Journal of Information Security and Applications 102 (2026) 104559

Contents lists available at ScienceDirect

Journal of Information Security and Applications

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

FedDC: Efficient protection scheme based on chaotic system in federated learning

Yihan Liao [a], Jacky Keung [a], Jingyu Zhang [b,*], Yurou Dai [c]

[a] City University of Hong Kong, Hong Kong, China
[b] Hong Kong Metropolitan University, Hong Kong, China
[c] Lehigh University, Bethlehem, Pennsylvania, USA

ARTICLE INFO

Keywords:
Federated learning
Privacy-preserving
Chaotic system

ABSTRACT

Federated Learning (FL) enables collaborative model training across decentralized clients while keeping raw data local. However, existing privacy-preserving mechanisms, such as secure aggregation and differential privacy (DP), often introduce significant computational overhead or degrade model utility. To address this challenge, we propose *FedDC*, a lightweight FL framework that combines DP with chaos-based parameter scrambling. Unlike existing approaches that uniformly protect entire model updates, *FedDC* introduces selective and layer-aware protection for sensitive neural network parameters, enabling flexible privacy protection with minimal overhead. Extensive experiments on four image and text datasets show that *FedDC* effectively mitigates privacy leakage under both black-box and white-box attacks while maintaining competitive model accuracy and negligible computational overhead.

1. Introduction

The rapid advancement of artificial intelligence (AI) has significantly accelerated the progress of computer science, leading to widespread applications in natural language processing [1], computer vision [2], and network security [3]. Machine Learning (ML) is an analysis and processing method that relies on large volumes of data, underscoring the pivotal role of data in driving ML performance. Traditionally, ML relies on centralized architectures, where locally stored data is transmitted to centralized servers for training. However, this paradigm raises critical privacy concerns, such as personal medical records [4,5], which become vulnerable to unauthorized access. Growing awareness of data protection has prompted regulations such as the General Data Protection Regulation (GDPR) [6], which imposes limitations on data sharing, necessitating innovative approaches to data privacy in ML tasks. Against this backdrop, Federated Learning (FL) has emerged as a pioneering framework that addresses privacy concerns by enabling decentralized entities to collaboratively train shared models without exchanging raw data [7,8]. In FL, participants locally train the model parameters and upload only model updates to a centralized server for aggregation. The server then distributes the aggregated global model back to the participants, allowing further local training. This iterative process ensures that private data is retained locally throughout the training process [9], enhancing the appeal of FL in security-critical domains such as healthcare, blockchain applications, and the Internet of Things (IoT) [10–12].

Nevertheless, even with mitigated direct data exposure, FL remains vulnerable to privacy threats such as model leakage. Recent studies have demonstrated that adversaries can exploit model gradients to recover sensitive input data [13]. These attacks pose significant risks in multi-party FL settings, where adversarial participants may compromise privacy by leveraging internal model parameters or shared gradients. To address these vulnerabilities, several categories of privacy-preserving techniques have been explored in FL. The most widely adopted approaches include differential privacy (DP) [14], secure aggregation (SA) [15], and homomorphic encryption (HE) [16]. DP introduces carefully calibrated noise into local updates to provide statistical privacy guarantees, while SA protects individual client updates by only revealing their aggregated results to the server. HE further enables computation over encrypted data without exposing plaintext model parameters. However, these methods often suffer from notable limitations. DP-based approaches typically degrade model accuracy under strong privacy constraints, while SA- and HE-based methods, or DP-hybrid encryption approaches [17], may introduce significant communication or computational overhead. Moreover, most existing approaches apply uniform protection to all model parameters, without accounting for the varying sensitivities and structural roles of different neural network layers.

* Corresponding author.
E-mail addresses: yihanliao4-c@my.cityu.edu.hk (Y. Liao), jacky.keung@cityu.edu.hk (J. Keung), fzhang@hkmu.edu.hk (J. Zhang), yrd224@lehigh.edu (Y. Dai).

https://doi.org/10.1016/j.jisa.2026.104559

Available online 1 July 2026

Our contribution. Motivated by these challenges, we propose *FedDC*. This novel FL framework combines DP with chaos-based scrambling to enhance privacy protection while maintaining high model performance and computational efficiency. Chaotic systems, characterized by extreme sensitivity to initial conditions and intrinsic dynamic complexity, offer distinct advantages for scrambling. Small perturbations in initial states lead to vastly different system evolutions, and the unpredictable, pseudo-random nature of chaotic maps renders protected data resistant to privacy leakage.

In *FedDC*, DP introduces controlled noise to mitigate gradient leakage, while chaotic scrambles selected model parameters through dynamic mappings, preventing the server from inferring sensitive client information. To comprehensively assess privacy resilience, we design two adversarial settings aligned with the *FedDC* workflow. In the black-box scenario, an attacker accesses the protected global model without the inverse scrambling key, simulating protected model leakage or misuse. In the white-box scenario, a stronger adversary is assumed to have access to the model architecture and the protected parameters uploaded across multiple rounds, and is aware of the use of the chaotic scrambling mechanism. Empirical results demonstrate that *FedDC* effectively mitigates both threats while maintaining model utility. Furthermore, *FedDC* supports selective layer scrambling, offering flexible trade-offs between privacy protection and computational efficiency. Overall, *FedDC* provides a dual-layer defense that combines noise-based guarantees with chaotic scrambling, where its key innovations lie in a coordinated dual-layer privacy design, a selective layer-wise protection strategy, and a lightweight alternative to scrambling, enabling a balanced trade-off among privacy, efficiency, and model utility.

Specifically, our main contributions are as follows:

- We propose *FedDC*, a lightweight privacy-preserving FL framework that jointly integrates differential privacy and chaos-based scrambling, providing a dual-layer protection mechanism against privacy leakage.
- We introduce a selective layer-wise protection strategy, which enables adaptive scrambling of sensitive model components rather than uniformly protecting all parameters, thereby achieving a more flexible trade-off between privacy and efficiency.
- We conduct extensive experiments under both black-box and white-box adversarial settings, demonstrating that *FedDC* effectively mitigates privacy leakage while maintaining competitive model accuracy and low computational overhead.

2. Background and related work

2.1. Federated learning

FL was first proposed by Konečný et al. [18], offering a decentralized ML paradigm in which clients retain their local data and collaboratively train a shared model by transmitting only local updates. This design reduces the need for direct data sharing and has therefore been widely adopted in privacy-sensitive domains [7]. However, keeping raw data local does not fully eliminate privacy risks. In practice, shared gradients or model parameters may still leak sensitive information, enabling adversaries to infer client data or reconstruct inputs from intermediate training artifacts [19].

To mitigate such risks, prior privacy-preserving FL research has mainly followed three directions. The first direction is DP, which injects statistical noise into local updates to provide formal privacy guarantees [20]. The second direction is SA, which prevents the server from observing individual client updates by only revealing their aggregate [21]. The third direction is HE, which allows aggregation or computation to be performed directly over encrypted updates without exposing plaintext parameters [22,23]. In addition, hybrid approaches that combine DP with encryption mechanisms have been proposed to jointly leverage statistical and cryptographic protection. For example, ADPHE-FL [17] integrates adaptive DP with HE to provide dual-layer privacy protection. Furthermore, model fingerprinting and other parameter protection techniques have also been explored to reduce model leakage risks [24].

Although these directions have advanced privacy-preserving FL significantly, they reflect different trade-offs. DP offers statistical privacy guarantees but often reduces model utility when strong privacy is required [25]. SA protects individual updates from server inspection, yet it introduces nontrivial protocol and communication overhead, especially in large-scale settings [15]. While HE-based methods provide strong confidentiality guarantees by enabling computation over encrypted data, they typically incur significant computational and communication overhead, making them less efficient in large-scale or resource-constrained federated learning scenarios [17,26]. Therefore, existing methods do not equally satisfy privacy, efficiency, and model performance. This motivates exploring lightweight hybrid protection mechanisms that can strengthen privacy while remaining practical for resource-constrained FL environments.

2.2. Differential privacy

DP is a widely adopted technique for limiting the disclosure of sensitive information by introducing carefully calibrated randomness into released outputs or model updates [27]. In FL, DP is commonly applied on the client side before transmission, so that uploaded updates reveal less information about any individual data record [28,29]. Because of its formal privacy interpretation, DP has become one of the most representative defenses against privacy leakage and inference attacks in FL.

Geyer et al. [30] were among the earliest to study client-level DP in FL, showing that noisy local updates can significantly reduce the risk of privacy leakage. Choudhury et al. [31] further extended DP-based FL to healthcare settings by introducing local objective perturbation. Despite these advantages, prior studies consistently report a fundamental privacy and utility trade-off: stronger privacy typically requires more aggressive noise injection, which can degrade convergence and final model accuracy [20,25,32]. This issue becomes particularly pronounced in heterogeneous FL settings, where local data are often non-IID, and client data volumes are limited.

For this reason, *FedDC* does not treat DP as a standalone defense. Instead, DP serves as one layer in a hybrid protection design. It provides statistical perturbation against inference from local updates, while chaos-based scrambling further reduces the usability of exposed model parameters. This combination is intended to alleviate the limitations of DP-only solutions by distributing the privacy burden across complementary protection mechanisms rather than relying exclusively on noise injection.

