element of the vector, and l_{vec} represents the length of the vector.
- NPCR (Number of Pixels Change Rate): This metric represents the proportion of differing elements between two datasets relative to the total number of elements. Values approaching 1 indicate greater variability between vectors.

$$NPCR = \frac{1}{l_{vec}}\sum_{i=1}^{l_{vec}}\mathbb{I}(v_{1,i} \neq v_{2,i}) \times 100\% \tag{34}$$

  where $\mathbb{I}$ denotes the indicator function.

4.5.1 Performance Evaluation of the Proposed Privacy-Preserving Aggregation Mechanism

Fig. 5 illustrates the entropy values of ciphertexts corresponding to various plaintext lengths. The horizontal axis of the figure in this subsection represents the number of encrypted parameters. Although a noticeable gap exists between the observed entropy values and the ideal value when the plaintext length is short, this discrepancy can be attributed to the relatively larger space and number of states available in the ciphertext. In such cases, the number of ciphertext states utilized by short plaintexts is insufficient to fully exploit the available entropy. As the number of parameters encrypted in plaintext approaches or exceeds the modulus size (with $N = 25,601$ in our experiments), the entropy gradually approaches its theoretical optimum. It can therefore be hypothesized that the ciphertexts produced by the proposed encryption algorithm exhibit uniform randomness.

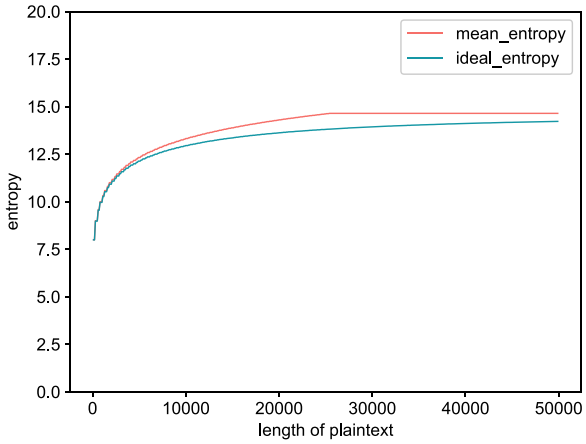


Figure 5: Comparison of ciphertext entropy and the theoretical optimum of the entropy value as the plaintext length increases.

In Fig. 6, a single bit in the plaintext is modified, and the UACI and NPCR values between the corresponding ciphertexts are evaluated. The results demonstrate that, regardless of plaintext length, both NPCR and UACI values approach their theoretical upper bounds. This indicates that the proposed algorithm exhibits high sensitivity to changes in plaintext, thereby offering strong resistance against conventional differential attacks.

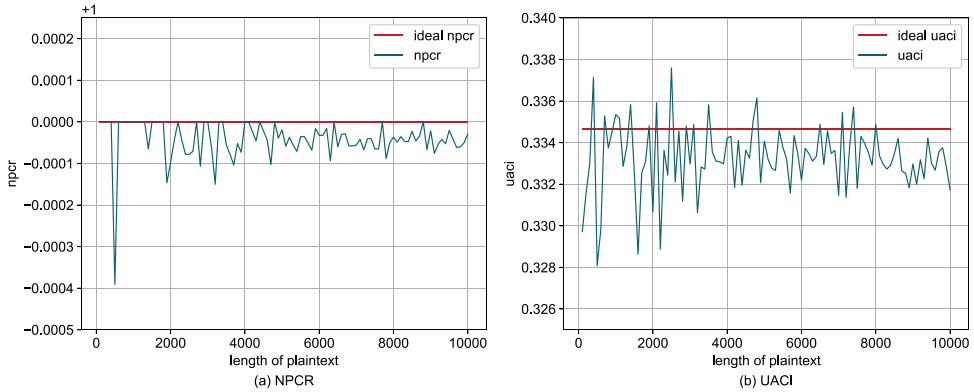


Figure 6: NPCR and UACI values under different plaintext lengths in differential testing experiments.

In Fig. 7, a slight modification is made to either the key or a subkey, and the ciphertexts generated by two separate encryption operations are compared. It is observed that the resulting UACI and NPCR values are both close to their respective theoretical upper bounds, indicating that the algorithm demonstrates strong key sensitivity. This implies that an attacker cannot effectively reduce the key search space through differential key analysis. Overall, the algorithm has demonstrated numerical robustness against statistical, differential, and linear cryptanalytic attacks.

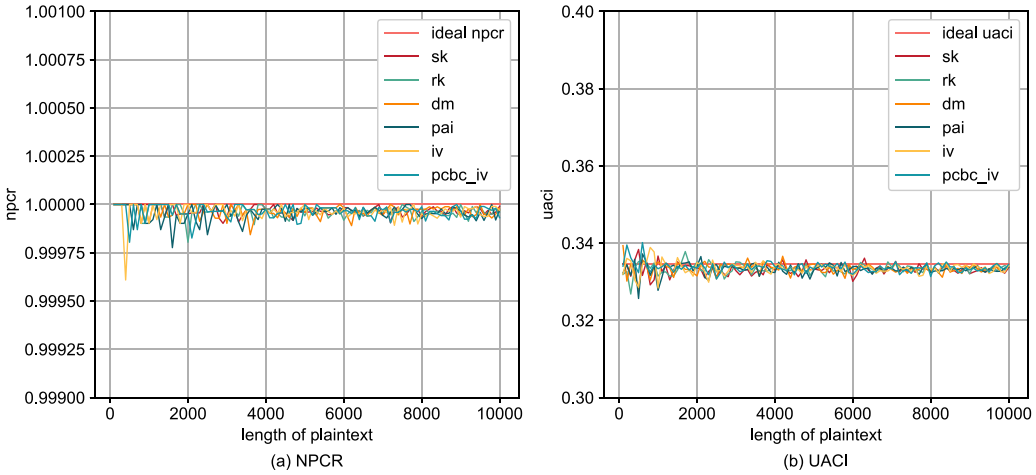


Figure 7: NPCR and UACI values under different plaintext lengths in key differential testing experiments.

Table 4 shows the encryption and decryption times, homomorphic addition times, and ciphertext sizes for various encryption algorithms when 10,000 parameters are to be encrypted, with results averaged over 10 experiments. It can be observed that, compared to traditional encryption methods based on elliptic curve and integer factorization problems, lattice-based encryption and the proposed algorithm exhibit multiple orders of magnitude differences in encryption and decryption times. Although the proposed algorithm has slightly longer encryption, decryption, and aggregation times than the BFV algorithm, it benefits from a relatively shorter ciphertext length.

Table 4: Comparison of cipher sizes and computation times for different encryption algorithms.

Metric	Our Work	CKKS	BFV	Paillier	EC ElGamal
EncTime (s)	3.57e−3	6.78e−3	3.00e−3	1656.53	96.75
DecTime (s)	3.43e−3	1.85e−3	7.29e−4	1667.85	416.34
AggTime (s)	2.86e−5	1.59e−4	7.02e−5	0.54	0.95
CipherSize (Bytes)	81,920	1,002,780	265,635.8	7,844,202.6	1,559,928.5

4.5.2 Ablation Experiments of the Proposed Privacy-Preserving Aggregation Mechanism

To verify the interactions among the modules in the privacy-preserving method, this paper conducts ablation experiments. Specifically, in this experiment, starting from the round key, each operator is sequentially incorporated into the proposed algorithm, and the above metrics are retested to verify the role of each module in the algorithm.

In Fig. 8, stage 1–6 respectively represent the progressive incorporation of the add round key, confusion matrix, permutation, CBC, disturb, and PCBC modules into the privacy-preserving aggregation mechanism.

