SHE-LoRA: Selective Homomorphic Encryption for Federated Tuning with Heterogeneous LoRA

Jianmin Liu
School of Cyber Science and Engineering
Xi'an Jiaotong University
Shaanxi, China 710049
jianmin.liu@stu.xjtu.edu.cn

Li Yan
School of Cyber Science and Engineering
Xi'an Jiaotong University
Shaanxi, China 710049

Borui Li
School of Cyber Science and Engineering
Xi'an Jiaotong University
Shaanxi, China 710049

Lei Yu
Department of Computer Science
Rensselaer Polytechnic Institute
Troy, NY USA 12180

Chao Shen
School of Cyber Science and Engineering
Xi'an Jiaotong University
Shaanxi, China 710049

Abstract

Federated fine-tuning of large language models (LLMs) is critical for improving their performance in handling domain-specific tasks. However, prior work has shown that clients' private data can actually be recovered via gradient inversion attacks. Existing privacy preservation techniques against such attacks typically entail performance degradation and high costs, making them ill-suited for clients with heterogeneous data distributions and device capabilities. In this paper, we propose SHE-LoRA, which integrates selective homomorphic encryption (HE) and low-rank adaptation (LoRA) to enable efficient and privacy-preserving federated tuning of LLMs in cross-device environment. Heterogeneous clients adaptively select partial model parameters for homomorphic encryption based on parameter sensitivity assessment, with the encryption subset obtained via negotiation. To ensure accurate model aggregation, we design a column-aware secure aggregation method and customized reparameterization techniques to align the aggregation results with the heterogeneous device capabilities of clients. Extensive experiments demonstrate that SHE-LoRA maintains performance comparable to non-private baselines, achieves strong resistance to the state-of-the-art attacks, and significantly reduces communication overhead by 94.901% and encryption computation overhead by 99.829%, compared to baseline. Our code is accessible at https://anonymous.4open.science/r/SHE-LoRA-8D84.

1 Introduction

Large language models (LLMs) have demonstrated impressive generalization capabilities across a wide range of tasks, but their deployment in domain-specific applications (e.g., healthcare, financial transactions) [15, 25] often requires private, user-generated data. While the demand for deploying LLMs across diverse domains is continuously growing, stringent privacy preservation regulations like GDPR [44] pose significant barriers to centralized fine-tuning. To address this issue, Federated

Preprint.

learning (FL) has emerged as a promising and privacy-preserving solution by enabling decentralized parameter-efficient fine-tuning (PEFT) of LLMs without exposing sensitive user data [52].

Among various PEFT techniques, Low-Rank Adaptation (LoRA) stands out due to its high efficiency and model quality. It reparameterizes the weight matrix $W \in \mathbb{R}^{m\times n}$ as $W = W_0 + BA$, where $W_0 \in \mathbb{R}^{m\times n}$ represents the frozen pre-trained parameters, and $B \in \mathbb{R}^{m\times r}$ and $A \in \mathbb{R}^{r\times n}$ are the adapter parameters to be learned. Given that $r \ll \min(m, n)$, LoRA significantly reduces both computation and communication costs in Federated PEFT. However, recent works [36, 6] have shown that the parameters or gradients transmitted during Federated PEFT can be exploited via *inversion attacks* to reconstruct private training data, highlighting the need for enhanced privacy protection in federated PEFT with LoRA.

Many privacy-preserving techniques have been proposed to mitigate privacy leakage risks in FL, including differential privacy (DP) [42, 51, 56], secure multi-party computation (MPC) [34, 29, 53], and homomorphic encryption (HE) [20, 27, 23]. DP ensures formal privacy guarantees by perturbing data or model updates with noise. However, in LoRA-based settings which involve the multiplication between A and B, the noise is amplified, which can hinder convergence and degrade model performance [42]. In contrast, cryptographic approaches such as MPC and HE can achieve higher accuracy. MPC-based secure aggregation employs techniques such as garbled circuits and secret sharing to securely compute PEFT updates without exposing sensitive data. Nevertheless, it often requires intricately designed computation and synchronization protocols, making it less practical for FL with heterogeneous data and system capabilities [28, 31]. Selective HE (SHE) [20, 27, 23] offers a compelling alternative. By encrypting only sensitive parameters and enabling computation on encrypted data without decryption, it offers strong privacy guarantees and relatively low HE costs without sacrificing accuracy, making it highly attractive for privacy-preserving federated PEFT.

However, existing SHE methods struggle to balance privacy and efficiency in cross-device federated PEFT with LoRA, particularly under Non-IID (Non-Independently Identically Distributed) data and heterogeneous device capabilities. As discussed in Section 2.4, two observations highlight these challenges: 1) LoRA matrix multiplication makes ΔW denser, which may increase the number of parameters requiring encryption, and 2) variations in encrypted positions across heterogeneous clients lead to an expanded union of encrypted parameters during aggregation, raising HE costs. Driven by these limitations, we aim to adaptively balance security and HE overhead per client in cross-device federated PEFT with LoRA. Achieving this requires addressing the following key challenges:

• **How to adaptively apply SHE across heterogeneous clients?** Algorithms like FedAvg [32] are not directly applicable to LoRA-based PEFT. Naively applying LoRA requires all clients to use the same low-rank configuration, which is impractical for devices with heterogeneous resources. Furthermore, separately aggregating the two LoRA matrices (A and B) is not mathematically equivalent to full-weight aggregation (Section 2.4). This challenge is exacerbated by the fact that clients with different hardware capabilities cannot encrypt parameter updates of uniform size, which disrupts aggregation and complicates reconstruction of low-rank matrices from encrypted weights. Hence, a new SHE-compatible aggregation algorithm is needed to support heterogeneity.

• **How to avoid the expansion of encrypted subsets on SHE?** In heterogeneous settings, clients may independently encrypt arbitrary positions in their model parameter matrix, inflating ciphertext size during aggregation. Moreover, mixing plaintext and ciphertext matrices introduces structural disorder, making aggregation inefficient. Without efficient negotiation of encryption positions, both ciphertext size and aggregation overhead can increase substantially.

To address these challenges, we propose SHE-LoRA, which integrates SHE and LoRA to enable efficient and privacy-preserving federated tuning of LLMs in cross-device environments. Specifically,

- We propose SHE-LoRA to enable client-specific encryption based on local resource constraints and privacy needs. Each client evaluates the sensitivity of LoRA parameters and selects a subset to encrypt within a local encryption budget. This subset is encoded using order-preserving encryption (OPE) and sent to the server, which then negotiates a global encrypted subset to optimally balance privacy and HE overhead per client.
- We introduce a column-swapping parameter obfuscation method that clusters plaintext and ciphertext parameters separately, enabling efficient matrix operations on plaintext and allowing ciphertexts to be batch encrypted. Additionally, the obfuscation of parameter positions making attacks more difficult and reducing the risk of privacy breaches.

• Our column-aware secure aggregation scheme aligns encrypted columns across clients and aggregates them, followed by low-rank decomposition for efficient reparameterization to restore LoRA structure. This improves both aggregation accuracy and communication efficiency.

• Experimental results on OpenLLaMA-3B and BERT-Large demonstrate that SHE-LoRA provides strong resistance to the state-of-the-art attacks, reducing reconstruction scores to zero when the batch size is $B \geq 8$ while maintaining performance comparable to non-private baselines. It also achieves up to 94.901% reduction in communication cost and 99.829% reduction in encryption overhead, compared to baseline.

2 Preliminaries and Motivations

2.1 Definition of Parameter Sensitivity

Inspired by model pruning theory, which removes unimportant model parameters to reduce model size while maintaining performance, SHE identifies and selectively encrypts the most important model parameters (i.e., the most sensitive ones whose removal will cause significant degradation of model performance). For a given model with parameters $\mathbf{W}$, let $\mathcal{L}(\mathbf{W})$ denote the loss function. Given a subset of the model parameters $\mathbf{w} \in \mathbf{W}$, their sensitivity is denoted as $\Omega(\mathbf{w})$, and the model parameters with $\mathbf{w}$ zeroed out are denoted as $\mathbf{W}_{-\mathbf{w}}$. Thus, $\Omega(\mathbf{w})$ is defined as the increase in loss when $\mathbf{w}$ is zeroed out:

$$\Omega(\mathbf{w}) = |\mathcal{L}(\mathbf{W}) - \mathcal{L}(\mathbf{W}_{-\mathbf{w}})|. \tag{1}$$

From Eq. (1), a larger loss increase after removing $\mathbf{w}$ indicates higher sensitivity of $\mathbf{w}$. Thus, $\Omega(\mathbf{w})$ reflects not only importance, but also the potential privacy risk that may be leaked via observing $\mathbf{w}$. However, directly computing $\Omega(w)$ for all parameters is computationally infeasible. Taylor approximation-based estimation [21, 16] requires gradient computation, which also incurs significant overhead, especially in large language models (LLMs), where high dimensionality and outlier activations [40] further exacerbate the cost. To mitigate this, we adopt Wanda [41] to estimate the sensitivity of a parameter W_{ij} in the i-th row and j-th column as:

$$\Omega(W_{ij}) = |W_{ij}| \cdot \|x_j\|_2, \tag{2}$$

where $|\cdot|$ represents the absolute value operator, $\|x_j\|_2$ evaluates the l_2 norm of the j-th features. Since both the model parameters $\mathbf{W}$ and the input data x are directly accessible, Wanda estimates parameter sensitivity with only a single forward pass.