2.3. Chaotic scrambling

Chaos theory describes the complex and highly sensitive behavior of nonlinear dynamical systems. Due to their sensitivity to initial conditions, chaotic systems can generate pseudo-random sequences with low computational cost, making them attractive for lightweight protection mechanisms [33]. In particular, logistic maps are widely used in chaos-based scrambling schemes because they can efficiently generate highly irregular sequences from a small number of parameters.

Recent studies have explored integrating chaotic mechanisms into distributed learning frameworks to provide lightweight privacy protection. For example, Arévalo et al. [34] combined chaotic maps with DP to protect distributed deep learning under incomplete and non-IID data. However, existing chaos-based FL approaches typically apply uniform transformations to model updates or protect the model as a whole, without distinguishing the sensitivity of different neural network layers or considering flexible protection granularity.

Compared with these approaches, *FedDC* differs in three key aspects. First, unlike prior chaos-based FL methods that typically apply uniform transformations to the entire model update, *FedDC* introduces selective

chaos-based scrambling over sensitive model components. Second, in contrast to DP-only or loosely coupled hybrid approaches, *FedDC* integrates chaotic scrambling with DP in a complementary manner, where DP mitigates statistical leakage while chaotic scrambling reduces the usability of exposed parameters. Third, compared with cryptographic approaches such as SA and HE, *FedDC* provides a lightweight protection mechanism with significantly lower computational overhead.

In summary, existing privacy-preserving FL approaches either rely on noise-based perturbations that may degrade utility or employ heavy cryptographic mechanisms that incur substantial overhead. Although chaotic scrambling has been explored, prior work typically applies uniform transformations and does not consider selective protection or its integration with DP. This gap motivates the design of *FedDC*, which combines lightweight chaotic scrambling with DP-based perturbation and introduces layer-aware protection for a more flexible privacy-utility trade-off.

3. Preliminaries

3.1. Federated learning

In this work, we follow the standard formulation where the global objective is to minimize a weighted sum of local objective functions across K clients:

$$\min_{w} F(w) = \sum_{k=1}^{K} \frac{n_k}{N} F_k(w), \tag{1}$$

$$w_k^{(t+1)} = w_k^t - \eta \nabla F_k(w), \tag{2}$$

where N is the total number of training samples across all clients, n_k is the local dataset size of the k^{th} client, and $F_k(w) = \frac{1}{n_k} \sum_{i \in d_k} f_i(w)$ represents the local objective on client k's dataset d_k. $f_i(w)$ denotes the loss of model parameters w on data point (x_i, y_i). After local training, model updates are transmitted to the central server, which aggregates them by the FedAvg algorithm [35], which is a widely adopted aggregation strategy in FL by computing a weighted average of client updates, where the weight is proportional to the size of each client's local dataset. The formal definition is:

$$w^t = \sum_{k=1}^{K} \frac{n_k}{N} w_k^t. \tag{3}$$

Through iterative local updates and global aggregation, FL progressively refines the shared model without compromising local data privacy.

3.2. ϵ-differential privacy

DP provides a rigorous framework for quantifying the privacy guarantees of algorithms that handle sensitive data. Considering that FL often assumes no trusted third party and that distributed DP imposes high computational costs [36], our scheme adopts local DP. The formal definition of ϵ-DP is:

$$Pr[M(x) \in S] \leq e^{\epsilon} Pr[M(x') \in S], \tag{4}$$

where x and x' are adjacent datasets differing in one element, $M(\cdot)$ is a randomized mechanism, S is any subset of possible outputs, and ϵ is the privacy budget. A smaller ϵ implies stronger privacy guarantees, ensuring that the outputs are statistically indistinguishable even when one data point changes.

In practice, a common mechanism to achieve DP is the Laplace mechanism, which adds noise drawn from the Laplace distribution to the output of the target function. Given a function f with sensitivity Δf, the perturbed output $\tilde{f}(x)$ under the Laplace mechanism is defined as:

$$\tilde{f}(x) = f(x) + \text{Lap}\left(0, \frac{\Delta f}{\epsilon}\right), \tag{5}$$

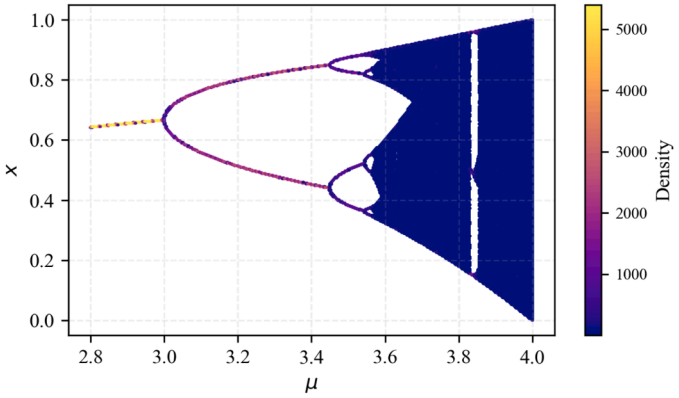


Fig. 1. Dynamics of the chaotic map across parameter μ.

where $\text{Lap}\left(0, \frac{\Delta f}{\epsilon}\right)$ denotes a Laplace distribution centered at 0 with scale parameter $\frac{\Delta f}{\epsilon}$. Δf denotes the sensitivity, to signify the variation in the function's output upon altering the input by a unit value. ϵ represents the privacy budget, which inversely correlates with data availability. The smaller the ϵ, the less data availability but the better privacy protection. In *FedDC*, the Laplace mechanism is selectively applied to model parameters after local training, forming a parameter-level local DP mechanism.

Due to the effectiveness of DP in obfuscating model parameters to defend against differential attacks, *FedDC* enables the selective introduction of DP and the adjustment of the noise range according to the accuracy requirements. If the prioritization of model utility takes precedence over the need for data privacy, it is feasible to either diminish the DP noise level or entirely forego the utilization of DP.

3.3. Chaotic system

Chaos theory describes the deterministic yet highly sensitive behavior of nonlinear dynamical systems. A widely studied chaotic map is the logistic map, mathematically defined as:

$$x_{n+1} = \mu x_n (1 - x_n), \quad \mu \in (0, 4], \quad x_n \in (0, 1), \tag{6}$$

where μ is the control parameter, x_0 is the initial seed, and x_n denotes the system state at iteration n.

Fig. 1 illustrates the principle of chaotic mapping. Depending on the value of μ, the logistic map exhibits distinct dynamical behaviors. When $0 < \mu < 1$, the system rapidly converges to a zero steady state. As μ increases into the range $1 < \mu < 3$, the system transitions into a periodic regime characterized by stable oscillations. Further increasing μ beyond 3 induces bifurcations, and when $3 < \mu < 4$, the system displays chaotic behavior, where small perturbations in the initial conditions result in vastly divergent trajectories. This progression from stability to chaos with increasing μ highlights the logistic map's sensitivity to parameter variations and its suitability for generating pseudo-random sequences. Additionally, color intensity reflects the density of points at each μ value, indicating how frequently certain x values occur. Darker regions correspond to higher densities, indicating stable attractors or common states that emerge under repeated iterations.

Leveraging the inherent unpredictability and low computational requirements of chaotic maps, particularly logistic maps, enables efficient key generation or scrambling sequences. These properties are highly suitable for FL scenarios, where resource limitations and privacy requirements converge.

4. Methodology

In this section, we first explain the major symbols used in this scheme, as listed in Table 1, and then introduce the *FedDC*.

Table 1
Major symbols explanation.

Symbols	Descriptions
P_i, P_i^s	The i^{th} model parameter and its subsets
D_i^s	Noise distribution for every P_i^s
μ, x_0	Control parameter and initial seed of the chaotic system
M	Chaotic logistic mapping
V_c	Convolutional layer in CNN model
H_l	Fully connected layer in CNN model
W_v	Self-attention layer in BERT model
θ	Protected parameters in convolutional layer
γ	Protected parameters in fully connected layer
K	Kernel size
L	Segment length for scrambling
N_s	Number of subsets to be protected
C	Chaotic random sequences for scrambling
C_{bs}	Batch size of C
T_p	List of protected position
T_x	List of synchronized seeds $x_0^{(s)}$ for selected segments

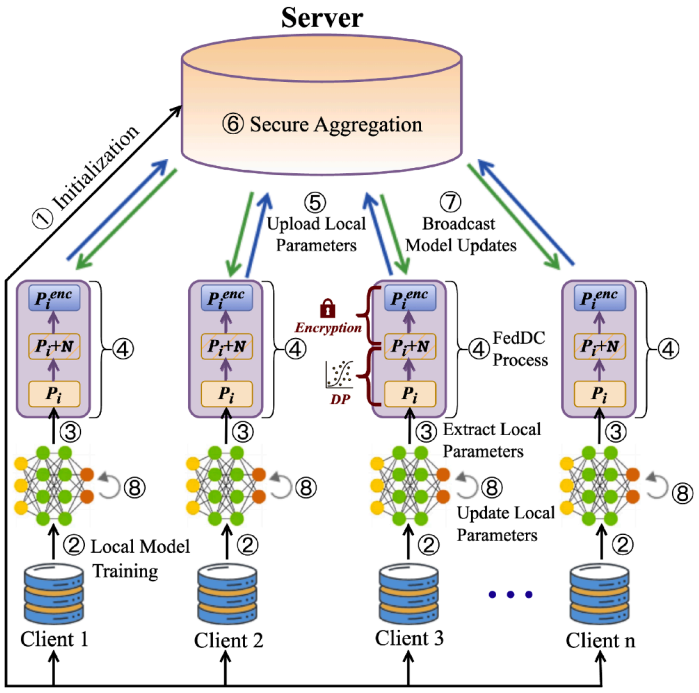


Fig. 2. The overview of FedDC.