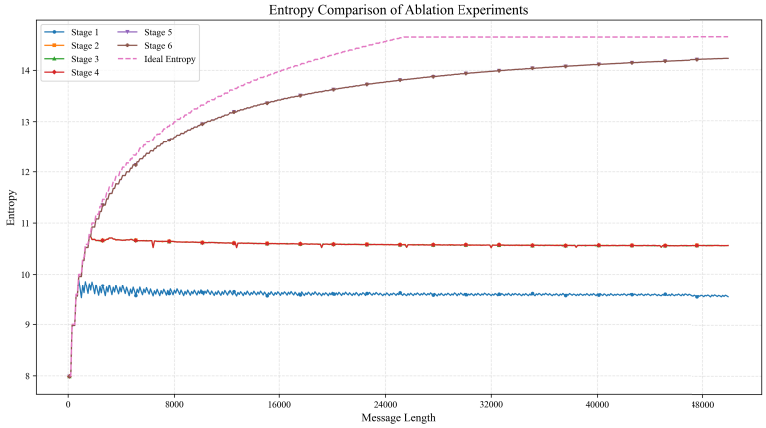


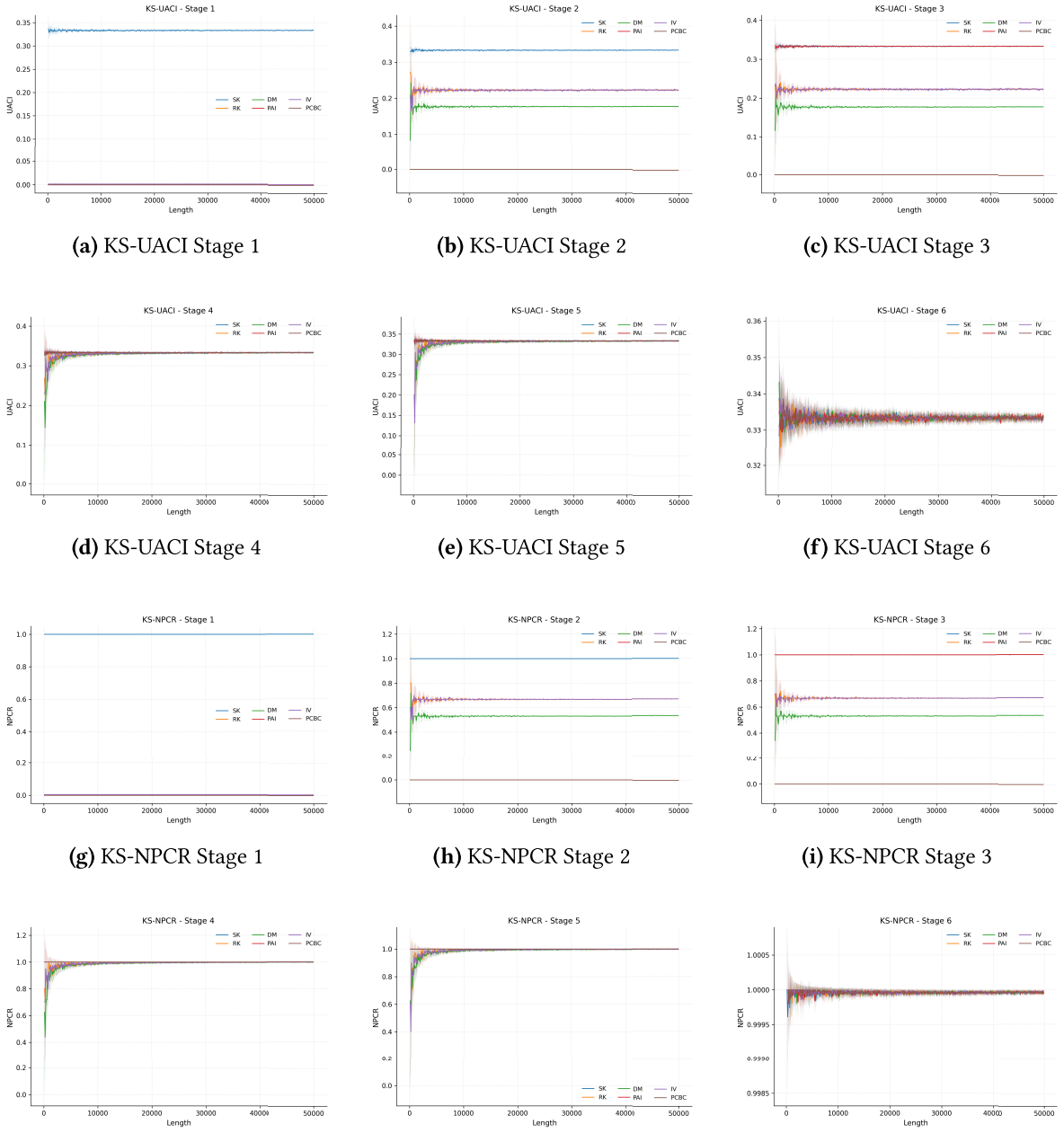
Figure 8: Entropy comparison of ablation experiments.

It can be seen that, through the ablation experiment in which modules are progressively introduced, the involved round-key addition, confusion, and permutation can significantly improve the randomness of the encryption output and rapidly raise its entropy to a relatively high level. However, without introducing perturbation and the encryption mode, the upper bound of the ciphertext entropy is limited by the amount of information contained in the key. After further introducing perturbation and the encryption mode, the entropy of the algorithm can continue to increase steadily and ultimately approach the theoretical ideal value. These results fully demonstrate the effectiveness of the proposed modules, especially the contributions of perturbation and the encryption mode.

Figs. 9 and 10 illustrate the results of the ablation experiments on differential sensitivity and key sensitivity sharing.

Since this is an ablation experiment, it is normal that the sensitivity of the modules that have not yet been incorporated remains constantly zero in the key-sensitivity results shown in Fig. 9. For the remaining modules, it can be observed that, without the collaborative effect of other modules, RK and IV can only affect a single ciphertext component within one ciphertext block. Therefore, when used independently, their key sensitivity is extremely low and close to zero. After the confusion and permutation modules are introduced, the ciphertext sensitivity increases. Specifically, DM and π can not only diffuse differential fluctuations across the entire ciphertext block by themselves, but also interact with RK and IV. With the introduction of CBC, the key sensitivity of the algorithm is further enhanced through the cross-block propagation of differences. The addition of perturbation further reduces the dependence of sensitivity on the ciphertext length. Finally, the introduced PCBC mechanism significantly strengthens the avalanche effect, enabling small fluctuations to propagate rapidly across different ciphertext blocks.

For plaintext differences, when round-key addition, matrix confusion, and permutation are introduced, the fluctuations caused by plaintext differences still cannot propagate across ciphertext blocks and can only affect a single ciphertext block. As a result, the overall sensitivity of the ciphertext decreases rapidly as the ciphertext length increases. After the introduction of CBC and the perturbation mechanism, ciphertext blocks within each CBC group can influence each other, thereby improving the sensitivity. However, the sensitivity still does not reach the ideal level. The PCBC mechanism further connects the entire ciphertext, allowing the influence of a single ciphertext block to diffuse across the whole ciphertext and thereby forming a stronger avalanche effect.



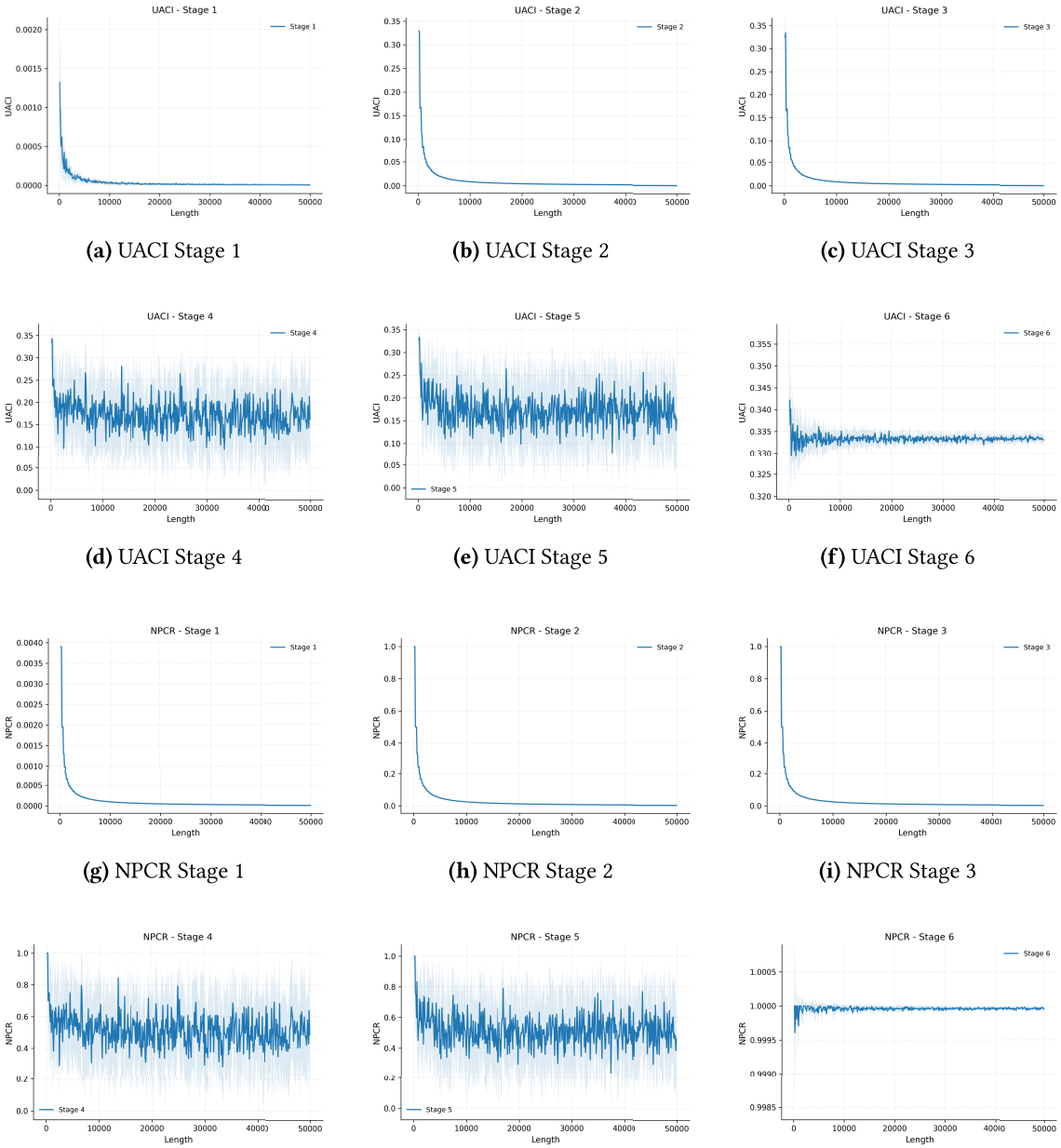
(a) KS-UACI Stage 1 **(b)** KS-UACI Stage 2 **(c)** KS-UACI Stage 3

(d) KS-UACI Stage 4 **(e)** KS-UACI Stage 5 **(f)** KS-UACI Stage 6

(g) KS-NPCR Stage 1 **(h)** KS-NPCR Stage 2 **(i)** KS-NPCR Stage 3

(j) KS-NPCR Stage 4 **(k)** KS-NPCR Stage 5 **(l)** KS-NPCR Stage 6

Figure 9: KS-test UACI and NPCR comparison results under different stages. The upper panel shows the UACI comparison results, and the lower panel shows the NPCR comparison results.



(a) UACI Stage 1 **(b)** UACI Stage 2 **(c)** UACI Stage 3

(d) UACI Stage 4 **(e)** UACI Stage 5 **(f)** UACI Stage 6

(g) NPCR Stage 1 **(h)** NPCR Stage 2 **(i)** NPCR Stage 3

(j) NPCR Stage 4 **(k)** NPCR Stage 5 **(l)** NPCR Stage 6

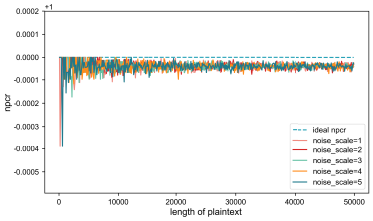
Figure 10: Diff-test UACI and NPCR comparison results under different Stages. The upper panel shows the UACI comparison results, and the lower panel shows the NPCR comparison results.