2.2 Privacy Leakage Quantification

We leverage mutual information to quantify privacy leakage caused by the selective encryption of $\mathbf{w}$. Specifically, we assume that once encrypted with HE, $\mathbf{w}$ does not leak any privacy information. While for the accessible plaintext portion of the model parameters (i.e., $\mathbf{W}_{-\mathbf{w}}$), we measure the mutual information shared between $\mathbf{W}_{-\mathbf{w}}$ and $\mathbf{W}$ as:

$$I(\mathbf{W}; \mathbf{W}_{-\mathbf{w}}) = \sum_{y \in \mathbf{W}_{-\mathbf{w}}} \sum_{x \in \mathbf{W}} p(x, y) \log_2 \frac{p(x, y)}{p(x)p(y)}, \tag{3}$$

where $p(x, y)$ denotes the joint probability distribution, with $p(x)$ and $p(y)$ representing the marginal distributions of $\mathbf{W}$ and $\mathbf{W}_{-\mathbf{w}}$, respectively. Given our objective of selectively encrypting the most privacy-sensitive model parameters, $I(\mathbf{W}; \mathbf{W}_{-\mathbf{w}})$ quantifies the extent of privacy leakage attributable to the unencrypted model parameters (i.e., $\mathbf{W}_{-\mathbf{w}}$). As the value of $I(\mathbf{W}; \mathbf{W}_{-\mathbf{w}})$ increases, so does the risk of privacy leakage resulting from the selective encryption of $\mathbf{w}$.

2.3 Threat Model

Following prior work [27, 20], we assume a semi-honest adversary $\mathcal{A}$ that may compromise the aggregation server or a subset of clients. While $\mathcal{A}$ follows the training protocol, it passively attempts to infer private client data from observed PEFT updates. Even when $\mathcal{A}$ compromises certain clients, it

only gains access to the updates from those clients. We adopt single-key HE, where all clients use the same HE key to enable secure ciphertext aggregation and prevent the server from learning individual updates. For potential attacks from compromised clients, we assume secure communication channels between each client and the server. To maintain the integrity of privacy guarantees under single-key HE, we assume that the server and clients do not collude. In particular, compromised clients do not reveal their encryption keys to the server. Relaxing this assumption would require multi-key HE, which we discuss as a possible extension in Appendix C. Protection against other malicious behaviors (e.g., collusion, poisoning, backdoor attacks) is beyond the scope of this work; we refer readers to existing defenses [54, 37].

2.4 Motivations

Naive averaging of LoRAs leads to mathematical errors. Popular federated LoRA methods [52, 49, 33, 4], such as FedAvg with Naive LoRA, require clients to possess homogeneous LoRA ranks, and aggregates the two low-rank matrices A and B separately across clients (i.e., server side $= \sum B \times \sum A, \text{rank}(A) = \text{rank}(B)$). However, this introduces inconsistency in global model updates, as the aggregation of LoRA updates that can be used for model tuning (i.e., $\sum(B \times A)$) is intrinsically unequal to $\sum B \times \sum A$. This mismatch will cause deviations in reconstructed weight updates and degrade model performance. Moreover, separately aggregating the LoRA matrices requires all clients to use the same rank, which is unrealistic for cross-device federated LoRA. As a result, this naive approach is not applicable to heterogeneous LoRA settings.

Matrix multiplication expands encryption positions. HE is a privacy-preserving technique that deprives attackers of access to model parameters or gradients, preventing them from obtaining any information and thus hindering attacks. However, this perfect security comes with a high cost. Although existing SHE methods like FedML-HE [27] and MaskCrypt [23] reduce HE overhead by selectively encrypting a subset of model parameters with masking, the matrix multiplication of LoRA will lead to an expanded encryption subset as shown in Fig. 1, which significantly increases the cost of SHE during LoRA.

Matrices from heterogeneous clients inflate ciphertext size. Our experiments show that clients with heterogeneous hardwares and Non-IID data tend to focus on different sensitive model parameters during fine-tuning. As a result, the positions selected for encryption vary across clients. In the aggregation phase, if a client encrypts a specific model parameter, the parameter corresponding to the same position must remain encrypted in the global model as well, leading to inflated ciphertext size as the number of clients grows, which is illustrated in Fig. 2.

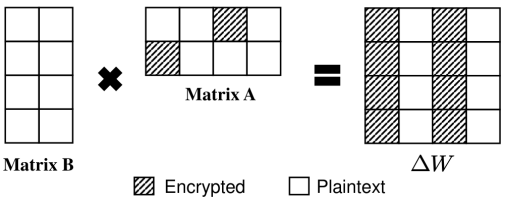


Figure 1: Expansion of encryption positions.

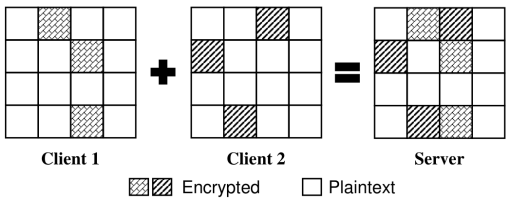


Figure 2: Inflation of ciphertext size.

3 Method

In order to adaptively balance security and HE overhead per client in cross-device federated LoRA, encrypting A offers a cost-effective solution. Since A directly operates on user data, it is more vulnerable to inversion attacks [36], making its protection essential for preventing privacy leakage. Following this rationale, the diagram of SHE-LoRA is as illustrated in Fig. 3.

Specifically, SHE-LoRA consists of the following components: **Step 1. HE Subset Negotiation**. First, based on the definition of model parameter sensitivity in Eq. (2), each client assesses its model parameter importance and transmits to server. Then, the server negotiates a global encryption subset and feeds it back to clients. **Step 2. Selective Encryption of Model Parameter Matrix**. Based on the global subset, clients perform column swapping to cluster encrypted columns on one side of the matrix, achieving parameter location obfuscation that enhances privacy protection and improves the efficiency of HE. **Step 3. Adaptive Aggregation**. The server performs adaptive, column-aware

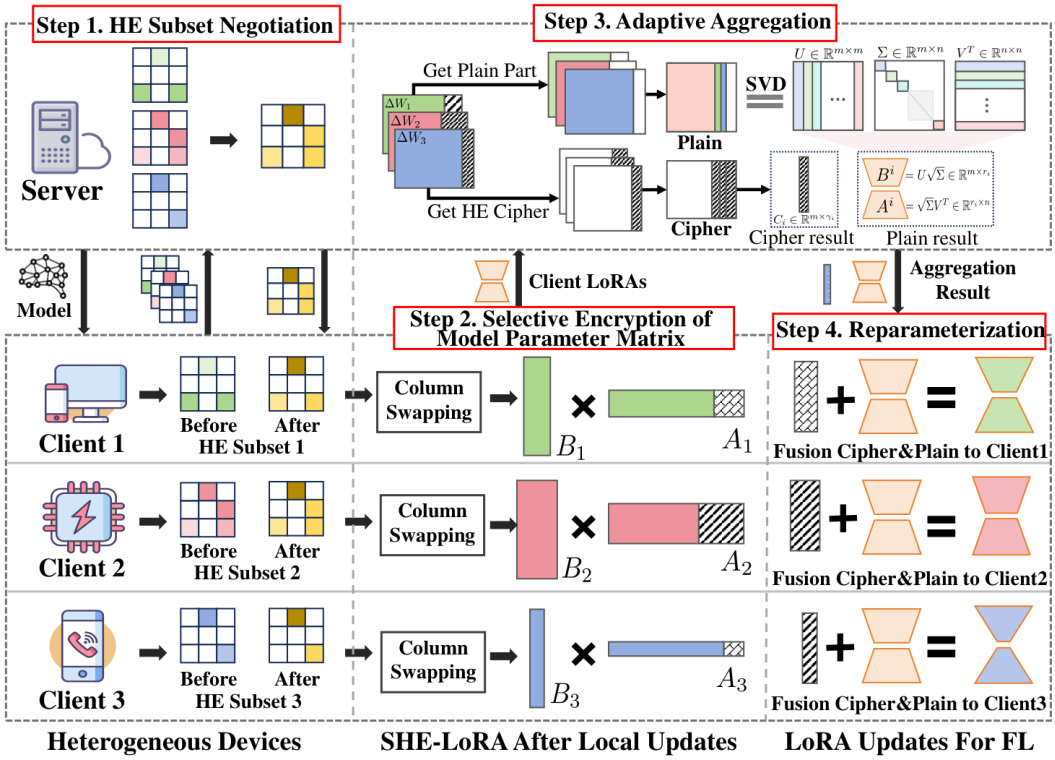


Figure 3: The diagram of the SHE-LoRA.

aggregation to the plaintext and ciphertext, results from heterogeneous encryption subsets, enabling efficient and accurate aggregation of model updates. **Step 4. Reparameterization**. Each client reparameterizes the aggregated plaintext and ciphertext results into LoRA parameters, matching its local rank and enabling continued local fine-tuning.

3.1 HE Subset Negotiation

3.1.1 Assessment of Parameter Importance

Fig. 4 illustrates the sensitivity of model parameters measured by Eq. (2), where darker color indicates higher importance. We can see that the sensitivity values generally differ by the columns of the parameter matrix, suggesting a strong correlation with specific data channels. Considering that as explained in Section 2.4, encrypting even a single element in a column will result in the expansion of encryption positions for that entire column due to matrix multiplication, and vectorized HE operation by columns is beneficial to improving HE efficiency [20, 12], we let each client assess the importance and determine its encryption subsets of columns by its encryption budget, which is pre-determined offline according to its device capability. Specifically, for a parameter matrix $A \in \mathbb{R}^{r\times n}$ and input $X \in \mathbb{R}^{L\times n}$, we use $S_j = \sum_{i=0}^{r} |W_{ij}| \cdot \|X_j\|_2$ to calculate the importance of the j-th channel, where $\|X_j\|_2$ evaluates the l_2 norm of the j-th features aggregated across L tokens. Thus, the proposed approach can not only provide channel-level privacy protection, but also prevent unnecessary expansion of encryption positions, and improve HE efficiency.