4.1. FedDC

This section presents the design of the *FedDC* framework, which aims to provide lightweight privacy protection for FL while maintaining model utility and computational efficiency. *FedDC* adopts a hybrid protection strategy that combines DP with selective chaos-based parameter scrambling. This design enables flexible protection of sensitive model components while avoiding the high overhead associated with heavyweight cryptographic methods.

We assume a commonly used threat model in FL where the server and participating clients are honest-but-curious [37,38]. In this setting, all parties follow the training protocol correctly but may attempt to infer sensitive information from the data or model updates they observe. We assume no collusion between the server and clients or among clients, as the primary focus of this work is to mitigate information leakage from shared model parameters. To clarify the adversarial capabilities, we consider two representative attack scenarios.

Black-box adversary. The attacker can access the protected global model (e.g., through model leakage or interception) but does not possess the scrambling parameters. The attacker cannot inspect internal parameter structures and can only interact with the model through its outputs [38]. The objective is to exploit the model for prediction or to infer sensitive information without inverse scrambling.

White-box adversary. The attacker has access to the global model architecture and the transmitted model parameters before inverse scrambling, representing a strong adversarial setting. Specifically, the attacker can observe the protected parameters exchanged between clients and the server and has full knowledge of the model structure and training procedure. Although the attacker does not possess the inverse scrambling key needed to recover the original parameter ordering, prior work has shown that model parameters and updates can still leak sensitive information [39]. In this setting, the attacker attempts to exploit observable parameters to infer sensitive patterns or approximate training data (e.g., model inversion), despite lacking direct access to plaintext parameters.

Under this threat model, *FedDC* focuses on reducing privacy leakage from shared parameters while preserving collaborative training effectiveness. A key design choice is to aggregate model parameters rather than gradients, which reduces direct exposure of instance-level information and improves robustness against inference attacks [40].

The architecture of *FedDC*, consisting of multiple clients and a centralized server, is illustrated in Fig. 2. *FedDC* establishes a TLS/SSL secure communication channel for each client to ensure the integrity of transmitted data. During the *Initialization* (①) phase, one client and the server collaboratively determine the round-wise chaotic control parameter used to synchronize scrambling patterns across participating clients. The synchronized segment-level seeds are then generated in each communication round and shared among participating clients through the protocol, while remaining hidden from the server.

During *Local Model Training* (②), each client trains the model on its local dataset and updates the local model parameters. In the *Extract Local Parameters* (③) step, each client extracts parameters P_i and applies the *FedDC* protection mechanism (④) where Laplace noise is first injected to obtain perturbed parameters P_i^*, after which chaos-based scrambling is applied to produce protected parameters P_i^{enc}. This privacy-preserving transformation is performed entirely on the client side before any communication with the server.

In the *Upload Local Parameters* (⑤) step, the protected parameters P_i^{enc} are transmitted to the server. The server randomly samples m clients and directly aggregates the protected parameters using the FedAvg algorithm, without performing inverse scrambling. The updated global model is then distributed to all clients in the *Broadcast Model Updates* (⑦) step. Finally, clients perform the corresponding de-scrambling operation and update their local models using the received parameters in the *Update Local Parameters* (⑧) step.

4.2. Differential privacy

In *FedDC*, a lightweight DP-based perturbation mechanism is applied on the client side before the scrambling process to mitigate information leakage from local model updates. Specifically, each client perturbs its local model parameters by injecting calibrated noise. To control the magnitude of local updates, model parameters are first clipped within a predefined norm bound:

$$\tilde{P}_i^{k+1} = \frac{P_i^{k+1}}{\max\left(1, \frac{\|P_i^{k+1}\|_2}{C}\right)}, \tag{7}$$

where P_i^{k+1} denotes the local model parameters of client i after the $(k+1)$th communication round, $\tilde{P}_i^{k+1}$ represents the clipped parameters, $\|\cdot\|_2$ denotes the ℓ_2-norm, and C is a predefined threshold that bounds the magnitude of updates. This clipping operation bounds the global sensitivity of the local update to $\Delta f = C$. After clipping, noise is injected into the parameters as follows: $\overline{P}_i^{k+1} = \tilde{P}_i^{k+1} + n$, where n is sampled from a Laplace distribution: $n \sim \text{Laplace}\left(0, \frac{C}{\epsilon}\right)$, and ϵ controls

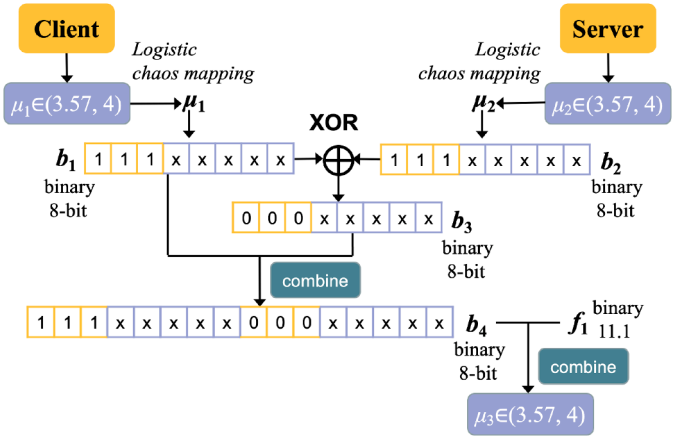


Fig. 3. The mechanism of generating parameter μ.

the scale of the perturbation. A smaller ϵ results in stronger noise and higher privacy protection, while a larger ϵ preserves more model utility. This perturbation mechanism reduces information leakage by limiting the influence of local updates and introducing randomness into the parameter space. It serves as a lightweight local protection strategy that enhances resistance to inference attacks while maintaining model performance. In *FedDC*, this perturbation acts as the first layer of defense by reducing statistical leakage from model updates. The subsequent chaos-based scrambling further disrupts parameter structures, forming a complementary dual-layer protection mechanism.

4.3. Scrambling

In this section, we detail the chaos-based scrambling component of *FedDC*. We first introduce the chaos-based scrambling mechanism used to secure client model parameters before they are uploaded. Then, we analyze the computational complexity of this encryption scheme and examine the privacy-performance trade-offs it entails in FL.

4.3.1. Scrambling mechanism

FedDC utilizes chaotic systems to generate deterministic yet pseudo-random permutations for parameter scrambling. As delineated in Eq. (6), the critical parameters involved in the chaotic mapping process are μ and x_0, which are set during the initialization phase. The collaborative generation of μ ensures that the scrambling control parameter is jointly determined rather than controlled by a single party, improving robustness against unilateral manipulation. The detailed procedure for the initialization of the parameter μ is illustrated in Fig. 3.

In each communication round, the server and a randomly selected client independently generate chaotic parameters μ_1 and μ_2 within the interval $[3.57, 4]$. These values are converted into binary representations and combined through a bitwise XOR operation to produce the final control parameter μ, as illustrated in Fig. 3. For each communication round, a set of synchronized seeds $x_0^{(s)}$ is generated for the selected segments. To prevent fixed scrambling patterns, new round-wise seeds are used in different communication rounds, ensuring that the scrambling structure varies over time and reducing the risk of cross-round correlation attacks.

In *FedDC*, we apply the proposed scrambling mechanism to both Convolutional Neural Network (CNN) and Transformer-based models. The design of the selective layer protection mechanism is motivated by the observation that different layers encode information with different levels of privacy sensitivity. Early layers generally capture low-level feature representations, while task-specific layers preserve more discriminative information that contributes directly to model prediction. As a result, exposing parameters from these structurally critical layers may provide more useful information for reconstruction or inference attacks.

Algorithm 1: ChaosEncrypt (Client-side, per round; DP first, then chaotic scrambling).