4.5.3 Sensitivity Analysis of Noise Parameters

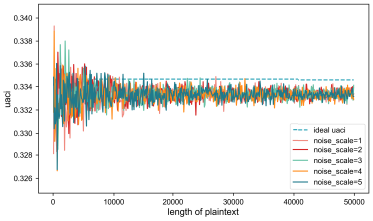
From the theoretical derivation, it can be seen that the security of the algorithm largely comes from the applied random perturbation. To test the sensitivity of the proposed algorithm to the noise perturbation parameter, we adjust the noise scale introduced into the algorithm while keeping the modulus used by the algorithm unchanged, and verify the metric values under different noise scales.

As shown in Figs. 11 and 12, when the noise parameter is varied, the statistical properties and sensitivity of the algorithm's ciphertext change only slightly. The experimental results indicate that our algorithm

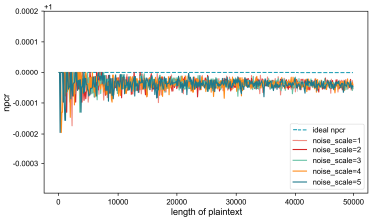
exhibits good statistical properties under different noise levels, suggesting that the algorithm has a low dependence on different noise levels.



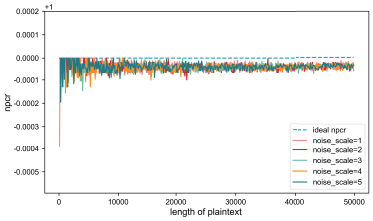
(a) Differential NPCR



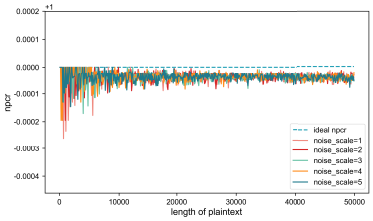
(b) Differential UACI



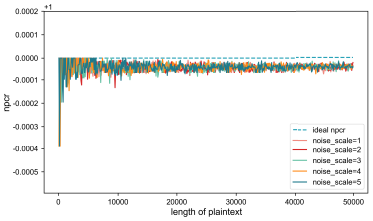
(c) KS-NPCR (DM)



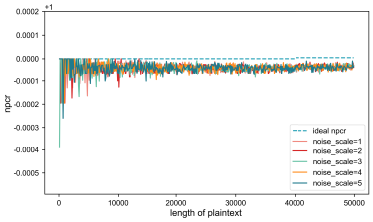
(d) KS-NPCR (IV)



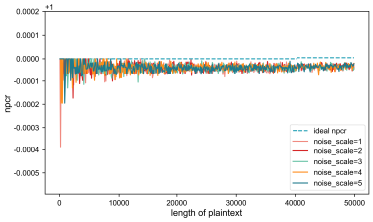
(e) KS-NPCR (PAI)



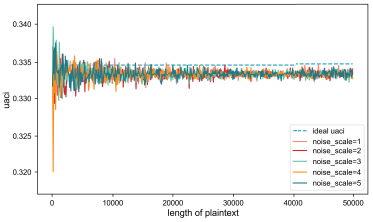
(f) KS-NPCR (PCBC)



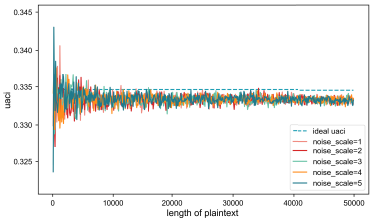
(g) KS-NPCR (RK)



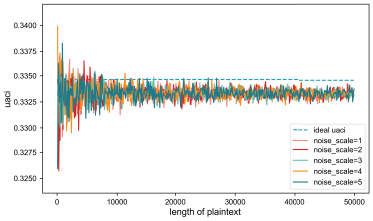
(h) KS-NPCR (SK)



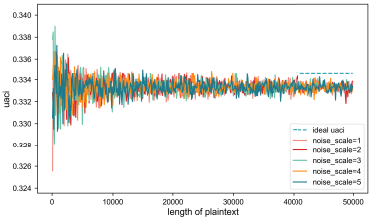
(i) KS-UACI (DM)



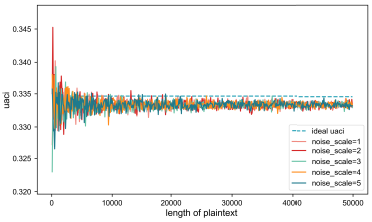
(j) KS-UACI (IV)



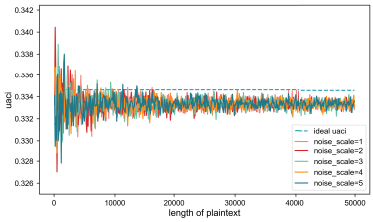
(k) KS-UACI (PAI)



(l) KS-UACI (PCBC)



(m) KS-UACI (RK)



(n) KS-UACI (SK)

Figure 11: Experimental results under different noise scales.

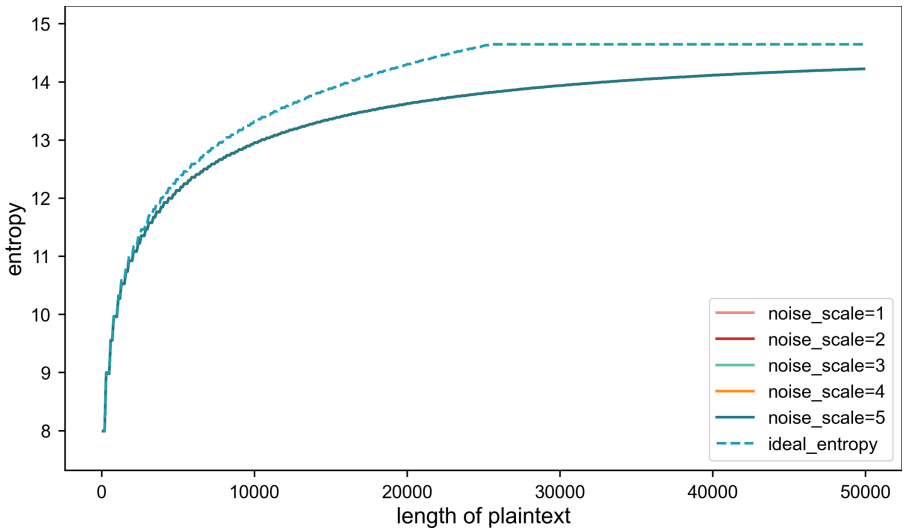


Figure 12: Entropy comparison under different noise scales.

4.6 Parameter Sensitivity Analysis of the Federated Learning Method

In this subsection, to further explore the performance of the proposed federated learning method, we conduct additional parameter-tuning tests on the proposed method. Specifically, to verify the robustness of the proposed algorithm to non-IID (non-independent and identically distributed) and non-balanced data when applied to the federated learning framework, this subsection first partitions the data into non-balanced datasets according to different Dirichlet distribution parameters, and tests the robustness of the algorithm to non-uniformly distributed data under different settings. Subsequently, to verify the analysis of the computational and communication costs of the proposed algorithm in Section 3.6, this paper adjusts the number of participants within a certain range, repartitions the complete dataset after each adjustment, and evaluates the accuracy, computational cost, and communication cost during the training process. Finally, to test the sensitivity of the model accuracy of the proposed method to quantization precision, the quantization precision is adjusted under the setting of 10 participants, IID, and balanced data.

As shown in Fig. 13, federated learning can proceed normally when the degree of data distribution shift among participants is relatively low, whereas training collapses under extremely severe data shift. This is because the proposed method performs aggregation by weighting the gradients according to the number of samples held by each participant, thereby exhibiting robustness to non-IID distributions under non-extreme settings, similar to FedAvg. These results demonstrate that the proposed quantization and encryption operations do not compromise the robustness of weighted aggregation.

As shown in Fig. 14a, the model accuracy decreases as the number of participants increases. However, this is a common phenomenon in federated learning training. As the number of participants increases, data fragmentation may lead to local overfitting. Meanwhile, since the test model used in this study contains normalization layers, specifically batch normalization (BN) layers, the reduced amount of data available to each participant may decrease the stability of the BN statistics estimation.

The experimental results in Fig. 14b and Fig. 14c indicate that the computational and communication costs of the proposed algorithm increase approximately linearly with the number of participants, which is consistent with the theoretical analysis. When the number of participants varies, the local training time remains almost unchanged. This is because, each time the number of participants is adjusted, the entire dataset is repartitioned, and each experiment still performs forward and backward propagation over the

full dataset, resulting in only minor differences in training time. In most cases, the negotiation time can be neglected because mainstream deep learning models generally contain a large number of parameters, and thus the computational cost introduced by key negotiation is negligible in comparison.