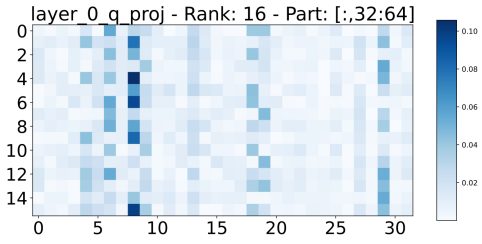


Figure 4: Sensitivity of model parameters.

3.1.2 Encryption Subset Negotiation

As explained in Section 2.4, clients with heterogeneous data distributions and device capabilities will select different positions and amounts of important model parameters for encryption according to their individual budgets. As the union of encryption subsets expands with the aggregation of client model updates, the size of the ciphertext may inflate significantly. To address this, we further narrow down the encryption subset by designing a globally shared encryption subset across clients, ensuring that the overall ciphertext size remains within the allocated encryption budget per client.

To prevent the server from snooping on the positions of important model parameters and conducting targeted attacks, we apply OPE to hide each client's encryption subset. As OPE preserves numerical ordering of the plaintext, the server cannot obtain any information from the cipher other than the plaintext order. Specifically, the client first encrypts a tuple (G, S) with OPE and sends it to the server, where G is the set of columns that needs HE, and S is their sensitivities. Then, the server maintains two shared lists based on all the clients' tuples: the *Common* list, which counts the occurrences of each column, and the *Sensitivity* list, which sorts all columns according to their importance. Finally, by taking into account both the overlap of important encryption subsets across clients (reflected in *Common* and *Sensitivity*) and the preferred encryption subsets of individual clients (reflected in G_i), the server negotiates a global encryption subset satisfying the budget of each client (denoted as $\gamma_i \in [0, 1]$), of which algorithmic details are elaborated in Appendix D.1. As a result, the negotiated global encryption subset optimally balances privacy and HE overhead per client.

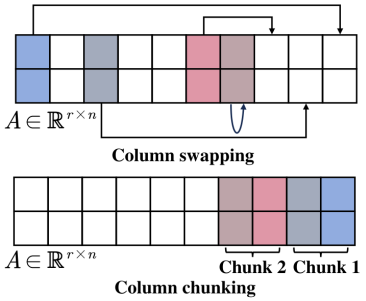


Figure 5: Selective encryption of columns.

3.2 Selective Encryption of Model Parameter Matrix

As illustrated in the top of Fig. 5, the selected columns for encryption may be scattered across the parameter matrix. This irregular distribution increases the complexity of matrix batching and raises the overhead of encryption, decryption, and computation. To address this issue, based on the negotiated encryption subset, we propose a column-swapping method to separately cluster the columns waiting to be encrypted and those remain unencrypted. This approach brings three key benefits: (1) encrypted columns are grouped together, allowing for efficient batch encryption, reduced storage and communication overhead; (2) the clustered unencrypted columns can be directly used in matrix operations, improving computational efficiency; and (3) the column-wise obfuscation increases the difficulty of potential privacy attacks.

After swapping, we selectively encrypt the model parameter matrix, with the encryption subset being the last k columns. Thus, the tensor to be encrypted, denoted as D, has the shape of (r, k). In HE, the CKKS (Cheon-Kim-Kim-Song) scheme [12] serves as an encryption technique that facilitates approximate floating-point arithmetic, making it particularly advantageous for the encryption of matrices or tensors. For large-scale tensors, due to the restricted cipher space for individual encryption, it requires the tensor to be split into blocks, with each block being encrypted separately. As illustrated in the bottom of Fig. 5, we divide D into blocks by column, assuming a block size of *chunk*, then the number of blocks is $n = \lceil k/chunk \rceil$, $D = \{D_1, D_2, \ldots, D_n\}$, where each block contains a tensor that has the shape of $(r, chunk)$. For each block D_i, CKKS encryption is performed to obtain cipher $C_i = \text{CKKS}(D_i, pk)$, where pk is the public HE key. After all blocks are encrypted, the complete cipher list $\{C_i\}^n$ is sent to the server.

3.3 Adaptive Aggregation

After receiving both unencrypted and encrypted model parameters from the clients, the server performs matrix multiplication to obtain each client's full model parameter matrix $\Delta W_i = B_i A_i$, where the last $k_i = \lfloor n \times \gamma_i \rfloor$ columns need encryption, while the rest remain unencrypted. Since decryption can only be performed on the client side, to prevent the inflation of ciphertext size caused by aggregating heterogeneous encryption subsets, we propose to aggregate the unencrypted and encrypted parts separately.

Aggregation of Unencrypted Model Parameters. By calculating the matrix multiplication of B and the unencrypted part of A (denoted as A^i_{plain}), we can obtain the aggregation result of unencrypted model parameters $\Delta W^i_{\text{plain}} = B^i A^i_{\text{plain}}$ for client i. As the number of encrypted columns of client i is k_i, then $\Delta W^i_{\text{plain}} \in \mathbb{R}^{m\times(n-k_i)}$. In our scenario, the encryption budgets of the clients are different, meaning that the shape of $\Delta W^i_{\text{plain}}$ varies across the clients. According to Section 3.2, we can ensure that the unencrypted model parameter positions of all the clients are left-aligned. Therefore, we can simply perform block-wise weighted averaging to aggregate the unencrypted model parameters. Specifically, we only average the unencrypted model parameter positions shared by all the clients as illustrated in Fig. 6. A detailed algorithmic introduction is available in Appendix D.2.

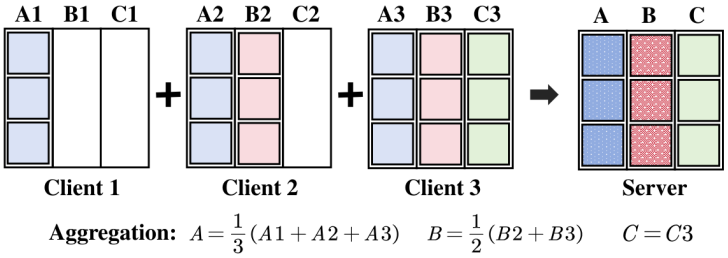


Figure 6: Aggregation of unencrypted model parameters (Same-colored columns are aligned; blank columns indicate the encrypted columns).

Aggregation of Encrypted Model Parameters: The server receives the set of ciphers $\Delta W^i_{\text{cipher}} = \{C_i\}^n \in \mathbb{R}^{m\times k_i}$ from the clients. Since the budget γ_i of each client is different, the generated cipher set may be inconsistent, but the cipher block order is consistent across all the clients. Similar to the above, the encrypted model parameter positions of all the clients are right-aligned, and the cipher sets are aggregated via block-wise weighted averaging. The detailed algorithm is in Appendix D.3.

To reduce communication overhead, we transmit the unencrypted and encrypted parts separately. For the unencrypted part, since most of the model parameters remain unencrypted during LoRA, we let the server, which generally has more computation capability, apply singular value decomposition (SVD) to decompose the aggregation result: $\Delta\overline{W}_{\text{plain}}(\in \mathbb{R}^{m\times K}) = U_p\Sigma_p V_p^\top$, where $K = n - \min(k_i)$,$U_p \in \mathbb{R}^{m\times m}, \Sigma_p \in \mathbb{R}^{m\times K}, V_p^\top \in \mathbb{R}^{K\times K}$. Then, according to each client's rank r_i, the server slices the decomposition results as: $U_p[:, : r_i] \in \mathbb{R}^{m\times r_i}, \Sigma_p[: r_i, : r_i] \in \mathbb{R}^{r_i\times r_i}, V_p^\top[: r_i, :] \in \mathbb{R}^{r_i\times K}$. Thus, each client receives the model parameters corresponding to its own rank. For the encrypted part, since each client only selectively encrypts a small portion of model parameters, we let the server first truncate the aggregation result by the encryption budget of respective clients, and then let the clients decrypt their corresponding encrypted model parameters as in Section 3.4. For example, as illustrated in the "Step 4" of Fig. 3, suppose Client 1's encryption budget can only afford 3 cipher blocks, while the aggregation result contains 6 cipher blocks in total. Therefore, the server will truncate the aggregation result to 3 blocks, and send it back to Client 1 for further decryption.

3.4 Reparameterization

To enable the next round of FL training, each client needs to merge the plaintext and ciphertext results returned by the server into new LoRA parameters. The client receives the plaintext decomposition results $B_p = U_p\sqrt{\Sigma_p} \in \mathbb{R}^{m\times r}$, $A_p = \sqrt{\Sigma_p}V_p^\top \in \mathbb{R}^{r\times K}$, and the cipher blocks from the server. For the plaintext decomposition results, the client directly zero-pads $B_p \in \mathbb{R}^{m\times r}$ and $A_p \in \mathbb{R}^{r\times n}$ for further update of its model parameter matrix. For the cipher blocks, the client first decrypts the ciphertext into plaintext, and then performs SVD and zero-padding to obtain the decomposed results $\Delta\overline{W}_{\text{cipher}} = U_c\Sigma_c V_c^\top$. By merging the decomposition results of the plaintext and ciphertext through Eq. (4) and restoring the position of the model parameters according to the corresponding positions in Section 3.2, the parameters can be restored to the correct aggregation result.

$$B_g = [U_p\sqrt{\Sigma_p} \quad U_c\sqrt{\Sigma_c}] \in \mathbb{R}^{m\times(r+r)}, \quad A_g = \begin{bmatrix}\sqrt{\Sigma_p}V_P^\top \\ \sqrt{\Sigma_c}V_c^\top\end{bmatrix} \in \mathbb{R}^{(r+r)\times n}. \tag{4}$$

Finally, the client performs SVD on A_g and B_g again, and re-adjust the parameter shapes according to the client's rank to $\hat{B} \in \mathbb{R}^{m\times r}$ and $\hat{A} \in \mathbb{R}^{r\times n}$. As the SVD conducted by the client is all on low-rank

matrices, the computation overhead is trivial. The detailed derivation of the reparameterization is presented in Appendix D.4.