Input: μ (round-wise chaotic parameter), $\tilde{\theta}$ (DP-perturbed parameter vector), N_s (number of segments), L (segment length), C_{bs} (chaos batch size)
Output: θ^{enc}, T_p (segment starts), T_x (list of x_0), C (chaos sequence), Π (scrambling)

Preconditions: ensure $N_s \cdot L \le |\tilde{\theta}|$.
Select N_s start indices $T_p = \{p_s\}$ (e.g., evenly spaced in $[0, |\tilde{\theta}| - L]$);
Initialize $T_x = \emptyset$, $C = \emptyset$, $\Pi = \emptyset$;
$\theta^{enc} \leftarrow \tilde{\theta}$;
for $s = 1$ **to** N_s **do**
  Use the synchronized round-specific seed $x_0^{(s)}$ for segment s; append to T_x;
  Generate a chaos sequence $C^{(s)}$ of length L using logistic map with $(\mu, x_0^{(s)})$;
  Compute a stable rank-scrambling $\pi_s \leftarrow \mathrm{argsort}(C^{(s)})$ (tie-break by index);
  Save $C^{(s)}$ into C and π_s into Π;
  Apply scrambling on window $[p_s : p_s + L - 1]$:
    $\theta^{enc}_{[p_s:p_s+L-1]} \leftarrow \theta^{enc}_{[p_s:p_s+L-1]}[\pi_s]$;
return θ^{enc}, T_p, T_x, C, Π

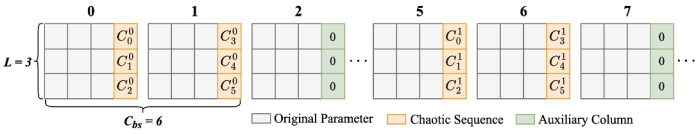


Fig. 4. Example of chaotic segment selection and batch configuration.

Prior studies have shown that sensitive information can be reconstructed from parameters of early network layers, even when only a small fraction of parameters is exposed [41,42]. Based on this observation, *FedDC* selects protection targets according to two criteria: (1) the contribution of a layer to feature representation and prediction, and (2) the potential privacy leakage risk associated with its parameters. Therefore, instead of uniformly protecting the entire model, *FedDC* focuses on scrambling parameters from sensitive layers, such as the first convolutional layer and fully connected layers in CNNs, and the value projection matrix in BERT. The same principle is applicable to different neural architectures. Although the specific layers vary between CNN-based and Transformer-based models, the selection criterion remains consistent: protecting layers that contain highly informative representations while avoiding unnecessary overhead from full-model protection. Different layer combinations can provide different privacy-utility trade-offs. Protecting more layers generally increases the protection coverage but introduces additional computation and communication overhead. Therefore, *FedDC* adopts selective protection to balance privacy enhancement and model efficiency.

Algorithm 1 summarizes the client-side scrambling process. For each selected segment, *FedDC* generates a chaos sequence from $(\mu, x_0^{(s)})$, derives a rank-based scrambling, and applies it to the corresponding parameter subset. Since the operation is applied independently to N_s segments, the protection strength can be flexibly adjusted by varying N_s and L. Fig. 4 illustrates how segment-level scrambling is applied to selected parameter subsets. Given the segment length L, chaotic batch size C_{bs}, and selected positions T_p, only sensitive local regions are permuted while the remaining parameters remain unchanged. This selective design reduces computational overhead compared with full-parameter protection.

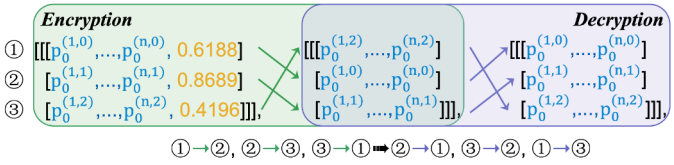


Fig. 5. Inverse scrambling example for a subset. The original forward scrambling follows ① → ②, ② → ③, ③ → ①. The reverse mapping restores: ② → ①, ③ → ②, ① → ③. This reordering is based on the same chaos sequence used for scrambling.

4.3.2. Computational complexity and privacy-performance trade-off

To reduce computational overhead, *FedDC* adopts a per-segment scrambling strategy: each segment incurs a cost of $O(L \log L)$ due to sorting, and the total runtime scales linearly with the number of selected segments N_s. When needed, the same scrambling π_s can be reconstructed using (μ, x_0, L) together with the recorded start positions T_p, enabling consistent inverse scrambling on the client side.

Unlike conventional heavyweight cryptography, this method preserves the model update's dimensional structure and remains compatible with FedAvg. Rather than providing full cryptographic confidentiality, the goal is to reduce the usability of exposed parameters by disrupting their structural alignment. By scrambling selected parameter subsets, the process mitigates parameter-alignment attacks, cross-round trajectory analysis, and reconstruction-oriented attacks such as model inversion, while its tunable design allows a flexible privacy-performance trade-off through (N_s, L).

4.3.3. Application to CNN and transformer architectures

In CNNs, fully connected layers are denoted H_l. The same mechanism is applied to the selected convolutional and fully connected layers. After generating the corresponding chaotic sequence from the round-wise parameters, the selected parameter subsets are permuted according to the induced ranking and then uploaded as protected parameters.

For BERT, the scrambling target is the value projection matrix W_v in the self-attention layer. In our implementation, *FedDC* supports variable scrambling depth, allowing the number of protected Transformer layers to be adjusted to achieve the desired privacy-efficiency balance. This layer-aware design further supports *FedDC*'s selective protection objective and avoids the unnecessary overhead of uniformly protecting the entire model.

4.4. Inverse scrambling

Once the server has aggregated and broadcast a new global model, each client initiates the inverse scrambling phase to recover their local parameters. This process ensures that protected parameters, previously permuted using chaos-induced scrambling, can be accurately restored to their original order.

For layers denoted as V_c, inverse scrambling requires the protected parameters θ^* along with the chaos parameters used during scrambling: the logistic map parameter μ, the initial condition list T_x, the protected position list T_p, and the chaotic batch size C_{bs}. Clients reconstruct the chaos sequences using (μ, T_x) and apply the corresponding inverse scrambling to restore the original parameter ordering. This inversion strategy is shown in Fig. 5, where a sample sequence undergoes forward scrambling using values including 0.6188, 0.8689, and 0.4196, and then reverse mapping is applied to restore the original configuration.

For fully connected layers H_l in CNNs and the value projection matrix W_v in BERT, a similar inverse scrambling process is applied. Clients regenerate the chaotic sequences using μ and the corresponding round-wise seeds in T_x, reconstruct the original scrambling, and apply the inverse mappings to the received protected parameters.

To reduce computational cost, a binary search-based inverse mapping is employed, significantly lowering the overhead compared with naive sorting. For BERT's value projection matrix $W_v \in \mathbb{R}^{768\times768}$, both scrambling and inverse scrambling are performed per chaotic batch of 16 columns. Once all protected subsets are restored, the model parameters recover their original ordering and can be directly used for subsequent local training or inference. This client-side inverse scrambling design allows *FedDC* to remain compatible with standard server-side FedAvg while avoiding explicit plaintext recovery at the server. As a result, the server observes only protected parameters, whereas each client restores the original structural ordering locally after the model is broadcast.

4.5. Security analysis

To evaluate the security of the proposed mechanism, we analyze its behavior from both structural and system-level perspectives under the defined threat model, including both black-box and white-box adversarial scenarios. Rather than providing full cryptographic confidentiality, *FedDC* aims to reduce the usability of exposed model parameters by disrupting their structural alignment, thereby increasing the difficulty of inference and reconstruction attacks.

To formally characterize this protection objective, we consider an adversary $\mathcal{A}$ attempting to recover the original parameter structure from the observed scrambled parameters. Specifically, the scrambling process is defined as a transformation $W' = \pi(W)$, where W is the original parameter matrix and π denotes the permutation operation over its elements. The adversary is assumed to observe uploaded model parameters and aggregated updates, but cannot access the corresponding scrambling configuration, including the chaotic control parameter μ and client-side seeds $x_0^{(s)}$. The objective of $\mathcal{A}$ is to recover the original mapping π and reconstruct $W = \pi^{-1}(W')$. At the element level, this corresponds to $W'_{ij} = W_{\pi(i,j)}$, meaning that parameter values are preserved while their structural relationships are disrupted. This design focuses on breaking the correspondence between parameter positions and their semantic roles, rather than concealing values. Chaotic maps are derived from nonlinear dynamical systems, exhibit initial-value sensitivity and unpredictability, enabling the generation of pseudo-random sequences at low computational cost.

In contrast to standard random permutation strategies, which adopt independent client-side shuffling with no shared generation mechanism, the proposed chaotic perturbation introduces a keyed, synchronized transformation mechanism tailored for FL. However, random permutation lacks a unified deterministic structure across clients and the server, making consistent inverse mappings hard to implement for aggregation in federated optimization. Besides, a purely random permutation has no temporal or state-dependent evolution, so cross-round consistency cannot be maintained without extra coordination. Furthermore, the chaotic mechanism derives from nonlinear dynamical systems with high sensitivity to initial conditions, producing structured yet unpredictable transformation trajectories. This design effectively defends against inference-based reconstruction attacks under partial observation, while retaining invertibility via shared initialization parameters.

In each communication round, the chaotic control parameter μ is jointly generated by the server and a randomly selected client, preventing any single party from independently determining the scrambling configuration, as neither the server nor any individual client has full knowledge of both the control parameter and the local seeds. The segment-level seeds $x_0^{(s)}$ are generated on the client side and remain hidden from the server. Although consistent across clients within a round, these seeds are refreshed in each communication round, ensuring that scrambling patterns vary over time and reducing the risk of cross-round pattern inference. Under the honest-but-curious and non-colluding setting considered in this work, adversaries can observe protected model parameters but cannot access the corresponding scrambling parameters prior to DP perturbation. In this context, disrupting parameter align-

ment is sufficient to significantly reduce the effectiveness of parameter inference and reconstruction attacks, as these attacks typically rely on consistent structural correspondence between model parameters and their semantic roles.