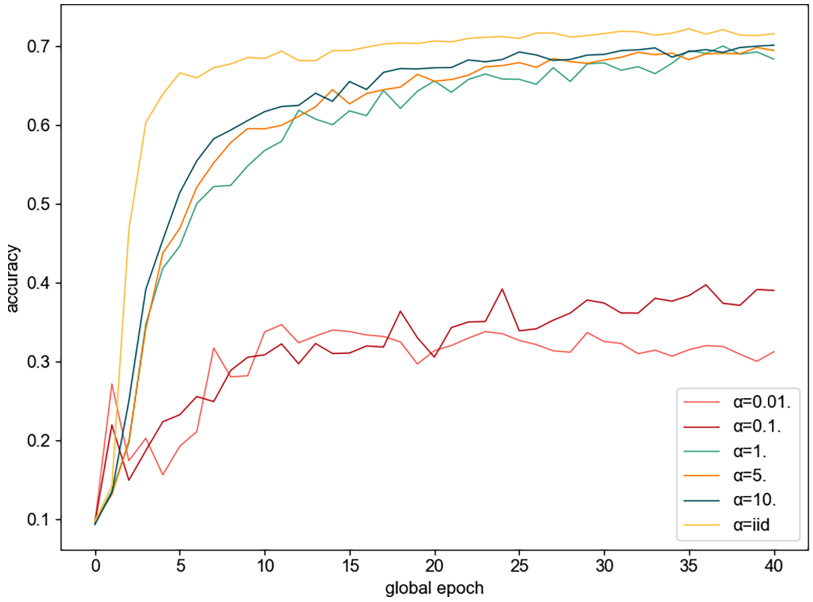
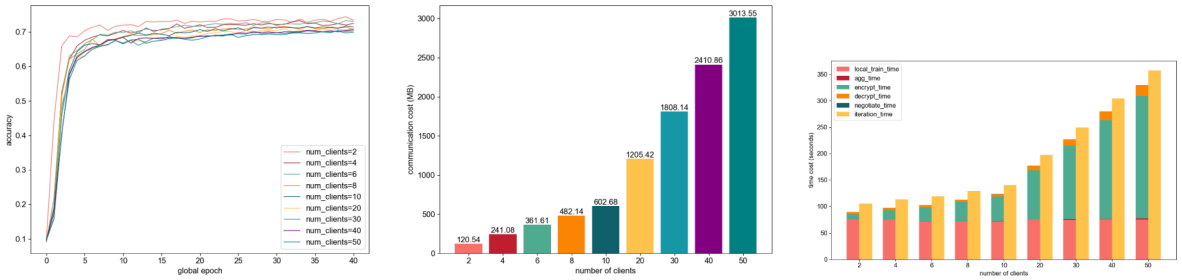


Figure 13: Test accuracy under different non-IID degrees.

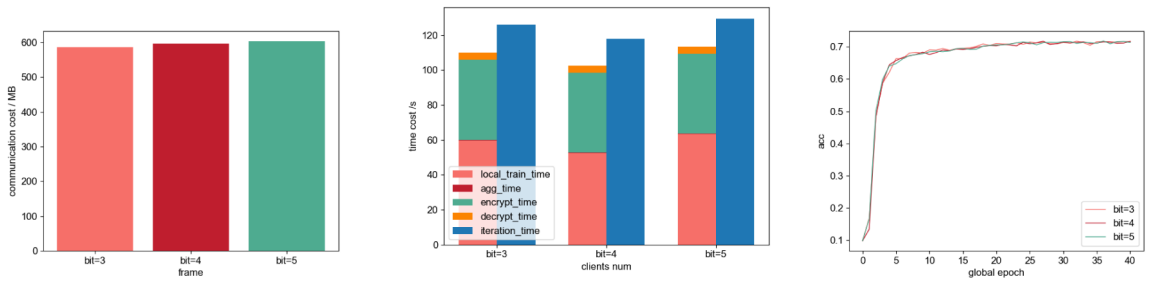


(a) Accuracy (b) Communication cost (c) Time cost

Figure 14: Performance comparison under different numbers of clients.

The experimental results in Fig. 15 show that the training accuracy remains consistent across different quantization precisions, indicating that the proposed algorithm can still maintain high accuracy even at relatively low quantization precision. The communication cost increases linearly with the quantization precision, which is consistent with the derivation in the theoretical analysis. Meanwhile, the computation time fluctuates only within a small range, which can be regarded as an acceptable measurement error.

Finally, we present in detail the percentage of time occupied by each step during the training process when the number of participants is set to 10.



(a) Communication cost (b) Time cost (c) Accuracy

Figure 15: Performance comparison under different quantization bit settings.

Table 5 presents the average time consumption and the proportion of total training duration for each step in each training round when the number of clients is 10. In each training round, the processes of local training and encryption account for the largest proportion of time consumption. The time taken by decryption and negotiation is relatively small. Ciphertext aggregation consumes the least time, which is almost negligible.

Table 5: Time consumption and proportion of each process.

Process	Time (s)	Proportion (%)
Local training time	64.03	49.68
Aggregation time	0.30	0.23
Encryption time	44.67	34.66
Decryption time	3.88	3.01
Negotiation time	1.95	1.52
Total computation time	128.90	100.00

5 Conclusion

In this work, a short-ciphertext multi-key privacy-preserving mechanism, referred to as MKPHE-FL, is first constructed. Built upon a symmetric block encryption framework, the scheme features low communication overhead and acceptable computational cost. Furthermore, it supports multi-key encryption, making it suitable for federated learning scenarios with strong privacy and security requirements. Leveraging the homomorphic properties of secret sharing, a secret-sharing-based key negotiation protocol (SS-KNP) is designed to aggregate the distinct keys generated by individual clients during encryption.

Based on the above algorithms, we design a federated learning method capable of effectively defending against collusion-based privacy attacks. Within this method, participants can adaptively select encryption precision and noise levels according to the scale of the federated learning task. Each client is allowed to use a distinct sub-key to prevent privacy leakage from other participants.

Theoretical analysis shows that, through the use of encryption modes and randomization mechanisms, our algorithm is resistant to linear and differential attacks. Furthermore, with appropriately chosen parameters, the encryption process does not significantly increase message length. Numerical experiments demonstrate that our algorithm resists statistical and differential analysis attacks and, when applied within the federated learning method, achieves moderate computational cost and low communication cost.

Although the proposed algorithm offers certain advantages in communication cost, its adoption of the classical FedAvg framework for model aggregation leads to relatively weak resilience against malicious attacks. Future research will investigate how the homomorphic properties of the encryption scheme can be leveraged to quantify the similarity among participant-uploaded gradients, thereby improving the robustness of federated learning against adversarial attacks.

At the same time, although the scheme proposed in this paper achieves significant advantages in computational efficiency and communication overhead, we note that the current protocol design still has room for improvement in robustness against client dropouts. Since our method relies on secret sharing to realize key agreement, if some participating nodes unexpectedly drop out during a training round due to network fluctuations, it is highly likely that the received shares among participants will become inconsistent, thereby preventing the successful aggregation of all keys. Although the training of those participants with stable network connections can still proceed through partial key aggregation, the frequent occurrence of such events will cause the updates from participants affected by network fluctuations to be ignored during aggregation. In future work, it should be considered how to quickly compensate for this missing contribution to the global model when dropped participants are occasionally able to reconnect.

Acknowledgement: The authors thank Dr. Dai Yu from the School of Mathematics and Statistics, Wuhan University of Technology, for helpful guidance and suggestions during manuscript revision.

Funding Statement: This paper received funding support from the PCL-CMCC Foundation for Science and Innovation (Grant No. 2024ZY2B0050).

Author Contributions: Conceptualization: Hongzhen Liu, Xi Fang; Methodology: Hongzhen Liu; Software: Hongzhen Liu; Formal Analysis: Hongzhen Liu; Investigation: Hongzhen Liu; Writing—Original Draft: Hongzhen Liu; Writing—Review and Editing: Zhe Zhang, Liang Xie, Yuan Wan, Xi Fang; Supervision: Zhiqiang Ru; Project Administration: Zhiqiang Ru; Validation: All authors. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The datasets used in this paper include Cifar-10 [41], AGNews [42], and PAMAP2 [43]. Among them, Cifar10 can be obtained from https://www.cs.toronto.edu/~kriz/cifar.html, AGNews can be obtained from https://www.kaggle.com/datasets/amananandrai/ag-news-classification-dataset, and PAMAP2 can be obtained from https://doi.org/10.24432/C5NW2H.

Ethics Approval: Not applicable. This study did not involve any human or animal participants.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A Security Analysis of the Proposed Privacy-Preserving Aggregation Mechanism

Appendix A.1 Algebraic Analysis

This appendix analyzes the algorithm's resistance to chosen-plaintext attacks based on linear algebraic techniques. The algorithm is simplified firstly using linear algebra, and then its resistance to linear and differential attacks under the assumption that the adversary possesses the key sk and a certain number of plaintext-ciphertext pairs is analysed.

Appendix A.1.1 Assumptions about the Adversary

Assume that the adversary holds l_{matrix} pairs of sequentially corresponding plaintext matrix lists and ciphertext matrix lists, i.e., $M = [M_0, M_1, \ldots, M_{l_{matrix}-1}]$ and $C = [C_0, C_1, \ldots, C_{l_{matrix}-1}]$. Meanwhile, due to the requirements of the federated learning setup, the adversary also holds the key sk used to generate the subkeys RK, DM, and π. The main goal of the adversary is to establish a linear equation model for recovering

the IV_s parameters used in generating these plaintext-ciphertext pairs, or to infer information about other original plaintexts therefrom.

Appendix A.1.2 Simplification of the Algorithm

In this subsection, we take the encryption procedure under the parameter setting $n_r = 2$ and $size_{matrix} = 2$ as an illustrative example (used solely for derivation purposes and not reflecting the parameters adopted in practical instantiation). The encryption process of a plaintext block $M_i (i \geq 2)$ is reformulated into a matrix-based computational representation. Specifically, we first redefine a subset of the encryption operators, then present the matrix-vector formulations of the operators introduced above, and finally compose these operators to obtain an equivalent matrix representation of the overall encryption procedure.

Since the initialization vector (IV) is treated as an unknown key to be solved in this subsection, we isolate the IV from Eq. (4) and denote the resulting operation as $RK(\cdot)$. The operators $DM(\cdot)$ and $\pi(\cdot)$ remain consistent with Eqs. (5) and (6), respectively.