4 Experiments

4.1 Settings

Model: In our experiments, we selected two representative pre-trained language models—namely, Bert-Large [14] and OpenLLaMA-3B [17] to investigate their performance across diverse task scenarios. Bert-Large is a model based on the Transformer encoder architecture, while OpenLLaMA-3B is founded on the LLaMA architecture.

Table 1: Heterogeneous device types.

Type	Rank	Budget(Bert,LLaMA)%
1	8	(0.4, 0.125)
2	16	(0.4, 0.125)
3	16	(0.8, 0.25)
4	32	(1.6, 0.50)

Datasets: We employed the IMDB and natural-instructions datasets for natural language understanding and natural language generation tasks, respectively. Natural-instructions [47] comprises 1616 NLP tasks spanning 76 distinct task types. Furthermore, we evaluated the performance of federated fine-tuned models on MMLU [22] for natural language generation tasks, and on GLUE [45] for natural language understanding tasks.

Hyperparameters of HE: We employed the CKKS from the TenSEAL library [7] to implement HE operations. We follow the instruction[1], by setting the polynomial degree to 8192,2048, and the modules chain to [60, 40, 60],[20,20] for OpenLLaMA-3B and Bert-Large, separately.

Devices: We established training on 50 clients for 200 rounds. SHE-LoRA is implemented in PyTorch based on the Flower framework[2], and conducted all experiments on a server equipped with four NVIDIA A6000 GPUs and an Intel Xeon Platinum 8369B CPU. In practical deployments, each client can be run on significantly less demanding hardware. As detailed in Table 1, we predefined four types of configurations with varying encryption budgets and LoRA ranks, and assigned them to clients based on their computational capabilities. From a naive perspective, we posit that weaker devices are characterized by lower rank and encryption budget. Similarly, high-performance devices are capable of supporting a higher rank and accommodating a more substantial encryption budget.

4.2 Privacy Attack

The SHE-LoRA is applicable to both parameter and gradient updates. Although parameter-based attacks can be transformed into gradient-based ones, this is highly inaccurate and impractical for large models, even with direct access to gradients [46]. Therefore, we evaluate SHE-LoRA under DAGER [36], the current state-of-the-art (SOTA) gradient inversion method. DAGER exploits the fact that gradients are linear combinations of input embeddings (Appendix A.5) . It iterates over the entire vocabulary and measures the distance between each embedding vector and the principal components of the gradient, aiming to identify tokens. This approach relies on a strong assumption that the number of unique tokens $b \ll rank$. However, this assumption rarely holds for resource-constrained clients with very low ranks, making such attacks significantly more difficult.

Table 2: Data reconstruction of DAGER on the OpenLLaMA-3B when Rank is 256.

Dataset	Method	B=4		B=8		B=16	
		R-1	R-2	R-1	R-2	R-1	R-2
SST2	Non-Privacy	90.343	89.197	80.976	77.834	42.281	31.79
	with DP	84.123	81.806	59.201	50.809	28.844	11.134
	SHE-LoRA	66.383	61.071	0	0	0	0
Rotten Tomatoes	Non-Privacy	59.14	55.316	19.297	10.606	1.366	0.563
	with DP	34.605	26.784	14.166	5.482	0	0
	SHE-LoRA	24.861	16.238	0	0	0	0

[1] https://github.com/OpenMined/TenSEAL

[2] https://flower.ai/

Theoretically, the change in the principal components of the gradient due to selective HE and column confusion is minimal, which does not lead to the failure of the DAGER attack. However, for the low-rank space of the LoRA parameter, this perturbation leads to a strong change in the orthogonal complement of the gradient, which is manifested by a drastic change in the vectors of the orthogonal complement in the column space of the gradient, leading to the failure of DAGER's span check. We show the defense effect of SHE-LoRA for different batch sizes at rank = 256, with only one column interfered. We did not report results for higher encryption ratios, as the attack failed when the ratio exceeded 0.125%.

The Table 2 demonstrates the data reconstruction effectiveness of the DAGER attack on two datasets, SST2 and Rotten Tomatoes, under the OpenLLaMA-3B, Rank=256 (smaller values indicate worse reconstruction and better privacy protection), comparing three methods: non-Privacy, with DP, and SHE-LoRA. Following the DP method mentioned by DAGER [36], we implement privacy protection by adding σ^2 noise to the gradient, and we choose $\sigma = 10^{-7}$. The SHE-LoRA shows strong defense in both datasets, with different batchsizes, especially at B=8 and above, the attacker cannot reconstruct any valid data (with a score of 0). Differential privacy can reduce the reconstruction effect, but the protection ability is far less than SHE-LoRA, and it will bring performance loss. SHE-LoRA is effective in improving the model security and resisting data reconstruction attacks, especially in the case of larger batch size, it can realize "complete protection", which is far better than the traditional differential privacy method.

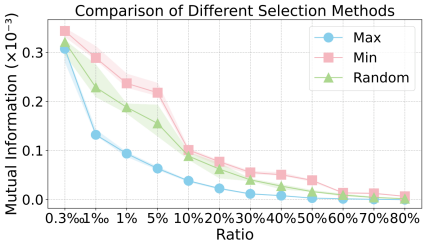


Figure 7: Impact of encryption ratio and strategy on mutual information.

4.3 Privacy Analysis from Mutual Information Perspective

In addition, as shown in Fig. 7, from an information theoretic perspective, we compare the variation of the mutual information of the SHE-LoRA leakage parameters under three different parameter selection strategies. As the parameter removal ratio increases, the mutual information values corresponding to all three strategies (Max, Min, and Random) gradually decrease, indicating that the more parameters are removed, the less information is contained in the remaining parameters, and the better privacy protection is. Among the three methods, the Max strategy has the lowest mutual information value and the strongest privacy protection at all ratios, while the Min strategy has the highest mutual information value and retains the most information, indicating a weaker degree of privacy protection. This indicates that more sensitive parameters contain more privacy information, and selective encryption for more sensitive parameters can more effectively reduce the risk of privacy leakage.

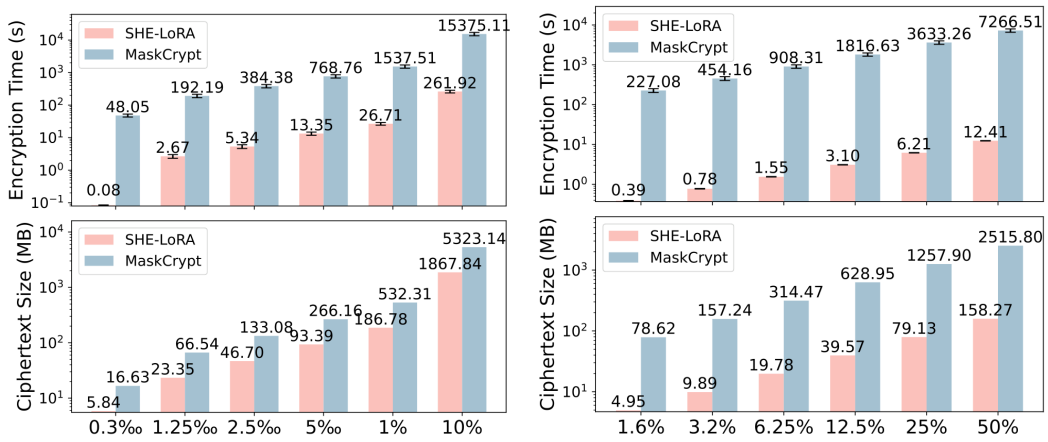
4.4 Performance

We report the performance comparison of SHE-LoRA with two homogeneous LoRA-based methods (FedIT [52] and FedSA [19]) and two heterogeneous LoRA-based methods (HeterLoRA [13] and FlexLoRA [5]). Natural language understanding tasks are evaluated on the GLUE benchmark [45], while natural language generation tasks are assessed on the MMLU benchmark [22]. Our method achieves comparable performance to the state-of-the-art method FlexLoRA and outperforms the other baselines. Detailed experimental results and analysis are in Appendix F.5.

4.5 Costs Comparison

To the best of our knowledge, we are the first to use HE specifically designed for federated LoRA. To show the overhead comparison of SHE-LoRA over other HE methods, we applied Paillier [35] for FedIT to implement MaskCrypt [23], who uses the paillier algorithm for FL, keeping the same encryption ratio as SHE-LoRA. The Fig. 8 shows the time of encryption and communication cost of MaskCrypt and SHE-LoRA under both OpenLLaMA-3B and Bert-Large models at fixed 5 rounds according to different encryption ratios.

Encryption Time: As the encryption ratio increases, the encryption time of both schemes increases significantly. On the OpenLLaMA-3B model, the encryption time of MaskCrypt is much higher than that of SHE-LoRA, especially at high encryption ratios (e.g., 10%), the time is as high as



(a) OpenLLaMA-3B. (b) Bert-Large.

Figure 8: The costs comparison of SHE-LoRA with MaskCrypt at different ratios.

15,375.11 seconds, while that of SHE-LoRA is only 261.92 seconds, which reduces the encryption overhead by 98.296%. At low encryption ratios (e.g., 0.3‰, 1.25‰), MaskCrypt's encryption time is also consistently higher than that of SHE-LoRA. And for the Bert model, at 50% encryption ratio, SHE-LoRA brings as much as 99.829% time gain, which suggests that MaskCrypt is extremely inefficient in large-scale parameter encryption scenarios, while SHE-LoRA greatly reduces the time overhead from encryption thanks to the batch packing encryption.