Compared with standard symmetric encryption schemes (e.g., AES [43] or ChaCha20 [44]), and lightweight cryptographic primitives designed for resource-constrained environments (e.g., SIMON, SPECK [45]), chaotic scrambling does not aim to provide strong cryptographic guarantees but instead offers a lightweight structural protection mechanism that preserves compatibility with direct aggregation. While both categories of cryptographic methods provide formal encryption-based confidentiality, they are typically incompatible with direct parameter aggregation and often require additional mechanisms in FL settings. In contrast, in FL scenarios where efficiency and scalability are critical, our design provides a practical balance between privacy protection and system overhead. From a combinatorial perspective, the large permutation space increases the difficulty of recovering the original parameter alignment when the scrambling configuration is unavailable. However, this observation is considered as an additional protection factor rather than a formal cryptographic security guarantee. At the system level, *FedDC* further strengthens protection by keeping all scrambling parameters local to each client. As a result, even if an adversary observes protected parameters across multiple rounds, the absence of consistent scrambling patterns and the lack of access to chaotic parameters prevent reliable reconstruction of parameter alignment, as the scrambling patterns are independently regenerated in each round without shared global consistency that could be exploited across observations. Overall, the proposed mechanism provides a lightweight privacy protection strategy under the considered threat model, increasing the difficulty of structural parameter leakage while maintaining low computational overhead.

5. Evaluation

This section presents the experimental setup, including datasets, models, and hyperparameters, followed by the evaluation of *FedDC*. We also explain the rationale behind DP parameter selection, evaluate the robustness of *FedDC* under both black-box and white-box attacks, and analyze the scrambling overhead and its influencing factors.

5.1. Experiment setup

All experiments are conducted on a Windows server equipped with an Intel Core i7-9700 CPU, one Nvidia GeForce RTX 2080-Ti GPU, and four Nvidia GeForce RTX 3090-Ti GPUs.

Datasets. *FedDC* is evaluated on four datasets: MNIST, Fashion-MNIST, CIFAR-10, and the Amazon Review (AR) dataset. For fair assessment, the image datasets are partitioned into IID subsets across clients, while the AR dataset is split under a non-IID setting, simulating heterogeneous data distributions.

- MNIST: A handwritten digit recognition dataset comprising 60,000 training and 10,000 test samples. Each sample is a 28×28 grayscale image (784 pixels), with pixel values in the range [0, 255].
- Fashion-MNIST: Similar in format to MNIST, this dataset contains grayscale images of 10 clothing categories. It includes 60,000 training and 10,000 test samples, each of size 28×28.
- CIFAR-10: A dataset consisting of 60,000 color images ($32 \times 32 \times 3$) across 10 categories. Of these, 50,000 are used for training and 10,000 for testing.
- Amazon Review (AR): A text dataset containing 80,000 product reviews annotated with ratings from 1 to 5. The test set includes 8000 samples. To simulate non-IID conditions, each client is trained on a subset containing different rating distributions.

Models. We select models based on dataset complexity. For MNIST and Fashion-MNIST datasets, we adopt a standard CNN architecture, given their similar grayscale format and classification complexity. The CNN consists of three convolutional layers (each with 3×3 kernels), interleaved with two max-pooling layers, followed by two FC layers comprising 64 and 10 units, respectively. For the more complex CIFAR-10 dataset, which contains color images with richer visual features, we employ the ResNet-18 architecture. This model includes four groups of convolutional layers, each comprising two residual blocks, and is finalized by two FC layers. For the text-based AR dataset, we utilize a pre-trained BERT model as the backbone and append a dropout layer and a linear classification layer to handle the rating prediction task.

Table 2
Final test accuracy and stability (Std. = Standard Deviation) of *FedDC* across benchmark datasets.

Dataset	Rounds	ϵ	Accuracy (%)	Std. (%)
MNIST	100	10	95.71	0.04
		60	98.76	0.06
Fashion-MNIST	100	10	81.67	0.07
		60	85.44	0.02
CIFAR-10	100	10	70.58	0.11
		60	74.23	0.11
Amazon Review	10	10	86.97	0.02
		60	89.26	0.05

Hyperparameters. For the image datasets, we simulate an FL environment with 100 clients in total. In each communication round, 10% of clients are randomly selected to participate in training, in a standard partial-participation setting. Unless otherwise specified, all clients share the same model architecture and training configuration. The number of communication rounds, denoted as R, is set to 100. For MNIST and Fashion-MNIST, the number of local epochs is $E = 5$, the initial learning rate is $\eta = 0.01$, the weight decay is 0.005, and the batch size is $BS = 16$. For CIFAR-10, we use a smaller learning rate $\eta = 0.001$ and a larger batch size $BS = 32$ to stabilize training on the more complex dataset, while keeping other settings consistent. For the AR dataset, we use a smaller-scale configuration with 10 clients, $R = 10$, $E = 5$, learning rate $\eta = 10^{-5}$, and batch size $BS = 16$, following common practice in text-based FL tasks. We adopt the Laplace mechanism to perturb model updates. Specifically, before aggregation, model updates are clipped to a fixed norm bound, and Laplace noise is added to each parameter to mitigate privacy leakage. The privacy budget is set to $\epsilon = 60$, representing a moderate level of noise that balances privacy protection and model utility.

5.2. Results

5.2.1. Performance of FedDC

The performance of *FedDC* is evaluated across four benchmark datasets under DP budgets. As shown in Table 2, *FedDC* consistently achieves high accuracy while maintaining stable convergence. Under a relatively large privacy budget ($\epsilon = 60$), the model achieves around 98% accuracy on MNIST, above 85% on Fashion-MNIST, and over 74% on CIFAR-10. When a stricter privacy constraint ($\epsilon = 10$) is applied, the accuracy decreases moderately (around 95% on MNIST), but remains competitive across all datasets. To further quantify training stability, we report the standard deviation (Std.) of test accuracy over the final communication rounds. The consistently low Std. values indicate that *FedDC* converges smoothly and maintains robust performance under varying privacy budgets.

To ensure a fair and representative evaluation, we compare *FedDC* with several widely studied privacy-preserving FL methods covering different protection paradigms. Specifically, **PE-DPFL** [46] introduces a regularization term and local DP noise before client updates, while **ALI-DPFL** [47] adopts adaptive local iterations to improve convergence under resource constraints. We also include representative SA approaches, namely **SecAgg** [37] and **Flamingo** [23], which are widely used lightweight SA-based methods that protect model updates during trans-

Table 3
Average accuracy (%) over 5 runs on MNIST, Fashion-MNIST, and CIFAR-10 for different privacy-preserving FL methods.

Method	MNIST	Fashion-MNIST	CIFAR-10
standard FL	98.85	85.92	76.84
standard FL with DP	98.53	83.07	74.06
PE-DPFL (DP)	98.08	83.31	54.21
ALI-DPFL (DP)	98.82	84.31	55.67
SecAgg (SA)	98.54	84.90	75.44
Flamingo (SA)	98.85	85.67	75.73
BlindFL (HE)	98.70	–	77.00
ADPHE-FL (DP + HE)	98.96	83.95	71.04
FedDC (w/o DP)	*98.83*	*85.52*	*74.71*
FedDC	*98.76*	*85.44*	*74.23*

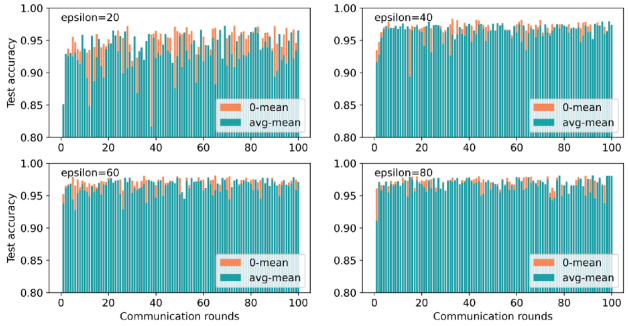


Fig. 6. The influence of privacy budget on model performance (MNIST dataset).

mission while incurring minimal overhead. In addition, **BlindFL** [16] employs HE to secure model updates during aggregation. **ADPHE-FL** [17] is a hybrid privacy-preserving FL framework that integrates adaptive DP with HE, employing a two-layer protection strategy to jointly ensure privacy. We consider an FL setting with 100 clients, where 10% are randomly selected in each communication round. Additionally, we include standard FL and standard FL with DP as reference baselines to better illustrate the impact of privacy mechanisms. All methods are evaluated under the same model architecture and training configuration.