For ease of exposition, we define one complete encryption round composed of these key-dependent operations as $Round(\cdot)$. The specific definitions of $RK(\cdot)$ and $Round(\cdot)$ are given in Eqs. (A1) and (A2), respectively.

$$RK(M) = RK \odot M \tag{A1}$$

$$Round(M) = RK(DM(\pi(M))) \tag{A2}$$

The three keys used in the encryption process, together with a single encryption round, can be transformed into the forms given in Eqs. (A3)–(A5):

For $RK(\cdot)$, the element-wise multiplication between matrices ($\odot$) can be reformulated as the multiplication of a diagonal matrix with a vector.

$$\begin{aligned} RK(M) &= \begin{bmatrix} rk_{1,1} & rk_{1,2} \\ rk_{2,1} & rk_{2,2} \end{bmatrix} \odot \begin{bmatrix} m_{1,1} & m_{1,2} \\ m_{2,1} & m_{2,2} \end{bmatrix} \\ &= \begin{bmatrix} rk_{1,1}m_{1,1} & rk_{1,2}m_{1,2} \\ rk_{2,1}m_{2,1} & rk_{1,2}m_{2,2} \end{bmatrix} \\ RK'\vec{M} &= diag\{rk_{1,1}, rk_{1,2}, rk_{2,1}, rk_{2,2}\}\vec{M} \end{aligned} \tag{A3}$$

For $DM(\cdot)$, since matrix multiplication is a linear transformation, the coefficients that map the original vector to each transformed element can be explicitly extracted.

$$\begin{aligned} DM^{-1} \times M \times DM &= \begin{bmatrix} dm_{1,1}^{inv} & dm_{1,2}^{inv} \\ dm_{2,1}^{inv} & dm_{2,2}^{inv} \end{bmatrix} \times \begin{bmatrix} m_{1,1} & m_{1,2} \\ m_{2,1} & m_{2,2} \end{bmatrix} \times \begin{bmatrix} dm_{1,1} & dm_{1,2} \\ dm_{2,1} & dm_{2,2} \end{bmatrix} \\ DM'\vec{M} &= \begin{bmatrix} dm_{1,1}^{inv}dm_{1,1} & dm_{1,1}^{inv}dm_{2,1} & dm_{1,2}^{inv}dm_{1,1} & dm_{1,2}^{inv}dm_{2,1} \\ dm_{1,1}^{inv}dm_{1,2} & dm_{1,1}^{inv}dm_{2,2} & dm_{1,2}^{inv}dm_{1,2} & dm_{1,2}^{inv}dm_{2,2} \\ dm_{2,1}^{inv}dm_{1,1} & dm_{2,1}^{inv}dm_{2,1} & dm_{2,2}^{inv}dm_{1,1} & dm_{2,2}^{inv}dm_{2,1} \\ dm_{2,1}^{inv}dm_{1,2} & dm_{2,1}^{inv}dm_{2,2} & dm_{2,2}^{inv}dm_{1,2} & dm_{2,2}^{inv}dm_{2,2} \end{bmatrix} \vec{M} \end{aligned} \tag{A4}$$

The permutation operation is equivalent to applying an elementary transformation matrix to shift the original vector.

$$\pi(M) = \Pi'\vec{M} = \begin{bmatrix} p_{1,1} & p_{1,2} & p_{1,3} & p_{1,4} \\ p_{2,1} & p_{2,2} & p_{2,3} & p_{2,4} \\ p_{3,1} & p_{3,2} & p_{3,3} & p_{3,4} \\ p_{4,1} & p_{4,2} & p_{4,3} & p_{4,4} \end{bmatrix} \vec{M} \tag{A5}$$

Here, $diag(\cdot)$ denotes a diagonal matrix. The terms $rk_{i,j}, dm_{i,j}$ represent the elements at the corresponding positions in the original key matrices. The matrices $RK^{'}$, $DM^{'}$, and Π' denote the key matrices transformed into their equivalent forms. $\vec{M}$ denotes the plaintext obtained by vectorizing the plaintext matrix in a left-to-right, top-to-bottom order, as shown in Eq. (A6). The value of $p_{i,j}$ is defined as in Eq. (A7), where $\pi_{r,c}$ denotes the element at row r and column c of the original permutation matrix π, and $\mathbb{I}(\cdot)$ is the indicator function.

$$\vec{M} = \begin{bmatrix} m_{1,1} \\ m_{1,2} \\ m_{2,1} \\ m_{2,2} \end{bmatrix} \tag{A6}$$

$$\begin{aligned} &p_{i,j} = \mathbb{I}_{\pi_{r,c}=j} \\ &(r, c = 1, \ldots, size_{matrix}, \\ &i, j = 1, \ldots, size^2_{matrix}, \\ &i = (r-1) \times size_{matrix} + c) \end{aligned} \tag{A7}$$

Consequently, the operation $Round(\cdot)$ can also be reformulated into a similar representation, as shown in Eq. (A8).

$$\begin{aligned} &Round(M) = RK(DM(\pi(M))) \\ &Round'\vec{M} = RK'DM'\Pi'\vec{M} \end{aligned} \tag{A8}$$

For the PCBC mode, since the adversary has access to certain plaintext-ciphertext pairs, the unknown variables that introduce interference into the recovery process are the perturbations R_i and R_{i-1} added to the current and previous plaintexts, respectively. The PCBC mechanism in the overall expression of Eq. (3) can therefore be decomposed into a known part and an unknown part.

$$\begin{aligned} C_{pcbc,i} &= scale(M_i + M_{i-1}) + C_{i-1} + R_i + R_{i-1} \\ &= M_{scale,i} + (R_i + R_{i-1}) \\ &= scale \times E_4 \times (\vec{M}_i + \vec{M}_{i-1}) + \vec{C}_{i-1} + (\vec{R}_i + \vec{R}_{i-1}) \end{aligned} \tag{A9}$$

$$\vec{C}_{pcbc,i} = \vec{M}_{scale} + \vec{R}_i + \vec{R}_{i-1} \tag{A10}$$

Here, $scale$ is a scalar, E denotes the identity matrix, and $\vec{C}_{i-1}, \vec{R}_i$ are the ciphertext and noise matrices, respectively, vectorized in the same manner as the plaintext.

For the CBC mode, the core operation lies in introducing the two preceding ciphertexts C'_{i-1} and C'_{i-2} during the encryption process. This step can be reformulated as shown in Eq. (A11).

$$\vec{C}_{cbc,i} = \begin{bmatrix} E_4 & E_4 & E_4 \end{bmatrix} \begin{bmatrix} \vec{C'}_i \\ \vec{C'}_{i-1} \\ \vec{C'}_{i-2} \end{bmatrix} \tag{A11}$$

After incorporating the initialization vector (IV), and under the setting $n_r = 2$, the entire encryption process can be expanded as follows:

$$\begin{aligned} \vec{C}_i = {} & Round'_{i,2} \begin{bmatrix} Round'_{i,1}RK'_{i,0} & Round'_{i-1,1}RK'_{i-1,0} & Round'_{i-2,1}RK'_{i-2,0} \end{bmatrix} \\ & \times \left(\begin{bmatrix} \vec{M}_{scale,i} \\ \vec{M}_{scale,i-1} \\ \vec{M}_{scale,i-2} \end{bmatrix} + \begin{bmatrix} \vec{R}_i + \vec{R}_{i-1} \\ \vec{R}_{i-1} + \vec{R}_{i-2} \\ \vec{R}_{i-2} + \vec{R}_{i-3} \end{bmatrix} \right) \\ & + Round'_{i,2} \begin{bmatrix} Round'_{i,1} & Round'_{i-1,1} & Round'_{i-2,1} \end{bmatrix} \begin{bmatrix} \vec{IV}_{i,0} \\ \vec{IV}_{i-1,0} \\ \vec{IV}_{i-2,0} \end{bmatrix} \\ & + Round'_{i,2} \begin{bmatrix} E_4 & E_4 & E_4 \end{bmatrix} \begin{bmatrix} \vec{IV}_{i,1} \\ \vec{IV}_{i-1,1} \\ \vec{IV}_{i-2,1} \end{bmatrix} + \vec{IV}_{i,2} \end{aligned} \tag{A12}$$

Considering the update strategy of IV_s ($order_a$), the IV_s used in different rounds can be equivalently transformed using matrix operations.

$$\begin{aligned} \begin{bmatrix} \vec{IV}_{i,r+1} \\ \vec{IV}_{i-1,r+1} \\ \vec{IV}_{i-2,r+1} \end{bmatrix} &= \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \vec{IV}_{i,r} \\ \vec{IV}_{i-1,r} \\ \vec{IV}_{i-2,r} \end{bmatrix} \\ \begin{bmatrix} \vec{IV}_{i,r+1} \\ \vec{IV}_{i-1,r+1} \\ \vec{IV}_{i-2,r+1} \end{bmatrix} &= P_{IV} \begin{bmatrix} \vec{IV}_{i,r} \\ \vec{IV}_{i-1,r} \\ \vec{IV}_{i-2,r} \end{bmatrix} \end{aligned} \tag{A13}$$

After substituting the corresponding transformations and merging the IV-related terms in Eq. (A12), the original equation can be reformulated as a noisy linear system over a ring.

$$\begin{aligned} \vec{C}_i &= A_i^M \begin{bmatrix} \vec{M}_{scale,i} \\ \vec{M}_{scale,i-1} \\ \vec{M}_{scale,i-2} \end{bmatrix} + A_i^R \begin{bmatrix} (\vec{R}_i + \vec{R}_{i-1}) \\ (\vec{R}_{i-1} + \vec{R}_{i-2}) \\ (\vec{R}_{i-2} + \vec{R}_{i-3}) \end{bmatrix} + A_i^{IV} \begin{bmatrix} \vec{IV}_{i,0} \\ \vec{IV}_{i-1,0} \\ \vec{IV}_{i-2,0} \end{bmatrix} \\ A_i^M &= A_i^R = Round'_{i,2} \begin{bmatrix} Round'_{i,1}RK'_{i,0} & Round'_{i-1,1}RK'_{i-1,0} & Round'_{i-2,1}RK'_{i-2,0} \end{bmatrix} \\ A_i^{IV} &= Round'_{i,2} \begin{bmatrix} Round'_{i,1} & Round'_{i-1,1} & Round'_{i-2,1} \end{bmatrix} + Round'_{i,2} \begin{bmatrix} E_4 & E_4 & E_4 \end{bmatrix} P_{IV} \\ &\quad + \begin{bmatrix} E_4 & O & O \end{bmatrix} P_{IV}^2 \end{aligned} \tag{A14}$$

Here, O denotes the zero matrix. Clearly, when n_r and $size_{matrix}$ take other values, $\vec{C}_i$ can likewise be expanded in a similar form.