Ciphertext Size: Similarly, the size of both ciphertexts increases rapidly with the encryption ratio enhancement. The MaskCrypt ciphertext expansion is much larger than that of SHE-LoRA. For example, at 10% encryption ratio, the size of the MaskCrypt ciphertext is 5323.14MB, while that of the SHE-LoRA is 1867.84MB, which saves 64.911% of the communication overhead. And for Bert model, at 10% encryption ratio, this gain is even more obvious, reaching 94.901%.

During the federated LoRA of the large model, our SHE-LoRA scheme significantly outperforms the MaskCrypt due to the large number of parameters, especially at high encryption ratios, where the Paillier-based HE scheme MaskCrypt is almost unavailable. Furthermore, Appendix F.2 includes a detailed report on the training time of SHE-LoRA across varying encryption budgets.

5 Conclusion

In this paper, we introduce SHE-LoRA, a heterogeneous federated fine-tuning approach leveraging selective homomorphic encryption, to address privacy challenges in heterogeneous environments. This method constrains the expansion of the ciphertext through encrypted subset negotiation. Column swapping enhances the computational efficiency of homomorphic encryption while obfuscating parameter positions, thereby heightening attack resistance. The column-aware aggregation technique ensures result accuracy. Ultimately, the reparameterization approach tailors aggregated outcomes to heterogeneous devices, addressing device heterogeneity. SHE-LoRA offers customized privacy protection tailored to device heterogeneity. Moving forward, we aim to investigate the theoretical relationships among parameter sensitivity, defense robustness, and information leakage. Additionally, we plan to integrate this approach with other technologies, such as multi-key homomorphic encryption, threshold homomorphic encryption, and proxy re-encryption, to explore the equilibrium between security and performance under broader conditions.

References

[1] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. Order preserving encryption for numeric data. In *Proc. of SIGMOD*, page 563–574, New York, NY, USA, 2004.

[2] Asma Aloufi, Peizhao Hu, Yongsoo Song, and Kristin Lauter. Computing blindfolded on data homomorphically encrypted under multiple keys: A survey. *ACM CSUR.*, 54(9), October 2021. ISSN 0360-0300.

[3] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM TISSEC*, 9(1):1–30, 2006.

[4] Sara Babakniya, Ahmed Roushdy Elkordy, Yahya H Ezzeldin, Qingfeng Liu, Kee-Bong Song, Mostafa El-Khamy, and Salman Avestimehr. Slora: Federated parameter efficient fine-tuning of language models. *arXiv preprint arXiv:2308.06522*, 2023.

[5] Jiamu Bai, Daoyuan Chen, Bingchen Qian, Liuyi Yao, and Yaliang Li. Federated fine-tuning of large language models under heterogeneous tasks and client resources. In *Proc. of NeurIPS*, 2024.

[6] Mislav Balunovic, Dimitar Dimitrov, Nikola Jovanović, and Martin Vechev. Lamp: Extracting text from gradients with language model priors. In *Proc. of NeurIPS*, 2022.

[7] Ayoub Benaissa, Bilal Retiat, Bogdan Cebere, and Alaa Eddine Belfedhal. Tenseal: A library for encrypted tensor operations using homomorphic encryption. arxiv 2021. *arXiv preprint arXiv:2104.03152*, 2021.

[8] Daniel J Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Javier Fernandez-Marques, Yan Gao, Lorenzo Sani, Hei Li Kwing, Titouan Parcollet, Pedro PB de Gusmão, and Nicholas D Lane. Flower: A friendly federated learning research framework. *arXiv preprint arXiv:2007.14390*, 2020.

[9] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In *Proc. of TACT*, pages 127–144. Springer, 1998.

[10] Alexandra Boldyreva, Nathan Chenette, Younho Lee, and Adam O'neill. Order-preserving symmetric encryption. In *Proc. of EUROCRYPT*, pages 224–241. Springer, 2009.

[11] Shuaijun Chen, Omid Tavallaie, Niousha Nazemi, and Albert Y Zomaya. Rbla: Rank-based-lora-aggregation for fine-tuning heterogeneous models in flaas. In *Proc. of ICWS*, 2024.

[12] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *Proc. of ASIACRYPT*, 2017.

[13] Yae Jee Cho, Luyang Liu, Zheng Xu, Aldi Fahrezi, and Gauri Joshi. Heterogeneous lora for federated fine-tuning of on-device foundation models. In *Proc. of EMNLP*, 2024.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[15] Zane Durante, Qiuyuan Huang, Naoki Wake, Ran Gong, Jae Sung Park, Bidipta Sarkar, Rohan Taori, Yusuke Noda, Demetri Terzopoulos, Yejin Choi, et al. Agent ai: Surveying the horizons of multimodal interaction. *arXiv preprint arXiv:2401.03568*, 2024.

[16] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *Proc. of ICLR*, 2023.

[17] Xinyang Geng and Hao Liu. Openllama: An open reproduction of llama, May 2023. URL `https://github.com/openlm-research/open_llama`.

[18] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proc. of STOC*, pages 169–178, 2009.

[19] Pengxin Guo, Shuang Zeng, Yanran Wang, Huijie Fan, Feifei Wang, and Liangqiong Qu. Selective aggregation for low-rank adaptation in federated learning. *arXiv preprint arXiv:2410.01463*, 2024.

[20] Junhao Han and Li Yan. Adaptive batch homomorphic encryption for joint federated learning in cross-device scenarios. *IEEE IoT-J*, 11(6):9338–9354, 2023.

[21] Babak Hassibi, David G Stork, and Gregory J Wolff. Optimal brain surgeon and general network pruning. In *Proc. of ICNN*, pages 293–299. IEEE, 1993.

[22] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *Proc. of ICLR*, 2021.

[23] Chenghao Hu and Baochun Li. Maskcrypt: Federated learning with selective homomorphic encryption. *IEEE TDSC*, 2024.

[24] Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *Proc. of ICLR*, 2022.

[25] Qiuyuan Huang, Naoki Wake, Bidipta Sarkar, Zane Durante, Ran Gong, Rohan Taori, Yusuke Noda, Demetri Terzopoulos, Noboru Kuno, Ade Famoti, et al. Position paper: Agent ai towards a holistic intelligence. *arXiv preprint arXiv:2403.00833*, 2024.

[26] Jinwoo Jeon, Kangwook Lee, Sewoong Oh, Jungseul Ok, et al. Gradient inversion with generative image prior. In *Proc. of NeurIPS*, 2021.

[27] Weizhao Jin, Yuhang Yao, Shanshan Han, Jiajun Gu, Carlee Joe-Wong, Srivatsan Ravi, Salman Avestimehr, and Chaoyang He. Fedml-he: An efficient homomorphic-encryption-based privacy-preserving federated learning system. *arXiv preprint arXiv:2303.10837*, 2023.

[28] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends in Machine Learning*, 14(1–2), 2021.

[29] Renuga Kanagavelu, Zengxiang Li, Juniarto Samsudin, Yechao Yang, Feng Yang, Rick Siow Mong Goh, Mervyn Cheah, Praewpiraya Wiwatphonthana, Khajonpong Akkarajitsakul, and Shangguang Wang. Two-phase multi-party computation enabled privacy-preserving federated learning. In *Proc. of CCGrid*, 2020.

[30] Virginia Klema and Alan Laub. The singular value decomposition: Its computation and some applications. *IEEE TAC*, 25(2):164–176, 1980.

[31] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. In *Proc. of SPM*, 2020.

[32] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proc. of AISTATS*, 2017.

[33] Fanxu Meng, Zhaohui Wang, and Muhan Zhang. Pissa: Principal singular values and singular vectors adaptation of large language models. In *Proc. of NeurIPS*, 2024.

[34] Vaikkunth Mugunthan, Antigoni Polychroniadou, David Byrd, and Tucker Hybinette Balch. Smpai: Secure multi-party computation for federated learning. In *Proc. of NeurIPS*, 2019.

[35] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proc. of EUROCRYPT*, 1999.

[36] Ivo Petrov, Dimitar I Dimitrov, Maximilian Baader, Mark Müller, and Martin Vechev. Dager: Exact gradient inversion for large language models. In *Proc. of NeurIPS*, 2024.

[37] Simon Queyrut, Valerio Schiavoni, and Pascal Felber. Mitigating adversarial attacks in federated learning with trusted execution environments. In *Proc. of ICDCS*, 2023.

[38] Ronald L Rivest, Len Adleman, Michael L Dertouzos, et al. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978.

[39] Shangchao Su, Bin Li, and Xiangyang Xue. Fedra: A random allocation strategy for federated tuning to unleash the power of heterogeneous clients. In *Proc. of ECCV*, 2024.

[40] Mingjie Sun, Xinlei Chen, J Zico Kolter, and Zhuang Liu. Massive activations in large language models. In *Proc. of ICLR Workshop*, 2024.

[41] Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. In *Proc. of ICLR*, 2024.

[42] Youbang Sun, Zitao Li, Yaliang Li, and Bolin Ding. Improving LoRA in privacy-preserving federated learning. In *Proc. of ICLR*, 2024.

[43] Chunlin Tian, Zhan Shi, Zhijiang Guo, Li Li, and Chengzhong Xu. Hydralora: An asymmetric lora architecture for efficient fine-tuning. In *Proc. of NeurIPS*, 2024.

[44] Paul Voigt and Axel Von dem Bussche. The eu general data protection regulation (gdpr). *A practical guide, 1st ed., Cham: Springer International Publishing*, 10(3152676):10–5555, 2017.

[45] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proc. of ICLR*, 2019.

[46] Fei Wang and Baochun Li. Data reconstruction and protection in federated learning for fine-tuning large language models. *IEEE TBD*, pages 1–13, 2024.

[47] Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, et al. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. In *Proc. of EMNLP*, 2022.

[48] Ziyao Wang, Zheyu Shen, Yexiao He, Guoheng Sun, Hongyi Wang, Lingjuan Lyu, and Ang Li. Flora: Federated fine-tuning large language models with heterogeneous low-rank adaptations. *arXiv preprint arXiv:2409.05976*, 2024.