As shown in Table 3, *FedDC* achieves a favorable balance between privacy protection and model utility across all datasets. Although it introduces a moderate drop in accuracy compared with standard FL (e.g., 74.23% vs. 76.84% on CIFAR-10), this trade-off is expected given the added protection mechanisms and remains competitive with existing privacy-preserving methods. Compared with DP-based methods, *FedDC* significantly improves performance, especially on more complex datasets. For example, on CIFAR-10, *FedDC* achieves 74.23%, substantially outperforming PE-DPFL (54.21%) and ALI-DPFL (55.67%), indicating that combining DP with complementary protection mechanisms effectively alleviates the utility degradation caused by noise injection. The relatively small performance gap between standard FL and standard FL with DP is due to the use of a moderate privacy budget, reflecting practical FL scenarios that prioritize utility. Compared with encryption-based approaches, including SA and HE methods, *FedDC* remains competitive while additionally incorporating DP-based perturbation. While Flamingo achieves slightly higher accuracy on CIFAR-10 (75.73%) and BlindFL reaches 77.00%, these methods rely on stronger cryptographic protection. In contrast, *FedDC* provides a lightweight alternative with lower overhead while maintaining comparable performance. Compared with the hybrid baseline ADPHE-FL, *FedDC* demonstrates better performance on CIFAR-10 (74.23% vs. 71.04%). We attribute this improvement to the selective, layer-wise protection strategy employed by FedDC, which preserves the utility of less sensitive layers, unlike the uniform protection often used in hybrid schemes. Finally, the minimal gap between *FedDC* and its variant without DP (e.g., 74.23% vs. 74.71% on CIFAR-10) indicates that the introduced DP perturbation causes only negligible utility loss, confirming that the proposed dual-layer mechanism effectively balances privacy and performance.

5.2.2. Impact of privacy budget on model performance

The privacy budget ϵ in DP controls the trade-off between privacy protection and model utility. A larger privacy budget (e.g., $\epsilon = 60$) provides weaker formal privacy guarantees but allows the model to maintain stable convergence and high accuracy. Such settings are commonly adopted in practical FL scenarios where stronger privacy guarantees are relaxed in favor of improved utility and system performance.

To provide a more comprehensive evaluation, we further investigate the behavior of *FedDC* under a wide range of privacy budgets, with $\epsilon \in \{5, 10, 20, 40, 60, 80\}$. As ϵ decreases, stronger noise injection leads to a gradual degradation in model accuracy, reflecting the expected privacy-utility trade-off. Notably, even under stringent privacy settings such as $\epsilon = 5$ and $\epsilon = 10$, *FedDC* consistently maintains over 90% accuracy on MNIST, highlighting its ability to preserve utility under strong privacy constraints. For larger privacy budgets ($\epsilon \geq 20$), we further analyze the training dynamics across communication rounds, as illustrated in Fig. 6. The results show that model performance improves with increasing ϵ due to reduced noise magnitude, while convergence remains stable across all settings. In particular, performance gains become marginal when $\epsilon \geq 60$, indicating diminishing returns under weaker privacy constraints.

Overall, these results demonstrate a clear and consistent privacy-utility trade-off, and confirm that *FedDC* maintains stable convergence and effective performance across a wide range of privacy budgets.

5.2.3. Protected model exposure evaluation

To evaluate the resilience of *FedDC* under the defined threat model, we simulate two practical adversarial scenarios. First, we conduct black-box inference attacks, assuming the attacker obtains access to the protected model but lacks the inverse scrambling key. Second, we explore white-box model inversion attacks under inverse scrambling access, where the adversary can inspect the inverse scrambling model parameters.

Black-box Attack. Under the defined threat model, we evaluate a black-box adversary scenario in which an adversary has obtained a copy of the protected global model, either by intercepting communication between clients and the server or by unauthorized access to the model stored on the server. Without knowledge of the inverse scrambling key, the attacker cannot interpret the internal parameters of the model. However, they can still attempt to use it directly as a prediction engine. This simulates a realistic threat where protected models are misused for membership inference or data leakage without inverse scrambling. The effectiveness of the scrambling scheme is demonstrated by varying the number and types of scrambled layers. Specifically, Table 4 presents the accuracy achieved by adversaries across different configurations, including encrypting only the convolutional layer, only the FC layers, or both. For instance, on MNIST and Fashion-MNIST, encrypting either the convolutional layer or the two FC layers leads to accuracy close to random guessing (9-12%), effectively neutralizing the stolen model's utility. This result confirms the security provided by the FedDC scheme even when only one type of layer is protected. However, in the CIFAR-10 setting, encrypting the convolutional layer alone yields a notably higher accuracy (50.95%), indicating partial model usability. This is attributed to the architectural complexity of ResNet18, whose deeper layers can still extract meaningful features despite the first layer being protected. When both the convolutional and FC layers are protected, the accuracy drops back to 10.14%, confirming that multi-layer scrambling is essential for defending against model misuse in deep architectures.

Interestingly, among the three datasets, encrypting only the FC layers results in even lower attack performance compared to encrypting both convolutional and FC layers. This phenomenon may be attributed to the shallow nature of the CNN architectures used for these datasets

Table 4
Black-box attack results under different encryption settings. Accuracy and F1-score are averaged over 10 attack runs and reported with standard deviation.

Dataset	Convolutional Layer	Fully Connected Layer	Accuracy (% ± std)	F1-score (% ± std)
	–	FC Layer 0&1	9.70 ± 0.14	9.80 ± 0.45
MNIST	Conv Layer 0	–	12.58 ± 0.23	12.40 ± 0.39
	Conv Layer 0	FC Layer 0&1	10.35 ± 0.18	10.20 ± 0.41
	–	FC Layer 0&1	9.98 ± 0.17	9.60 ± 0.42
Fashion-MNIST	Conv Layer 0	–	11.54 ± 0.20	11.50 ± 0.44
	Conv Layer 0	FC Layer 0&1	10.37 ± 0.16	10.10 ± 0.38
	–	FC Layer 0&1	10.03 ± 0.12	9.90 ± 0.40
CIFAR-10	Conv Layer 0	–	50.95 ± 0.29	49.90 ± 0.52
	Conv Layer 0	FC Layer 0&1	10.14 ± 0.14	10.30 ± 0.41
AR (1 layer protected)	–	–	13.99 ± 0.26	13.00 ± 0.48
AR (2 layers protected)	–	–	5.54 ± 0.19	5.10 ± 0.37
AR (3 layers protected)	–	–	3.34 ± 0.15	3.20 ± 0.29

(e.g., 1-2 conv layers followed by fully connected ones). In such cases, the majority of discriminative power lies in the FC layers, while the initial conv layer primarily serves for basic feature extraction. Therefore, encrypting only the FC layers may already destroy the majority of the model's utility for the adversary. In contrast, when both layers are protected, slight numerical noise or interaction between scrambling layers may still result in limited residual patterns being exploitable by the attacker. However, this gap is negligible and still within the range of random guessing, confirming the robustness of our method.

Furthermore, while encrypting only the FC layers may already achieve strong defense in black-box attacks, it potentially exposes the model to stronger white-box threats. For instance, if the adversary obtains access to the protected FC layer parameters, they may attempt to brute-force the scrambling key by exhaustive search, especially if the scrambling key space is limited. However, when both convolutional and FC layers are jointly protected, the adversary cannot easily isolate the FC layer's transformations, nor infer the specific scrambling subset size used by FedDC. This additional complexity significantly increases the brute-force cost and ambiguity, thereby providing an extra layer of security. Although such attacks fall beyond the black-box setting considered in this work, this highlights the proactive design of FedDC to mitigate potential future threats even in stronger threat models.

For the AR dataset, which uses a transformer-based architecture, we explore the effect of encrypting 1 to 3 layers. The results show a clear downward trend, with one protected layer, the model's accuracy and F1-score drop to 13.99% and 13.00%, respectively. With two layers, these drop to 5.54% and 5.10%, and with three layers, to 3.34% and 3.20%. The inclusion of standard deviations confirms that the performance degradation is consistent across repeated attack attempts, indicating the robustness of *FedDC*'s protection under varying conditions.

In summary, encrypting both convolutional and FC layers in *FedDC* provides strong protection against the adversarial attacks, rendering stolen models ineffective in practice. The low standard deviation across trials further demonstrates the robustness and consistency of the defense.

White-box Attack. Under the defined threat model, we evaluate a white-box adversary scenario to assess the resilience of *FedDC* against model inversion attacks. In this setting, the adversary obtains access to the protected model parameters and the model architecture, but does not possess the inverse scrambling key. This reflects a realistic scenario where model updates are intercepted during transmission or leaked from storage in protected form. Following the methodology from Melis et al. [38], we implement an input reconstruction attack, where the adversary generates random input and iteratively updates it via gradient descent to minimize the model's prediction error for a chosen class. Since the attacker operates on protected and perturbed parameters, the reconstruction process is significantly disrupted, making it difficult to recover meaningful input.

To quantify the success of the reconstruction attack, we adopt the Mean Absolute Error (MAE) between the reconstructed input $\tilde{x}$ and the original input x. While prior work often uses Mean Squared Error (MSE), MAE provides a more robust measure that is less sensitive to outliers [48]. The MAE is formally defined as:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|x_i - \tilde{x}_i|, \tag{8}$$

where n denotes the number of pixels per image. Lower MAE indicates a more successful inversion. Table 5 compares the MAE values under a standard FL framework (without chaotic scrambling or DP) and under *FedDC*, across three image datasets.

Table 5
Model inversion results under white-box access: MAE (lower is better). Compared over 10 runs.