Appendix A.1.3 Analysis of Algebraic Methods for Solving Keys

Since the IV_{pcbc} and IV_{cbc} can be solved separately from the other initialization vectors in a similar manner, and jointly recovering all these keys would further complicate the solution process, we do not

consider the effects of IV_{pcbc} and IV_{cbc} here. Accordingly, we assume $i \geq 2$, under which $n_r + 1$ laintext-ciphertext pairs are required. However, even if a sufficient number of plaintext-ciphertext pairs are obtained, the random perturbation R added during the encryption process remains unknown. Considering Eq. (A14), in the solving process, besides introducing one R_i for each additional equation, the $n_r + 1$ systems of equations used will also introduce noises corresponding to $i-1$, $i-2$ and $i-3$. Section 4.5 has demonstrated the sensitivity of the privacy-preserving aggregation mechanism to the original text and the secret key.

Meanwhile, the random perturbation vector $\vec{R}$, after undergoing a complete encryption process, becomes a vector whose components are approximately uniformly distributed in the frequency domain, thereby introducing a significant disturbance into the underlying equations. Therefore, it is relatively difficult to eliminate the impact of perturbations on the equations. On the other hand, if one intends to obtain the correct R through traversal, a total of $(2\beta+1)^{((n_r+4)\times size_{block}^2)}$ cases need to be considered. By adjusting β, n_r, and $size_{block}^2$ according to the required security level, this problem can be rendered computationally infeasible. Due to the nature of exponents, this can be easily achieved by increasing n_r and $size_{block}^2$.

Appendix A.1.4 Analysis of Algebraic Methods for Stealing Plaintext Information

Consider the case where the algorithm encounters a differential attack. In the algorithm proposed in this paper, to avoid encrypting all ciphertexts in the same pattern, the position of the key used is continuously rotated according to the plaintext and encryption process. However, for the subkeys, this essentially operates in the ECB (Electronic Codebook) mode. During the rotation process, after every $cycle = n_r \times (n_r + 1)$ plaintexts, the order of the keys used during encryption will repeat.

That is, A_i^{IV}, A_i^M, A_i^R, together with the employed IV_s,become identical. In this case, IV_s can be eliminated by taking differences between ciphertexts, leading to Eq. (A15). The following is an analysis of this scenario.

$$\vec{C}_{i+cycle} - \vec{C}_i = A_i^M \begin{bmatrix} \vec{M}_{scale,i+cycle} - \vec{M}_{scale,i} \\ \vec{M}_{scale,i+cycle-1} - \vec{M}_{scale,i-1} \\ \vec{M}_{scale,i+cycle-2} - \vec{M}_{scale,i-2} \end{bmatrix} + A_i^R \begin{bmatrix} (\vec{R}_{i+cycle} + \vec{R}_{i+cycle-1}) - (\vec{R}_i + \vec{R}_{i-1}) \\ (\vec{R}_{i+cycle-1} + \vec{R}_{i+cycle-2}) - (\vec{R}_{i-1} + \vec{R}_{i-2}) \\ (\vec{R}_{i+cycle-2} + \vec{R}_{i+cycle-3}) - (\vec{R}_{i-2} + \vec{R}_{i-3}) \end{bmatrix} \quad \text{(A15)}$$

Although this approach eliminates IV_s, it introduces more R.

Although Eq. (A14) indicates that $A_i^M = A_i^R$, and under our assumption that the adversary has knowledge of RK, DM and Π', the equation appears solvable by directly inverting the matrix. in practice, however, under the setting $n_r = 2, size_{matrix} = 2$ (with other cases being analogous), solving this equation via matrix inversion would require finding a 12×4 matrix $A_i^{inv,M}$ such that $A_i^{inv,M} \times A_i^M = E_{12}$. Due to the dimensional constraints of the matrices, we have $rank(A_i^M) \leq 4$ and $rank(A_i^{inv,M}) \leq 4$. Consequently, the product of these two matrices cannot yield the identity matrix E_{12}, which has rank 12. From an encryption-decryption perspective, this phenomenon can be interpreted as follows: during the CBC operation, three vectors processed through different encryption procedures are added together, and as a result, correct decryption cannot be performed until the inverse CBC operation is applied. Although this equation eliminates the effect of the IV, it introduces additional R. Through reasoning similar to that in the previous subsection, it can be concluded that solving this equation to extrapolate other plaintexts from the known ones is also computationally infeasible.

Based on the aforementioned simplifications and deductions, it can be concluded that the proposed algorithm exhibits resistance to commonly used chosen-plaintext attacks.

Appendix A.2 Resistance to Structural Cryptanalysis Based on Noise Evolution

Although the simplification against algebraic attacks in the previous section indicates that the core security of the proposed algorithm is achieved through the perturbation and diffusion of noise in the encryption process, which may render conventional theoretical analyses of differential and linear attacks inapplicable, the encryption process is still built upon a conventional multi-round block cipher structure. Therefore, in this section, we further investigate differential cryptanalysis (DC) and linear cryptanalysis (LC) in the context of traditional block ciphers by formally estimating the cumulative noise distribution after multiple rounds of iteration, thereby showing that the algebraic structure of the proposed algorithm does not introduce exploitable structural weaknesses.

Without considering the encryption mode, the proposed algorithm mainly consists of three key operators: round-key addition, matrix confusion, and permutation. Among them, permutation is merely a shift operation and does not affect the overall noise introduced during the perturbation stage. Therefore, in this section, we mainly analyze the effects of multi-round round-key addition and matrix confusion on the noise distribution. For ease of analysis, we first consider the distribution of the noise after multiple rounds of encryption over the integer domain, and then study its distribution over the modular ring after modular reduction.

Let the initially injected noise be $e_0 \sim U([-\beta, \beta])$, whose variance is approximately $\beta^2/3$. During encryption, this noise is repeatedly transformed by round-key addition and matrix confusion. Since the round keys and confusion matrices are sampled from the ring and are independent of the noise, each round amplifies the noise variance. After n_r rounds, the accumulated noise can be summarized as

$$\mathrm{Var}(E_{final}) = O\left((dim \cdot N^4)^{n_r} \cdot \beta^2 \cdot N^2\right). \tag{A16}$$

This indicates that the perturbation grows rapidly with the number of rounds. When the accumulated noise is reduced modulo N, it is wrapped into the ring $\mathbb{Z}_N$:

$$E_{ring} = E_{final} \bmod N. \tag{A17}$$

Since the standard deviation of E_{final} is much larger than N, the wrapped distribution becomes close to uniform over $\mathbb{Z}_N$ according to the smoothing effect of modular reduction. Therefore, the ciphertext distribution is dominated by the accumulated noise rather than by the original plaintext structure.

For differential cryptanalysis, let two plaintexts be M_1 and M_2, and define $\Delta M = M_1 - M_2$. The corresponding ciphertext difference can be written as

$$\Delta\vec{C} = \vec{C}_1 - \vec{C}_2 = A^M \Delta\vec{M} + \Delta\vec{E}_R \pmod{q}, \tag{A18}$$

where $\Delta\vec{E}_R$ denotes the difference between two independently accumulated noise terms. Under the above noise amplification process, $\Delta\vec{E}_R$ dominates the deterministic term $A^M \cdot \Delta\vec{M}$. Thus, for any fixed differential characteristic γ, we have

$$\Pr[\Delta C = \gamma \mid \Delta M] \approx \frac{1}{N^{size^2_{matrix}}}. \tag{A19}$$

This probability is close to random guessing, which means that no useful differential trail can be exploited.

For linear cryptanalysis, an adversary attempts to find nonzero linear masks α and $beta$ such that a linear approximation between the plaintext and the ciphertext holds with a non-negligible bias. Specifically, the adversary considers whether there exists a constant $c \in \mathbb{Z}_N$ such that

$$\Pr\left[\langle \beta, \vec{C} \rangle \equiv \langle \alpha, \vec{M} \rangle + c \pmod{N}\right] = \frac{1}{N} + \varepsilon \tag{A20}$$

where ε denotes the linear bias. A successful linear attack requires $|\varepsilon|$ to be non-negligibly larger than zero. According to the ciphertext generation process, the ciphertext can be written as Eq. (A14).

After multi-round diffusion and noise amplification, $\vec{E}_R$ is close to uniform over $\mathbb{Z}_N$ and independent of $\vec{M}$, which masks any plaintext-dependent linear term. Consequently, $\varepsilon \to 0$, and no exploitable linear correlation exists between $\vec{M}$ and $\vec{C}$.

In summary, the multi-round diffusion and noise amplification mechanism makes both differential and linear statistical structures indistinguishable from random behavior over the modular ring. Therefore, the proposed mechanism does not leave exploitable structural weaknesses for differential or linear cryptanalysis.

Appendix A.3 Standard Cryptographic Model and Formal Proof

As shown by the previous derivation of the algebraic structure, the encryption process proposed in this paper essentially constitutes a noisy sparse linear evolution system over a ring. Although its single-step operations are highly similar to the classical Learning With Errors (LWE) problem, the specific algebraic correlation introduced by the multi-round iterative operations between the coefficient matrix and the accumulated noise makes a strict reduction to the standard LWE problem an extremely complex and non-trivial task in cryptography.