[49] Yuxuan Yan, Qianqian Yang, Shunpu Tang, and Zhiguo Shi. Federa: Efficient fine-tuning of language models in federated learning leveraging weight decomposition. *arXiv preprint arXiv:2404.18848*, 2024.

[50] Liping Yi, Han Yu, Gang Wang, Xiaoguang Liu, and Xiaoxiao Li. pfedlora: model-heterogeneous personalized federated learning with lora tuning. *arXiv preprint arXiv:2310.13283*, 2023.

[51] Da Yu, Saurabh Naik, Arturs Backurs, Sivakanth Gopi, Huseyin A Inan, Gautam Kamath, Janardhan Kulkarni, Yin Tat Lee, Andre Manoel, Lukas Wutschitz, Sergey Yekhanin, and Huishuai Zhang. Differentially private fine-tuning of language models. In *Proc. of ICLR*, 2022.

[52] Jianyi Zhang, Saeed Vahidian, Martin Kuo, Chunyuan Li, Ruiyi Zhang, Tong Yu, Guoyin Wang, and Yiran Chen. Towards building the federatedgpt: Federated instruction tuning. In *Proc. of ICASSP*, 2024.

[53] Chao Zheng, Liming Wang, Zhen Xu, and Hongjia Li. Optimizing privacy in federated learning with mpc and differential privacy. In *Proc. of CACML*, 2024.

[54] Yifeng Zheng, Shangqi Lai, Yi Liu, Xingliang Yuan, Xun Yi, and Cong Wang. Aggregation service for federated learning: An efficient, secure, and more resilient realization. *IEEE TDSC*, 20(2):988–1001, 2022.

[55] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. In *Proc. of NeurIPS*, 2019.

[56] Meilu Zhu, Axiu Mao, Jun Liu, and Yixuan Yuan. Deer: Deviation eliminating and noise regulating for privacy-preserving federated low-rank adaptation. *IEEE TMI*, 44(4):1783–1795, 2025.

A Preliminary

A.1 Singular Value Decomposition

Singular value decomposition (SVD) [30] is a mathematical method that decomposes any real or complex matrix into the product of three standard matrices. Applicable to matrices of arbitrary shapes, it can be adopted for tasks such as dimensionality reduction, data compression, and recommendation systems. For a real matrix $A \in \mathbb{R}^{m\times n}$, SVD decomposes it as follows: $A = U\Sigma V^\top$, where $U \in \mathbb{R}^{m\times m}$ is an orthogonal matrix, $\Sigma \in \mathbb{R}^{m\times n}$ is a diagonal matrix with non-negative real numbers on its diagonal, known as singular values, arranged in descending order, and $V^\top \in \mathbb{R}^{n\times n}$ is an orthogonal matrix as well. To match heterogeneous LoRAs from clients, many methods [49, 4] have employed SVD decomposition, breaking down the aggregated matrix into $B = U\sqrt{\Sigma}$ and $A = \sqrt{\Sigma}V^\top$, which enables the projection of the original parameters into a lower-dimensional space while preserving the essential features.

A.2 Federated Parameter-Efficient Fine-Tuning

Federated Parameter-Efficient Fine-Tuning (PEFT) is a technology designed for the efficient fine-tuning of large models within a distributed learning framework that prioritizes data privacy. For instance, consider LoRA [24], a well-established method for high-efficiency parameter fine-tuning. Assuming that the weight matrix of the global pre-trained model is denoted as $\mathbf{W} \in \mathbb{R}^{m\times n}$, LoRA introduces trainable parameters, $B \in \mathbb{R}^{m\times r}$ and $A \in \mathbb{R}^{r\times n}$. Each client freezes the original weights $\mathbf{W}$ and learns only the trainable low-rank parameters B and A on its local data. Then, the updates (e.g., weights or gradients) from clients are aggregated by the server via

$$\Delta\bar{\mathbf{w}} = \sum_{i=1}^{N} \alpha_i B_i A_i, \tag{5}$$

where α_i indicates the weight of client i. Federated PEFT uses the Eq. (5) to guarantee heterogeneous LoRA exact parameter aggregation. SVD is used to reparameterize the aggregation result into heterogeneous LoRA parameters to reduce the communication overhead of the result $\Delta\bar{\mathbf{w}}$. The new LoRA parameters are then sent back to each device for continued local training, and the process is iterated repeatedly to progressively optimize the model. It is worth noting that the model parameters are transmitted in plaintext throughout this process, making them visible to the server.

A.3 Homomorphic Encryption

Homomorphic encryption [38, 18] is a cryptographic primitive that allows computations to be performed on encrypted data without revealing the underlying plaintext. It is exceptionally well-suited for FL, as it enables the computation of aggregation in server without exposing clients' updates. The Cheon–Kim–Kim–Song (CKKS) encryption scheme [12] supports approximate numerical computations and floating-point arithmetic. The CKKS offers relatively high computational efficiency and supports vectorized operations, making it highly suitable for LoRA tuning in FL. It can be used to encrypt the parameters or gradients of local models, allowing the server to perform model aggregation without decrypting any data, thereby protecting user privacy. In the implementation of SHE-LoRA, we employ the CKKS provided by the TenSEAL library[3], which supports homomorphic computations on both vectors and tensors, thereby enabling encrypted computation operations over complex model parameters.

A.4 Order-Preserving Encryption

Order-Preserving Encryption (OPE) [1, 10] is a cryptographic technique that preserves the numerical order of plaintexts. If two plaintexts, a and b, satisfy the condition $a < b$, then their encrypted ciphertexts will also satisfy $Enc(a) < Enc(b)$. This enables comparisons, sorting, or range queries to be performed on ciphertexts without decryption. In principle, OPE maps plaintexts to an interval within the ciphertext space, while ensuring that the mapping function is monotonically increasing. Although OPE does not provide semantic security in the traditional sense (i.e., it is possible to infer

[3] https://github.com/OpenMined/TenSEAL

the approximate range of the plaintext from the ciphertext), it is highly useful in applications that require sorting or range operations on encrypted data, such as encrypted databases or cloud storage queries. In SHE-LoRA, we employ OPE to hide clients' encryption subset, which prevents the server from snooping on the positions of important model parameters and conducting targeted attacks.

A.5 Data Reconstruction via Inversion Attacks

Inversion attacks [36, 6, 55, 26] aim to reconstruct private training data or model parameters, such as pixel values in images or sensitive information in text, by reversing the clients' updates uploaded during federated fine-tuning, such as gradients, model parameter updates or prediction results. Specifically, for a model $f_{\mathbf{W}}(x) = \mathbf{W}^\top x$, a client trains it with its local data (x, y), and calculates the local gradient g as the derivative of the loss function $\mathcal{L}$:

$$g = \frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}}{\partial f_{\mathbf{W}}} \frac{\partial f_{\mathbf{W}}}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}}{\partial f_{\mathbf{W}}} \cdot x, \tag{6}$$

and uploads it to the server. The uploaded gradient g contains a linear combination of the original data. An attacker can analyze the principal components of the model update to search for the client's data distribution space and then reconstruct the data [36].

A.6 Matrix Multiplication in LoRA Amplifies DP Noise.

Differential privacy [51, 56] is a common privacy defense method that adds specific noise to updated gradients or parameters, making the relationship between gradients and data non-linear, which misleads the attacker's optimization direction. Given that the LoRA fine-tuning parameter is $\Delta W = BA$, and ΔW plays a role during model inference, if a noise ϵ satisfying the privacy budget is added to the parameters, then:

$$\Delta W = (B + \epsilon_B)(A + \epsilon_A) = BA + B\epsilon_A + A\epsilon_B + \epsilon_A\epsilon_B \tag{7}$$

Thus, except for BA, all the terms in Eq. (7) are noises that will be aggregated into the global model parameters, which may severely affect the model's performance and convergence direction.

B Related Work

LoRA Tuning in FL. Among existing PEFT techniques, LoRA [24] stands out due to its excellent performance and lesser trainable parameters in Federated PEFT. The application of efficient fine-tuning techniques in FL can reduce communication overhead and enhance overall efficiency. FedIT [52] employs FedAvg to aggregate updates from locally performed LoRA fine-tuning. FeDeRA [49] modifies the initialization of matrices A and B by the SVD results of pre-trained parameters. It helps mitigate the client drift that arises from the random initialization of A in the presence of data heterogeneity, thereby addressing the challenges of model convergence during the early stages of training. SLoRA [4] employs SVD to initialize matrices A and B, subsequently calculating a mask to reduce the parameters to train and communication costs. FedRA [39] partitions the pre-trained model by layer, enabling clients to fine-tune specific layers while the server aggregates the results to produce the whole model. Although it reduces training costs, the model remains fundamentally homogeneous. HydraLoRA [43] is inspired by the mixture of experts (MOE) architecture and learns multiple LoRAs corresponding to different knowledge. PiSSA [33] directly fine-tunes the principal component of the pre-trained model, facilitating rapid convergence and enhancing overall performance. These LoRA variants primarily focus on reducing training costs in homogeneous settings, but neglect the heterogeneity of device capabilities and Non-IID data in cross-device federated PEFT scenarios.

Heterogeneous LoRA. FLoRA [48] contends that the aggregation used in FedIT [52] is flawed and employs stacking to aggregate parameters from heterogeneous clients. HeterLORA [13] implements different ranks on clients and aggregates heterogeneous LoRA modules through zero-padding. However, zero-padding may dilute certain parameters, when aggregating the raw model parameters and padded ones. Additionally, it redistributes LoRA modules in a heterogeneous manner via truncation. RBLA [11] designs a rank-based LoRA aggregation to prevent model parameter dilution caused by

zero-padding. FlexLoRA [5] synthesizes a complete set of LoRA weights from individual client contributions, and employs SVD for weight reparameterization, thereby fully leveraging heterogeneous client resources. Furthermore, pFedLoRA [50] aggregates adapters to facilitate personalized FL, and treats LoRA as a mechanism for knowledge transfer, allowing clients to locally train heterogeneous models while maintaining a homogeneous adapter. Although these works have made significant contributions to heterogeneous LoRA, they still suffer from potential privacy leakage risks under inversion attacks.