Framework	MNIST	Fashion-MNIST	CIFAR-10
standard FL	0.03 ± 0.01	0.02 ± 0.01	0.62 ± 0.05
standard FL with DP	0.14 ± 0.03	0.36 ± 0.04	1.42 ± 0.10
FedDC (w/o DP)	1.92 ± 0.06	0.57 ± 0.02	9.14 ± 0.21
FedDC	2.39 ± 0.18	0.68 ± 0.04	9.93 ± 0.64

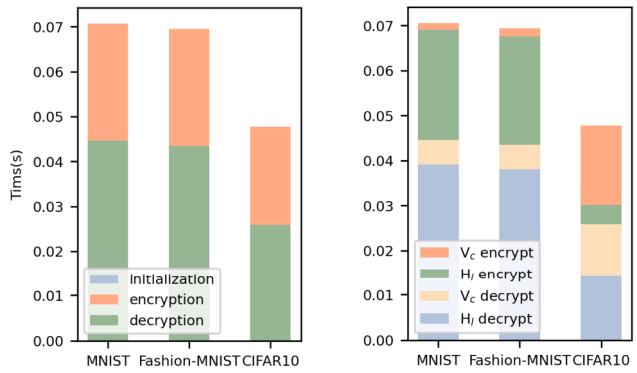
Table 5 presents inversion attack results under four settings: (1) standard FL, (2) standard FL with DP, (3) *FedDC* without DP, and (4) full *FedDC*. The results show a clear and consistent increase in MAE across all datasets as more protection mechanisms are introduced. For example, on MNIST, the MAE increases from 0.03 under standard FL to 1.92 with *FedDC* (without DP) and further to 2.39 with full *FedDC*, demonstrating substantial degradation in reconstruction quality. A similar trend is observed on CIFAR-10, where the MAE rises from 0.62 to 9.14 and 9.93, respectively. The larger MAE values on CIFAR-10 are expected due to its higher data complexity, which makes inversion inherently more challenging.

Comparing *FedDC* with its variant without DP reveals that chaotic scrambling alone already provides strong protection by disrupting the structural correspondence between parameters and their semantic roles, which is critical for inversion-based reconstruction. The addition of DP further increases uncertainty in parameter values, leading to additional degradation in attack effectiveness. This confirms that the proposed dual-layer design provides complementary protection beyond a single mechanism. The reported standard deviations across 10 runs remain small across all settings, indicating stable, consistent behavior.

Notably, we exclude AR text data from inversion analysis due to its discrete nature and poor attack convergence in preliminary trials. This suggests that inversion attacks are more applicable to vision tasks, while *FedDC* still provides meaningful robustness under white-box access.

5.2.4. Time overhead and efficiency

To evaluate the time efficiency of the proposed *FedDC* framework, we conducted a detailed analysis of the scrambling pipeline across different datasets and model complexities. Specifically, we decomposed the total processing time into four components: chaotic system initial-



(a) Overhead breakdown of *FedDC* scrambling pipeline.

(b) Component-wise scrambling and inverse scrambling time in FedDC.

Fig. 7. Computation overhead analysis of FedDC across different datasets.

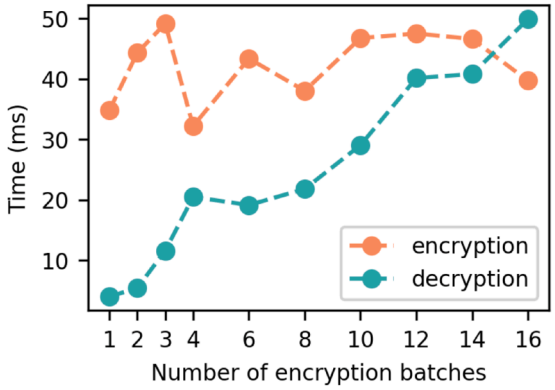


Fig. 8. Scrambling and inverse scrambling time with varying numbers of protected batches in layer V_c (MNIST).

ization, scrambling, inverse scrambling, and idle time (e.g., waiting for model aggregation or communication latency). The results are shown in Fig. 7. In all cases, the parameters of the initial convolutional layer and FC layers are selected for scrambling. The measured idle time is negligible compared with the overall processing time and thus omitted from the analysis.

For the MNIST dataset, initialization of the chaotic system consumed an average of 3 μs, scrambling required 26 ms, and inverse scrambling took 44.7 ms. Similarly, in Fashion-MNIST, the times are 2 μs, 26 ms, and 43.5 ms, respectively. In CIFAR-10, due to increased model complexity, the initialization cost rose slightly to 4 μs, while scrambling and inverse scrambling times are 21.9 ms and 25.9 ms. These results illustrate that even with protected layers, *FedDC* imposes a marginal temporal burden. Notably, we observed that scrambling and inverse scrambling together contributed less than 2.3% of the total round time in all CNN-based datasets.

This overhead is further influenced by layer-wise differences. For instance, in the convolutional layer $V_c^{(1)}$, the scrambling batch size is set to six, while in H_l (the FC layer), the entire parameter vector is protected. The discrepancy in parameter volume led to consistently shorter processing times for $V_c^{(1)}$ compared to H_l. Furthermore, inverse scrambling is invariably more time-consuming than scrambling due to the use of binary matching during inverse mapping, resulting in a logarithmic time complexity of $O(\log n)$. In CIFAR-10, the inverse scrambling time of H_l exceeded its scrambling time by approximately 85%, indicating this asymmetry. When deploying *FedDC* on complex architectures such as BERT, we evaluated the scrambling and inverse scrambling costs over the parameters of three self-attention layers on the AR dataset. The results showed scrambling and inverse scrambling times of 0.16 s and 0.79 s, respectively. The weight matrix dimensions in BERT ([768, 768]) led to 48 protected batches (each with 16 elements). Despite the larger parameter space than CNNs, the overall overhead remained acceptable, accounting for under 5% of the total training time per round, demonstrating the scalability of *FedDC* on transformer-based models.

To investigate the tunability of scrambling overhead, we measured the influence of the number of protected batches in V_c on runtime. As shown in Fig. 8, we varied the protected batch count from 0 to 16 on the MNIST dataset. Scrambling time remained nearly constant regardless of the number of batches. This is because our chaotic encoding algorithm is computationally lightweight, operating independently per batch with no cross-batch dependencies. Each batch undergoes a fixed-cost permutation based on a deterministic chaotic sequence, leading to a per-batch complexity of $\mathcal{O}(1)$, and thus an overall scrambling complexity of $\mathcal{O}(n)$ with a tiny constant. In practice, this results in a flat scrambling curve as the batch count increases. Furthermore, the inherently parallel nature of per-batch scrambling enables efficient utilization of hardware-level parallelism, such as multi-threaded CPU or GPU execution. This amortizes the computational cost across cores, resulting in a nearly constant scrambling time regardless of the number of protected batches. In contrast, inverse scrambling time increased approximately linearly with the number of protected batches. This is attributed to the chaotic index recovery mechanism, where each data point must locate its correct position within the shuffled index space. Assuming a binary search over a sorted key space, the per-index recovery cost is $\mathcal{O}(\log m)$, where m is the size of the index table for one batch. As the number of protected batches grows, the total number of lookups increases linearly, resulting in an overall inverse scrambling complexity of $\mathcal{O}(k \log m)$, where k is the number of protected data points. This leads to a roughly linear runtime trend with respect to the number of protected batches. Nevertheless, when all 16 batches are protected, the inverse scrambling latency remained below 60 ms, confirming that our approach is highly efficient and practical for real-world deployments.

In summary, the experimental results confirm that the time overhead of *FedDC* is both small and predictable. Across CNNs and BERT, scrambling and inverse scrambling operations account for only a small fraction of each round's training time. Moreover, the overhead scales roughly linearly with model complexity, ensuring the system's applicability to real-world scenarios. This validates *FedDC* as a low-overhead yet secure solution suitable for FL in both image and text domains.

6. Conclusion

This paper discusses the limitations of traditional scrambling methods in balancing security and model utility. To mitigate these issues, we propose *FedDC*, a novel privacy-preserving FL framework that integrates DP with a chaos-based scrambling mechanism. Leveraging the sensitivity to initial conditions and pseudo-randomness of chaotic systems, our method effectively permutes local model parameters before aggregation, thereby reducing the risk of parameter leakage. We evaluate *FedDC* on image and text datasets, demonstrating that *FedDC* achieves competitive accuracy while providing strong robustness against both black-box and white-box privacy attacks. Furthermore, we show that the added scrambling incurs minimal computational overhead, and we analyze its sources and implications in detail. These results indicate that *FedDC* offers a favorable trade-off between privacy, performance, and efficiency.

In summary, *FedDC* contributes a practical and effective solution for privacy-preserving FL, aligning with the growing demand for secure, efficient, and trustworthy intelligent systems. Future work may explore the application of *FedDC* in more adversarial settings, involving malicious clients, model poisoning, and evaluation under additional threat models.

CRediT authorship contribution statement

Yihan Liao: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization; **Jacky Keung:** Writing – review & editing, Supervision, Project administration; **Jingyu Zhang:** Writing – review & editing, Investigation, Conceptualization; **Yurou Dai:** Writing – review & editing, Investigation, Conceptualization.