Given that the lattice-cryptography community has established that multiple matrix variants with special structures can still retain asymptotically equivalent hardness to standard LWE. In particular, the latest breakthrough work by Bangachev et al. [45] rigorously provides a super-polynomial lower bound for solving the Sparse LWE problem when the coefficient matrix is highly sparse (with only k nonzero elements per row). Based on this theoretical consensus, we formally define the following specific computational hardness assumption in this paper as the theoretical foundation for the security of the scheme:

Assumption A1 (Hardness Assumption for the Decision Problem of the Equivalent Matrix of Block Encryption): *Let A^M denote the specific diffusion matrix used in the scheme that is equivalent to the linear block encryption process in the proposed scheme, and $\vec{R}_{final}$ denote the accumulated amplified perturbation generated after n_r rounds of iteration. For any given plaintext vector $\vec{M} \in \mathbb{Z}_N^n$, define the following two distributions:*

1. ***Real Evolution Distribution:*** *the ciphertext generated by Eq. (A14).*

2. ***Uniform Random Distribution:*** *$U \leftarrow \mathbb{Z}_N^n$, that is, a completely uniformly random sampled vector.*

We make the following assumption: as long as the number of iterations n_r is sufficient to maintain the lower bound on the minimum sparsity of the matrix A^M, and the variance of the initially injected noise ensures that the standard deviation of the final accumulated error E_R satisfies the security threshold, then this multi-round correlated structure will not undergo catastrophic algebraic degeneration. For any probabilistic polynomial-time (PPT) adversary $\mathcal{A}$, we define its advantage in distinguishing the multi-round evolved noise during the encryption process as $\mathrm{Adv}_{\mathcal{A}}^{\text{MRNE-Dist}}(\lambda)$ We assume that this advantage is negligible, i.e., $\mathrm{Adv}_{\mathcal{A}}^{MRNE\text{-}Dist}(\lambda) \leq \mathrm{negl}(\lambda)$. Then the distinguishing advantage between the encrypted ciphertext and uniformly distributed random values is also negligible, i.e.,

$$|\Pr[\mathcal{A}(C) = 1] - \Pr[\mathcal{A}(U) = 1]| \leq \mathrm{negl}(\lambda). \tag{A21}$$

where C denotes the ciphertext and U denotes a uniformly random element over the same ciphertext space.

Security Model.

Definition A1 (IND-CPA Security of the Aggregation Scheme): *We define the indistinguishability under chosen-plaintext attack (IND-CPA) via the following game between a Challenger $\mathcal{C}$ and an Adversary $\mathcal{A}$:*

Challenge: *$\mathcal{A}$ chooses two distinct quantified gradient vectors g_0, g_1 of equal length and sends them to $\mathcal{C}$.*

Encryption: *$\mathcal{C}$ flips a random coin $b \in \{0,1\}$. It encrypts g_b using the proposed scheme, generating ciphertext C^*, and sends C^* to $\mathcal{A}$.*

Guess: *$\mathcal{A}$ outputs a guess $b' \in \{0,1\}$. The scheme is IND-CPA secure if for any probabilistic polynomial-time (PPT) adversary $\mathcal{A}$, the advantage*

$$\mathrm{Adv}_{\mathcal{A}}^{IND\text{-}CPA} = \left|\Pr[b' = b] - \frac{1}{2}\right|. \tag{A22}$$

is negligible.

Security Proof of the privacy-preserving aggregation mechanism.

Theorem A1: *Under Assumption A1, and that the accumulated noise $\vec{R}_{final}$ satisfies the Smudging Lemma, the proposed homomorphic encryption scheme achieves IND-CPA security.*

Proof: Let $\mathcal{A}$ be any PPT adversary, and we construct the following game sequence:

Game 0 (Real Attack Game): The challenger generates the challenge ciphertext according to the real protocol,

$$\vec{C} = A^M \cdot \vec{M} + R_{final} + A^{IV} I\vec{V} \pmod{N}. \tag{A23}$$

and gives it to the adversary $\mathcal{A}$.

Game 1: The challenger directly samples a uniform vector U at random from $\mathbb{Z}_N^n$ and sends it to $\mathcal{A}$ as the challenge ciphertext C^*.

Analysis: According to the Assumption A1, the truly generated noisy sparse linear ciphertext is computationally indistinguishable from a purely random vector. Therefore, the probability difference for the adversary in distinguishing Game 0 and Game 1 is strictly bounded by

$$|\Pr[\mathcal{A}(\text{Game } 0) = 1] - \Pr[\mathcal{A}(\text{Game } 1) = 1]| \le \mathrm{Adv}_{\mathcal{A}}^{\text{MRNE-Dist}}(\lambda) \le \mathrm{negl}(\lambda). \tag{A24}$$

In addition, by injecting sufficiently large initial variance and performing diffusion through n_r rounds, the standard deviation of the accumulated noise E_{final} is much larger than the maximum difference of the plaintext gradients, i.e., $\sigma \gg \|\Delta M\|_\infty$. According to the Smudging Lemma, the ciphertexts generated from different plaintexts are also statistically indistinguishable.

In Game 1, as the ciphertext is computationally indistinguishable from a uniform random value, the probability that the challenger wins differs from 1/2 by at most a negligible function of the security parameter. Therefore, the adversary's advantage in Game 0 is negligible, and the scheme achieves IND-CPA security. □

Appendix B Notation Table

Tables A1–A4 contain the notations used in this paper for the sections of federated learning, privacy-preserving aggregation mechanism, key agreement algorithm, experiments and security proof of the privacy-preserving aggregation mechanism, respectively.

Table A1: Notations of federated learning system.

Notation	Meaning
n	Number of participants
n_{select}	Number of participants selected per round
num_{para}	Number of parameters in the global model
$round_{fed}$	Number of global model rounds
bit_{float}	Number of bits of floating-point Numbers (plaintext) in the computer
$[[w_i g_i]]$	Encrypted weight
$l_{localstep}$	Number of local training rounds
bit_{quant}	Quantization precision
θ	Global model

Table A2: Notations of encryption algorithm.

Notation	Meaning	Notation	Meaning
$N_{\min}$	Minimum modulus to prevent overflow	nonce	Random state
N	Prime modulus used for encryption	l_{array}	Length of plaintext array
$\mathbb{Z}_N$	Ring of integers modulo N	l_{matrix}	Length of plaintext matrix list
α	Quantization boundary value	M_i	Original plaintext matrix (before encryption)
β	Noise boundary value	C_i	Completed ciphertext matrix
$\text{bit}^{\text{cipher}}$	Number of bits occupied by a single ciphertext	$C_{\text{disturb},i}$	Ciphertext after adding noise
n_r	Number of encryption rounds	$C_{\text{add},i}$	Ciphertext after round key addition
sk	Secret key for generating bitstream	$C_{\text{pcbc},i}$	Ciphertext after PCBC operation
$\text{size}_{\text{block}}$	Size of a single ciphertext block	$C_{\text{pi},i}$	Ciphertext after permutation
l_{string}	Length of bitstring for generating secret key	$C_{\text{dm},i}$	Ciphertext after obfuscation
l_π	Length of bitstring for generating P-box	$C_{\text{round},i}$	Ciphertext after one round of encryption
RK	Round key	$C^{'}$	Ciphertext encrypted up to $\left\lceil \frac{n_r}{2} \right\rceil$ rounds
R	Perturbation matrix	$C_{\text{cbc},i}$	Ciphertext after CBC operation
π	Permutation box (P-box)	$\text{round}_{\text{enc}}$	Number of encryption rounds
DM	Obfuscation matrix	$\text{order}_{\text{init}}$	Key index for PCBC
IV_{round}	Initial vector for round key	order_a	Key index for round key
IV_{cbc}	Initial vector for CBC	order_π	Index for permutation box π
IV_{pcbc}	Initial vector for PCBC	order_{dm}	Index for obfuscation matrix
IV_s	Set of all initial vectors		

Table A3: Notations of secret sharing.

Notation	Meaning
t	Threshold for reconstructing shares
X	Selected sequence of random numbers
x_i	A certain random number
s_i	A certain secret value
S	Secret array

(Continued)

Table A3 (continued)

Notation	Meaning
$f(\cdot)$	Polynomial used for secret sharing
l_{secret}	Length of the secret in secret sharing
$\text{share}_{i,j}$	Share sent from participant i to participant j
$\text{shares}_{\text{agged},i}$	Aggregated shares from other participants Constructed by participant i

Table A4: Notations of experiments and security proof.

Notation	Meaning	Notation	Meaning
sample_I	Sample category	$rk_{i,j}$	Element of RK matrix
v_i	Sample value	$dm_{i,j}$	Element of DM matrix
l_{vec}	Length of sample vector	$dm^{\text{inv}}_{i,j}$	Element of the inverse matrix of DM matrix
$\text{cate}_{\text{sample}}$	Total number of sample categories	$p_{i,j}$	Element of the equivalent matrix of the permutation matrix
$\mathbb{I}$	Indicator function	$\text{Round}_{i,j}(\cdot)$	One round of encryption, i, j denote the index of
$\times$	Matrix multiplication		The plaintext block and the encryption round number, respectively
$\odot$	Element-wise multiplication	A	The aggregated (or overall) matrix representing
$RK(\cdot)$	Element-wise multiplication of the round key matrix		The encryption operation on the vectors
$DM(\cdot)$	Matrix multiplication of the obfuscation matrix i	M_{scale}	Plaintext after PCBC operation and scaling
$\pi(\cdot)$	Permutation of matrix	cycle	Cycle of key rotation

References

1. McMahan B, Moore E, Ramage D, Hampson S, Arcas BAY. Communication-efficient learning of deep networks from decentralized data. In: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS 2017). Ft. Lauderdale, FL, USA: Society for Artificial Intelligence and Statistics; 2017. p. 1273–82.
2. Zhu L, Liu Z, Han S. Deep leakage from gradients. In: Proceedings of the 33rd International Conference on Neural Information Processing Systems. Red Hook, NY, USA: Curran Associates Inc.; 2019. p. 14774–14784. doi:10.5555/3454287.3455610.
3. Fan M, Zhang Z, Li Z, Sun G, Yu H, Guizani M. Blockchain-based decentralized and lightweight anonymous authentication for federated learning. IEEE Trans Veh Technol. 2023;72(9):12075–86. doi:10.1109/tvt.2023.3265366.
4. Kumbhar HR, Rao SS. Federated learning enabled multi-key homomorphic encryption. Expert Syst Appl. 2025;268(C):126197. doi:10.1016/j.eswa.2024.126197.
5. Truex S, Liu L, Chow KH, Gursoy ME, Wei W. LDP-Fed: federated learning with local differential privacy. In: EdgeSys '20: Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking. New York, NY, USA: ACM; 2020. p. 61–6. doi:10.1145/3378679.3394533.
6. Fang H, Qian Q. Privacy preserving machine learning with homomorphic encryption and federated learning. Future Internet. 2021;13(4):94. doi:10.3390/fi13040094.
7. Bonawitz K, Ivanov V, Kreuter B, Marcedone A, McMahan HB, Patel S, et al. Practical secure aggregation for privacy-preserving machine learning. In: CCS '17: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. New York, NY, USA: ACM; 2017. p. 1175–91. doi:10.1145/3133956.3133982.