Privacy Preservation Techniques. Recent works have demonstrated that the model parameters or gradients transmitted during federated PEFT can be exploited via inversion attacks to reconstruct private training data [36, 6, 55, 26]. To address this issue, Yu et al. [51] employ the DP-SGD optimizer for fine-tuning, providing formal DP guarantees for model parameters. Considering that the DP noise may be amplified by LoRA multiplication, FFA-LORA [42] modifies the LoRA training procedure by freezing matrix A after initialization and applying DP solely to model parameter matrix B. However, it reduces the volume of trainable model parameters, which may negatively impact fine-tuning performance. Similarly, FedSA-LoRA [19] proposes to globally train model parameter matrix A while reserving matrix B for local training without participating in aggregation. Meanwhile, several approaches have been proposed to utilize HE for privacy preservation in FL. Han and Yan [20] proposes an adaptive, precision-lossless batch HE method that transforms model parameters into non-negative values to prevent overflow errors. Inspired by model pruning techniques that leverage sensitivity to eliminate redundant model parameters, FedML-HE [27] encrypts only a subset of sensitive model parameters to reduce the encryption overhead associated with HE. MaskCrypt [23] selects a consensus mask for SHE to minimize overhead across homogeneous devices. In summary, DP faces the dilemma of hindering model convergence or improving privacy protection, while existing HE methods struggle to balance privacy and efficiency in cross-device federated PEFT with LoRA, particularly under scenarios with Non-IID data and heterogeneous device capabilities.

C Key Management

In the configuration outlined herein, the system necessitates an honest-but-curious server to execute the aggregation of models. Then, a trusted third party is required to oversee the distribution of keys for both Order-Preserving Encryption (OPE) and Homomorphic Encryption (HE), with the assumption that it will not collude with the server. However, to relax above strict assumption in serverless scenarios, cryptographic primitives such as threshold homomorphic encryption, multi-key homomorphic encryption, and proxy re-encryption can be employed to achieve distributed secure computation. **Threshold Homomorphic Encryption** [2] is a cryptographic scheme that amalgamates the threshold cryptography and homomorphic encryption. It not only supports computations on encrypted data (homomorphism) but also enables joint decryption by multiple participants (thresholding), thereby enhancing both security and fault tolerance while preserving privacy. Decryption can only be accomplished with the collective participation of a predetermined number of participants, known as the threshold. **Proxy Re-Encryption** [9, 3] permits a semi-trusted third party (the proxy) to transform ciphertext encrypted for one party into ciphertext decryptable by another, all without accessing the original plaintext or either party's private keys. **Multi-Key Homomorphic Encryption** [2] enables homomorphic operations to be performed on user data encrypted under different keys, obviating the need for key sharing. The resultant ciphertext requires collaborative partial decryption by all participants, each utilizing their respective private key, to yield the final plaintext outcome. This mechanism ensures that joint computations can be performed in multi-party data collaborations while protecting the data privacy of each participant.

D Additional Technical Details of Our Method

D.1 The Algorithm of Encryption Subset Negotiation

The workflow of encryption subset negotiation is summarized as algorithm 1. Note that server keeps the $rank$ and budget γ of all clients. The number of encrypted columns is denoted as $k = \gamma \cdot n$, where n is the number of columns of the parameter matrix. As described in Section 3.1.2, the server receive a set of tuple (G, S) from clients as input, where G is the set of columns that needs HE, and S is their sensitivities. At Lines 1-3, the server first initiates the negotiation result and number of positions to be selected at each round. Then, the server maintains two shared lists: the *Common* list, which counts

Algorithm 1: Encryption Subset Negotiation

Input: $clients = \{rank, k, (G, S)\}_{i=1}^{N}$, a, b, c are three hyper-parameters of selection ratio.
Output: *results*: the column index of public encryption subset.

results,selected_num $\leftarrow \{\}, 0$;
Common $\leftarrow$ count each column occurrences in G from all clients;
Sensitivity $\leftarrow$ sort columns by S from all clients, and remove duplicate columns;
$\Gamma \leftarrow$ sort all the unique budget columns;
for *each budget* k *columns in* Γ **do**
    $\lambda \leftarrow$ update current budget by $k - selected\_num$;
    $P \leftarrow$ average sampling $(a * \lambda)$ columns from clients whose budget is k;
    $C \leftarrow$ get top-$(b * \lambda)$ columns from *Common*;
    $S \leftarrow$ get top-$(c * \lambda)$ columns from *Sensitivity*;
    results $\leftarrow$ result of k columns is $\{S, C, P, \{results\}\}$;
    selected_num $\leftarrow k$;
return *results*

Algorithm 2: Aggregation of Unencrypted Model Parameters

Input: $\{\Delta W^i\}_{i=1}^{N}$: Set of N matrices, each ΔW^i with shape $(m, n - k_i)$
Output: $\Delta \overline{W}_{\text{plain}}$: Aggregated matrix with shape (m, K)

$K \leftarrow n - \min(k_1, \ldots, k_N)$;
$\Delta \overline{W}_{\text{plain}} \leftarrow \mathbf{0}_{m \times K}$;
$Counts \leftarrow \mathbf{0}_{1 \times K}$;
for *each client* $i = 1$ **to** N **do**
    $c_i \leftarrow$ get column count of ΔW^i ;
    $\Delta \overline{W}_{\text{plain}}[:, : c_i] \leftarrow \Delta \overline{W}_{\text{plain}}[:, : c_i] + \Delta W^i$;
    $Counts[: c_i] \leftarrow Counts[: c_i] + 1$;
for $j = 1$ **to** K **do**
    if $Counts[j] > 0$ **then**
        $\Delta \overline{W}_{\text{plain}}[:, j] \leftarrow \Delta \overline{W}_{\text{plain}}[:, j] / Counts[j]$;
return $\Delta \overline{W}_{plain}$

the occurrences of each column, and the *Sensitivity* list, which sorts all columns according to their importance. At Line 4, budgets from all clients are sorted after deduplication. At Lines 5-11, the server repeats the process certain times, which determined by the number of unique budgets. For each process, the number of column positions to be negotiated, termed as λ, is calculated by the difference between current budget k columns and those, which have been determined. The server takes into account among *Common*, *Sensitivity*, and the preferred encryption subsets of individual clients (reflected in G_i) in different proportions and selects λ columns. By splicing the columns in this round of negotiation in front of the columns that have been negotiated, and the server will get the result of a new round of negotiation. Starting with the smallest columns of encrypted budget, the server gradually negotiates up to the maximum budget columns, while keeping the previous result. Finally, the server produces negotiated global encryption subset.

During the negotiation process, the final result involves the selection of three hyperparameters, and we analyze the impact of hyperparameters on the negotiation result in the Appendix F.3.

D.2 Aggregation of Unencrypted Model Parameters

In this section, algorithm 2 demonstrated how server aggregates the unencrypted model parameters, termed as $\Delta W^i_{\text{plain}}$, which is calculated the matrix multiplication of B and the unencrypted part of A(denoted as A_{plain}). For simplicity, we use $\mathbf{\Delta W}^i$ instead of $\mathbf{\Delta W}^i_{\text{plain}}$, which indicates the multiplication result of client i. The input of the algorithm is the set of plaintext matrices $\mathbf{\Delta W}$ from all N clients. At Line 1, the columns of aggregation result is initialized by $n - \min(k_1, \ldots, k_n)$,

which n is the number of all columns of frozen pre-train parameter matrix, and k_i indicates the number of encrypted columns of client i. At Lines 2-3, the server initializes the result to $\mathbf{0} \in \mathbb{R}^{m \times K}$ and sets a counter to count the number of clients contributing when each column is aggregated. At Lines 4-7, the server incorporates client parameters $\mathbf{\Delta W}^i$ into the results and updates the counter to record the number of clients contributing to each column. At Lines 8-11, the server averages the results based on the counters and returns the final aggregated result $R \in \mathbb{R}^{m \times K}$.

D.3 Aggregation of Encrypted Model Parameters

Similar to the above algorithm 2, the server needs to aggregate the encrypted model parameters as well. It should be noted that after the client performs column swapping according to the global encryption parameter subset (as shown in Figure 5 in Section 3.1.2), all client parameters' position are aligned, where the plaintext is left-aligned and the ciphertext is right-aligned. The algorithm 3 takes the set of cipher blocks $\mathbf{\Delta W}^i_{\text{cipher}} = \{C_i\}^n \in \mathbb{R}^{m \times k_i}$ as the input. At Lines 1-3, the server first identify the max number of cipher blocks, and initialize the results and counter. At Lines 4-8, the server combines encrypted model parameters $\mathbf{\Delta W}^i_{\text{cipher}}$ into the results and updates the counter to record the number of clients contributing to each block. Finally, at Lines 9-12, the server averages the encrypted model parameters block by block, and return the final aggregated encrypted model parameters, denoted as $\Delta \overline{W}_{\text{cipher}} = \{C^k\}_{k=1}^B$. Although the blocks of encrypted model parameters extends to B, each client only receives a part of the result matching its encryption budget.