Declaration of competing interest

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

Data availability

Data will be made available on request.

References

[1] Min B, Ross H, Sulem E, Veyseh A PB, Nguyen TH, Sainz O, et al. Recent advances in natural language processing via large pre-trained language models: a survey. ACM Comput Surv 2023;56(2):1–40.

[2] Zhao X, Wang L, Zhang Y, Han X, Deveci M, et al. A review of convolutional neural networks in computer vision. Artif Intell Rev 2024;57(4):99.

[3] Wu F, Zhang N, Jha S, McDaniel P, Xiao C. A new era in LLM security: exploring security concerns in real-world LLM-based systems. 2024, arXiv preprint arXiv:240218649.

[4] Bernett J, Blumenthal DB, Grimm DG, Haselbeck F, Joeres R, Kalinina OV, et al. Guiding questions to avoid data leakage in biological machine learning applications. Nat Methods 2024;21(8):1444–53.

[5] Rosenblatt M, Tejavibulya L, Jiang R, Noble S, Scheinost D. Data leakage inflates prediction performance in connectome-based machine learning models. Nat Commun 2024;15(1):1829.

[6] Voigt P, Von dem Bussche A. The EU general data protection regulation (GDPR). A practical guide, 1st ed, Cham: Springer International Publishing 2017;vol. 10(3152676):10–5555.

[7] Wen J, Zhang Z, Lan Y, Cui Z, Cai J, Zhang W. A survey on federated learning: challenges and applications. Int J Mach Learn Cybern 2023;14(2):513–35.

[8] Dong F, Ge X, Li Q, Zhang J, Shen D, Liu S, et al. PADP-FedMeta: a personalized and adaptive differentially private federated meta learning mechanism for AIoT. J Syst Archit 2023;134:102754.

[9] Liu W, Cheng J, Wang X, Lu X, Yin J. Hybrid differential privacy based federated learning for internet of things. J Syst Archit 2022;124:102418.

[10] Rodríguez-Barroso N, Jiménez-López D, Luzón MV, et al. Survey on federated learning threats: concepts, taxonomy on attacks and defences, experimental study and challenges. Inf Fusion 2023;90:148–73.

[11] Zhu J, Cao J, Saxena D, Jiang S, Ferradi H. Blockchain-empowered federated learning: challenges, solutions, and future directions. ACM Comput Surv 2023;55(11):1–31.

[12] Javeed D, Saeed MS, et al. Quantum-empowered federated learning and 6g wireless networks for iot security: concept, challenges and future directions. Future Gener Comput Syst 2024;160:577–97.

[13] Yang H, Ge M, Xue D, et al. Gradient leakage attacks in federated learning: research frontiers, taxonomy, and future directions. IEEE Netw 2023;38(2):247–54.

[14] El Ouadrhiri A, Abdelhadi A. Differential privacy for deep and federated learning: a survey. IEEE Access 2022;10:22359–80.

[15] Cui K, Feng X, Wang L, Ying Z. Secure aggregation with logarithmic overhead for federated learning in vanets. IEEE Trans Consum Electron 2025;71(1):2072–89.

[16] Gronberg E, d'Aliberti L, Saebo M, Hook A. BlindFL: segmented federated learning with fully homomorphic encryption. 2025, arXiv preprint arXiv:250111659.

[17] Wu T, Deng Y, Zhou Q, Chen X, Zhang M. ADPHE-FL: federated learning method based on adaptive differential privacy and homomorphic encryption. Peer-to-Peer Netw Appl 2025;18(3):141.

[18] Konečnỳ J, McMahan B, Ramage D. Federated optimization: distributed optimization beyond the datacenter. 2015, arXiv preprint arXiv:151103575.

[19] Kasyap H, Tripathy S. Beyond data poisoning in federated learning. Expert Syst Appl 2024;235:121192.

[20] Yang X, Huang W, Ye M. Dynamic personalized federated learning with adaptive differential privacy. Adv Neural Inf Process Syst 2023;36:72181–92.

[21] Lu S, Li R, Liu W, Guan C, Yang X. Top-k sparsification with secure aggregation for privacy-preserving federated learning. Comput Secur 2023;124:102993.

[22] Zhang C, Li S, Xia J, Wang W, et al. Batchcrypt: efficient homomorphic encryption for cross-silo federated learning. In: 2020 USENIX Annual technical conference (USENIX ATC 20). 2020, p. 493–506.

[23] Ma Y, Woods J, Angel S, Polychroniadou A, Rabin T. Flamingo: multi-round single-server secure aggregation with applications to private federated learning. In: 2023 IEEE Symposium on security and privacy (SP). IEEE; 2023, p. 477–96.

[24] Chen J, Li M, Cheng Y, Zheng H. Fedright: an effective model copyright protection for federated learning. Comput Secur 2023;135:103504.

[25] Zhang Y, Lu Y, Liu F. A systematic survey for differential privacy techniques in federated learning. J Inf Secur 2023;14(2):111–35.

[26] Xie Q, Jiang S, Jiang L, et al. Efficiency optimization techniques in privacy-preserving federated learning with homomorphic encryption: a brief survey. IEEE Internet Things J 2024;11(14):24569–80.

[27] Wei K, Li J, Ding M, Ma C, Yang HH, Farokhi F, et al. Federated learning with differential privacy: algorithms and performance analysis. IEEE Trans Inf Forensics Secur 2020;15:3454–69.

[28] Wei K, Li J, Ma C, Ding M, Chen W, et al. Personalized federated learning with differential privacy and convergence guarantee. IEEE Trans Inf Forensics Secur 2023;18:4488–503.

[29] Batool H, Anjum A, Khan A, Izzo S, et al. A secure and privacy preserved infrastructure for VANETs based on federated learning with local differential privacy. Inf Sci 2024;652:119717.

[30] Geyer RC, Klein T, Nabi M. Differentially private federated learning: a client level perspective. 2017, arXiv preprint arXiv:171207557.

[31] Choudhury O, Gkoulalas-Divanis A, Salonidis T, et al. Differential privacy-enabled federated learning for sensitive health data. 2019, arXiv preprint arXiv:191002578.

[32] Ling J, Zheng J, Chen J. Efficient federated learning privacy preservation method with heterogeneous differential privacy. Comput Secur 2024;139:103715.

[33] Kocarev L. Chaos-based cryptography: a brief overview. IEEE Circuits Syst Mag 2001;1(3):6–21.

[34] Arévalo I, Salmeron JL. A chaotic maps-based privacy-preserving distributed deep learning for incomplete and non-iid datasets. IEEE Trans Emerg Top Comput 2023;12(1):357–67.

[35] Li X, Huang K, Yang W, Wang S, Zhang Z. On the convergence of fedavg on non-iid data. 2019, arXiv preprint arXiv:190702189.

[36] Ponomareva N, Hazimeh H, Kurakin A, Xu Z, Denison C, et al. How to DP-FY ML: a practical guide to machine learning with differential privacy. J Artif Intell Res 2023;77:1113–201.

[37] Bonawitz K, Ivanov V, Kreuter B, Marcedone A, McMahan HB, et al. Practical secure aggregation for privacy-preserving machine learning. In: Proceedings of the 2017 ACM SIGSAC conference on computer and communications security. 2017, p. 1175–91.

[38] Melis L, Song C, De Cristofaro E, Shmatikov V. Exploiting unintended feature leakage in collaborative learning. In: 2019 IEEE Symposium on security and privacy (SP). IEEE; 2019, p. 691–706.

[39] Zhao Z, Luo M, Ding W. Deep leakage from model in federated learning. 2022, arXiv preprint arXiv:220604887.

[40] Yuan W, Wang X. FedAgg: adaptive federated learning with aggregated gradients. 2023, arXiv preprint arXiv:230315799.

[41] Li W, Xu Q, Dras M. Seeing the forest through the trees: data leakage from partial transformer gradients. 2024, arXiv preprint arXiv:240600999;.

[42] Wang F, Li B. Hear no evil: detecting gradient leakage by malicious servers in federated learning. 2025, arXiv preprint arXiv:250620651.

[43] Abdullah AM, et al. Advanced encryption standard (AES) algorithm to encrypt and decrypt data. Cryptogr Netw Secur 2017;16(1):11.

[44] Kebande VR. Extended-chacha20 stream cipher with enhanced quarter round function. IEEE Access 2023;11:114220–37.

[45] Beaulieu R, Shors D, Smith J, Treatman-Clark S, Weeks B, Wingers L. The SIMON and SPECK lightweight block ciphers. In: Proceedings of the 52nd annual design automation conference. 2015, p. 1–6.

[46] Shen X, Liu Y, Zhang Z. Performance-enhanced federated learning with differential privacy for internet of things. IEEE Internet Things J 2022;9(23):24079–94.

[47] Ling X, Fu J, Wang K, Liu H, Chen Z. ALI-DPFL: differentially private federated learning with adaptive local iterations. In: 2024 IEEE 25th international symposium on a world of wireless, mobile and multimedia networks (WoWMoM). IEEE; 2024, p. 349–58.

[48] Willmott CJ, Matsuura K. Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. Clim Res 2005;30(1):79–82.