8. Marcoll C, Sucasas V, Manzano M, Bassoli R, Fitzek FTP, Aaraj N. Survey on fully homomorphic encryption, theory, and applications. Proc IEEE. 2022;110(10):1572–609.
9. Yang Q, Liu Y, Chen T, Tong Y. Federated machine learning: concept and applications. ACM Trans Intell Syst Technol. 2019;10(2):12:1–19. doi:10.1145/3298981.
10. Ma J, Naas SA, Sigg S, Lyu X. Privacy-preserving federated learning based on multi-key homomorphic encryption. Intl J Intell Syst. 2022;37(9):5880–901. doi:10.1002/int.22818.
11. Chen H, Dai W, Kim M, Song Y. Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference. In: CCS '19: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. New York, NY, USA: ACM; 2019. p. 395–412. doi:10.1145/3319535.3363207.
12. Sun J, Chen T, Giannakis GB, Yang Q, Yang Z. Lazily aggregated quantized gradient innovation for communication-efficient federated learning. IEEE Trans Pattern Anal Mach Intell. 2022;44(4):2031–44. doi:10.1109/tpami.2020.3033286.
13. Rothchild D, Panda A, Ullah E, Ivkin N, Stoica I, Braverman V, et al. FetchSGD: communication-efficient federated learning with sketching. In: Proceedings of the 37th International Conference on Machine Learning. Cambridge, MA, USA: PMLR; 2020. p. 8253–65.
14. Jhunjhunwala D, Gadhikar A, Joshi G, Eldar YC. Adaptive quantization of model updates for communication-efficient federated learning. In: 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Piscataway, NJ, USA: IEEE; 2021. p. 3110–4. doi:10.1109/ICASSP39728.2021.9413697.
15. Hu C, Li B. MaskCrypt: federated learning with selective homomorphic encryption. IEEE Trans Dependable Secur Comput. 2025;22(1):221–33. doi:10.1109/tdsc.2024.3392424.
16. Singh A, Rathee G, Kerrache CA, Ghanem MC. A relay-chain–powered ciphertext-policy attribute-based encryption in intelligent transportation systems. Transp Eng. 2026;23(1):100424. doi:10.2139/ssrn.5397593.
17. Zhang C, Li S, Xia J, Wang W, Yan F, Liu Y. BatchCrypt: efficient homomorphic encryption for cross-silo federated learning. In: USENIX ATC'20: Proceedings of the 2020 USENIX Conference on Usenix Annual Technical Conference. Berkeley, CA, USA: USENIX Association; 2020. p. 493–506.
18. Zhang L, Xu J, Vijayakumar P, Sharma PK, Ghosh U. Homomorphic encryption-based privacy-preserving federated learning in IoT-enabled healthcare system. IEEE Trans Netw Sci Eng. 2023;10(5):2864–80. doi:10.1109/tnse.2022.3185327.
19. Burlachenko K, Alrowithi A, Albalawi FA, Richtárik P. Federated learning is better with non-homomorphic encryption. In: DistributedML '23: Proceedings of the 4th International Workshop on Distributed Machine Learning. New York, NY, USA: ACM; 2023. p. 49–84. doi:10.1145/3630048.3630182.
20. Pan Y, Chao Z, He W, Jing Y, Hongjia L, Liming W. FedSHE: privacy preserving and efficient federated learning with adaptive segmented CKKS homomorphic encryption. Cybersecurity. 2024;7(1):40. doi:10.1186/s42400-024-00232-w.
21. Chait K, Laouid A, Laouamer L, Kara M. A multi-key based lightweight additive homomorphic encryption scheme. In: Proceedings of the 2021 International Conference on Artificial Intelligence for Cyber Security Systems and Privacy (AI-CSP). Piscataway, NJ, USA: IEEE; 2021. p. 1–6. doi:10.1109/AI-CSP52968.2021.9671216.
22. Ming Y, Wang S, Wang C, Liu H, Deng Y, Zhao Y, et al. VCSA: verifiable and collusion-resistant secure aggregation for federated learning using symmetric homomorphic encryption. J Syst Arch. 2024;156:103279.
23. Hariss K, Noura H. Homomorphic additive lightweight block cipher scheme for securing IoT applications. J Inf Secur Appl. 2023;76(C):103540. doi:10.1016/j.jisa.2023.103540.
24. Azzouzi O, Anane M, Ghanem MC, Himeur Y, Kheddar H. Efficient fine-grained LuT-based optimization of AES MixColumns and InvMixColumns for FPGA implementation. Electronics. 2025;14(24):4912. doi:10.3390/electronics14244912.
25. Park J, Lim H. Privacy-preserving federated learning using homomorphic encryption. Appl Sci. 2022;12(2):734. doi:10.3390/app12020734.
26. Jiang ZL, Guo H, Pan YJ, Liu Y, Wang X, Zhang J. Secure neural network in federated learning with model aggregation under multiple keys. In: Proceedings of the 2021 8th IEEE International Conference on Cyber Security

and Cloud Computing (CSCloud)/2021 7th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom). Piscataway, NJ, USA: IEEE; 2021. p. 47–52. doi:10.1109/CSCloud-EdgeCom52276.2021.00019.
27. Cai Y, Ding W, Xiao Y, Yan Z, Liu X, Wan Z. SecFed: a secure and efficient federated learning based on multi-key homomorphic encryption. IEEE Trans Dependable Secur Comput. 2024;21(4):3817–33.
28. Pham CH, Huynh-The T, Sedgh-Gooya E, El-Bouz M, Alfalou A. Extension of physical activity recognition with 3D CNN using encrypted multiple sensory data to federated learning based on multi-key homomorphic encryption. Comput Methods Programs Biomed. 2024;243(2):107854. doi:10.1016/j.cmpb.2023.107854.
29. Li Y, Lai J, Zhang R, Sun M. Secure and efficient multi-key aggregation for federated learning. Inf Sci. 2024;654(2):119830. doi:10.1016/j.ins.2023.119830.
30. Walskaar I, Tran MC, Catak FO. A practical implementation of medical privacy-preserving federated learning using multi-key homomorphic encryption and flower framework. Cryptography. 2023;7(4):48. doi:10.3390/cryptography7040048.
31. Wu G, Xu S, He D, Chan S. Blockchain-based efficient and secure cloud cross-domain data sharing with dynamic revocation by multiple authorities. Comput Netw. 2025;270(2):111518. doi:10.1016/j.comnet.2025.111518.
32. Tian Y, Fan Z, Zhang Y. Toward secure and auditable data sharing: a cross-chain CP-ABE framework. Comput Mater Contin. 2026;87(1):62. doi:10.32604/cmc.2025.073935.
33. Kong X, Vellayappan K, Chin JT, Khan SA, Wu Z. Homomorphically encrypted neural network modeling and predictive control for data-privacy-preserving control systems. Control Eng Pract. 2026;171(8):106848. doi:10.1016/j.conengprac.2026.106848.
34. Noura HN, Salman O, Couturier R, Chehab A. LoRCA: lightweight round block and stream cipher algorithms for IoV systems. Veh Commun. 2022;34:100416.
35. Hariss K, Noura H. Towards a fully homomorphic symmetric cipher scheme resistant to plain-text/cipher-text attacks. Multimed Tools Appl. 2022;81(10):14403–49. doi:10.1007/s11042-022-12043-7.
36. Shamir A. How to share a secret. Commun ACM. 1979;22(11):612–3. doi:10.1145/359168.359176.
37. Lin Y, Han S, Mao H, Wang Y, Dally W. Deep gradient compression: reducing the communication bandwidth for distributed training. arXiv:1712.01887v2. 2018.
38. Zhang J, Liu Y, Hua Y, Wang H, Song T, Xue Z, et al. PFLlib: a beginner-friendly and comprehensive personalized federated learning library and benchmark. arXiv:2312.04992. 2025.
39. Benaissa A, Retiat B, Cebere B, Belfedhal AE. TenSEAL: a library for encrypted tensor operations using homomorphic encryption. arXiv:2104.03152. 2021.
40. Serengil SI, Ozpinar A. LightPHE: integrating partially homomorphic encryption into python with extensive cloud environment evaluations. arXiv:2408.05219. 2025.
41. Krizhevsky A, Nair V, Hinton G. Learning multiple layers of features from tiny images [Dataset]. 2009 [cited 2025 Nov 13]. Available from: https://www.cs.toronto.edu/.
42. Rai AA. AG news classification dataset [Dataset]. Kaggle datasets. 2021 [cited 2025 Nov 15]. Available from: https://www.kaggle.com/datasets/amananandrai/ag-news-classification-dataset.
43. Reiss A. PAMAP2 physical activity monitoring. UCI machine learning repository. 2012 [cited 2025 Nov 15]. Available from: http://archive.ics.uci.edu/dataset/231/pamap2+physical+activity+monitoring.
44. Radwan AG, AbdElHaleem SH, Abd-El-Hafiz SK. Symmetric encryption algorithms using chaotic and non-chaotic generators: a review. J Adv Res. 2016;7(2):193–208. doi:10.1016/j.jare.2015.07.002.
45. Bangachev K, Bresler G, Tiegel S, Vaikuntanathan V. Near-optimal time-sparsity trade-offs for solving noisy linear equations. In: Proceedings of the 57th Annual ACM Symposium on Theory of Computing, STOC '25. New York, NY, USA: ACM; 2025. p. 1910–20. doi:10.1145/3717823.3718284.