Algorithm 3: Aggregation of Encrypted Model Parameters

Input: $\{\{C_i^k\}_{k=1}^{m_i}\}_{i=1}^N$: Ciphertexts from N clients, where m_i indicates blocks of client$_i$.
Output: $\{C^k\}_{k=1}^B$: Aggregated ciphers, $B = \max(m_1, m_2, \ldots, m_N)$

1 $B \leftarrow \max(m_1, m_2, \ldots, m_N)$;
2 $\{C^k\}_{k=1}^B \leftarrow \mathbf{0}$;
3 $\{n_k\}_{k=1}^B \leftarrow 0$;
4 **for** *block* $k = 1$ **to** B **do**
5 **for** *client* $i = 1$ **to** N **do**
6 **if** *blocks of client* $m_i \geq k$ **then**
7 $C^k \leftarrow C^k + C_i^k$;
8 $n_k \leftarrow n_k + 1$;
9 **for** *block* $k = 1$ **to** B **do**
10 **if** *counter*$n_k > 0$ **then**
11 $C^k \leftarrow C^k / n_k$;
12 **return** $\{C^k\}_{k=1}^B$

D.4 Reparameterization of LoRA

Note that the decryption and reparameterization of parameters is running on the client, since only the client maintains the private key for decryption. The client gets the two low rank matrices (B_p, A_p) and a set of cipher blocks $(\{C\}^k)$ from the server, where the two matrices need pad to $B_p \in \mathbb{R}^{m \times r}, A_p \in \mathbb{R}^{r \times n}$, and cipher blocks need to be decrypted to $\Delta \overline{W}_{\text{cipher}}$. The updated full-parameter for each client, termed as ΔW, can be formulated as two parts, the plaintext update $\Delta \overline{W}_{\text{plain}} = B_p A_p$ and the ciphertext update $\Delta \overline{W}_{\text{cipher}} \in \mathbb{R}^{r \times k}$ as shown in Eq. (8). In order to reparameterize the two parts of the model parameters into the parameter matrices $\hat{B}$ and $\hat{A}$ of LoRA, we first apply SVD and zero-padding to the ciphertext update to generate two low-rank matrices $(B_c \in \mathbb{R}^{m \times r}, A_c \in \mathbb{R}^{r \times n})$, which ensures the next fusion. The final LoRA parameter matrices$(\hat{B}, \hat{A})$ are calculated as follows:

$$
\begin{aligned}
\Delta W &= \Delta \overline{W}_{\text{plain}} + \Delta \overline{W}_{\text{cipher}} \\
&\xlongequal{\text{SVD}} B_p A_p + B_c A_c \\
&= (U_1\sqrt{\Sigma}_1)\sqrt{\Sigma}_1 V_1^\top + (U_2\sqrt{\Sigma}_2)\sqrt{\Sigma}_2 V_2^\top \\
&= \left[U_1\sqrt{\Sigma}_1, U_2\sqrt{\Sigma}_2\right]^{m\times(r+r)} \begin{bmatrix}\sqrt{\Sigma}_1 V_1^\top \\ \sqrt{\Sigma}_2 V_2^\top\end{bmatrix}^{(r+r)\times n} \\
&\xlongequal{\text{SVD}} (U_3\Sigma_3 V_3^\top)(U_4\Sigma_4 V_4^\top) \\
&= (U_3\Sigma_3 V_3^\top U_4\sqrt{\Sigma}_4)_{:,:r}(\sqrt{\Sigma}_4 V_4^\top)_{:r,:} \\
&= \hat{B}\hat{A}
\end{aligned}
\tag{8}
$$

E Distributability and Scalability

Our system implementation is predicated upon Flower[4] [8], an open-source FL framework developed by a team at the University of Oxford. Flower is designed to streamline the construction of FL systems while affording a high degree of flexibility and scalability. It supports a variety of mainstream machine learning frameworks, such as PyTorch, TensorFlow, and Hugging Face Transformers, rendering it suitable for researchers and engineers addressing FL requirements across diverse scenarios. Flower allows users to extensively configure the framework according to their specific needs, thereby accommodating various FL scenarios while offering substantial support for AI research. Based on Flower, our SHE-LoRA supports parallelized simulation and multi-machine deployment, capable of satisfying the distributed and scalable requirements inherent in real-world applications.

F Additional Experimental Results

F.1 Data Heterogeneity

Data heterogeneity in FL is a significant difference in the distribution, scale, and characteristics of local data of each participant in a distributed environment. We visualize the results of SHE-LoRA heterogeneous dataset partitioning by dividing the dataset into ten clients as an example on the IMDB datasets in the Fig. 9. From the perspective of data heterogeneity, the Fig. 9 presents the double differences of client data in the FL: 1). Client 5 shows a significant data magnitude imbalance, and its total sample size is only 2% of client 1, which leads to the aggravation of gradient update direction bias during model training. 2). Clients 2/4/7/8 show an extreme unimodal label distribution, which is non-independent and identical (non-IID) in the Dirichlet distribution parameter α=0.3, which may lead to the collapse of the feature space, the polarization of the decision boundary, and the gradient conflict during global aggregation of the local model.

Following the Flower Dadaset community's[5] division, the Fig. 10 illustrates the distribution of positive and negative samples (pos/neg) in each partition (Partition ID) of the IMDB dataset under three different partitioning strategies (IIdPartitioner, DirichletPartitioner, and ShardPartitioner [32]). Each subgraph represents a partitioner, with the horizontal axis being the percentage of samples in each partition and the vertical axis being the partition number. The different colors represent positive and negative categories.

The proportion of positive and negative samples in each partition is more balanced and consistent by IIdPartitioner and ShardPartitioner, reflecting the homogeneity of data distribution. However, under the way of DirichletPartitioner, the proportion of positive and negative samples in each partition is quite different, some partitions are almost all positive samples, and some partitions are dominated by negative samples, showing obvious data heterogeneity (Non-IID). Therefore, we use the DirichletPartitioner to simulate the inconsistent data distribution of each participant in the real scenario, so as to better study and evaluate the performance of the SHE-LoRA algorithm in the heterogeneous data environment.

[4] https://flower.ai/

[5] https://flower.ai/docs/datasets

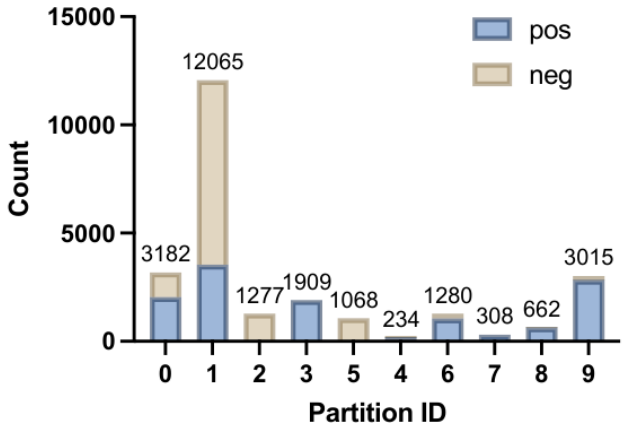


Figure 9: Non-IID data visualization for heterogeneous clients.

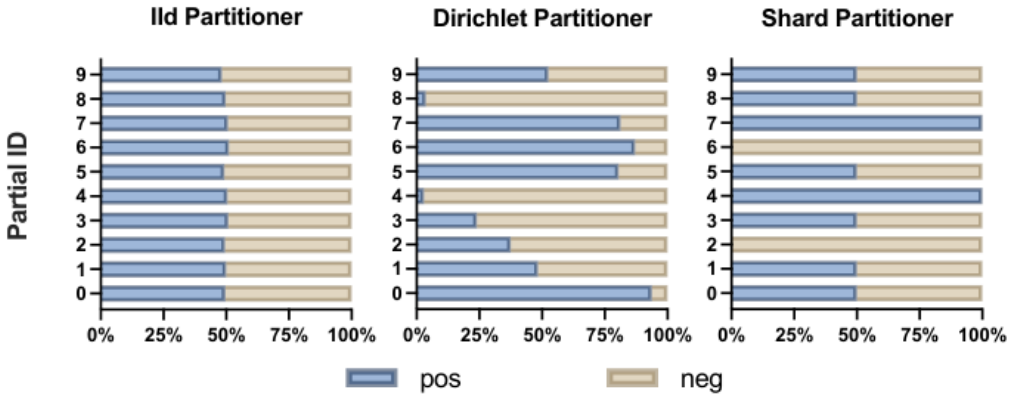


Figure 10: The comparison of label distribution under three dataset partitioning strategies.

F.2 Training Time Comparison

In addition, we compare the single-round training times of SHE-LoRA with those of FedIT, Flex-LoRA, and HeterLoRA on two models, Bert-Large and OpenLLaMA-3B, for different encryption ratios in the Fig. 11. The single-round training times of FedIT, Flex-LoRA, and HeterLoRA are very similar to one another. Among them, FlexLoRA brings a slight computational overhead on OpenLLaMA-3B due to the SVD decomposition, but this increase is minimal and not significant on slightly smaller matrix decompositions.

For SHE-LoRA, the increase in training time is due to the encryption of ciphertext blocks, the calculation of ciphertext, the decryption of ciphertext, and the decomposition of SVD, among which ciphertext calculation overhead is the largest. The total training time of the SHE-LoRA increases significantly by the calculation of ciphertext. As the encryption ratio increases, the number of ciphertext blocks becomes more and more, leading to a linear increase in the encryption cost and calculation overhead. Therefore, it is very important to choose the proper encryption ratio to balance the overhead and privacy that comes with encryption.

F.3 The Impact of Hyperparameter of Negotiation

For the three parameters, a, b, c, involved in the negotiation process in Appendix D.1, they indicate the proportion of encryption parameters selected from *Clients*, *Common* and *Sensitivity*, respectively. This mainly reflects whether the negotiation strategy focuses more on global or personalized privacy-sensitive locations. In the case of extreme heterogeneity, the elements in Common have a very low value and the negotiation result focuses more on personalized privacy-sensitive locations.

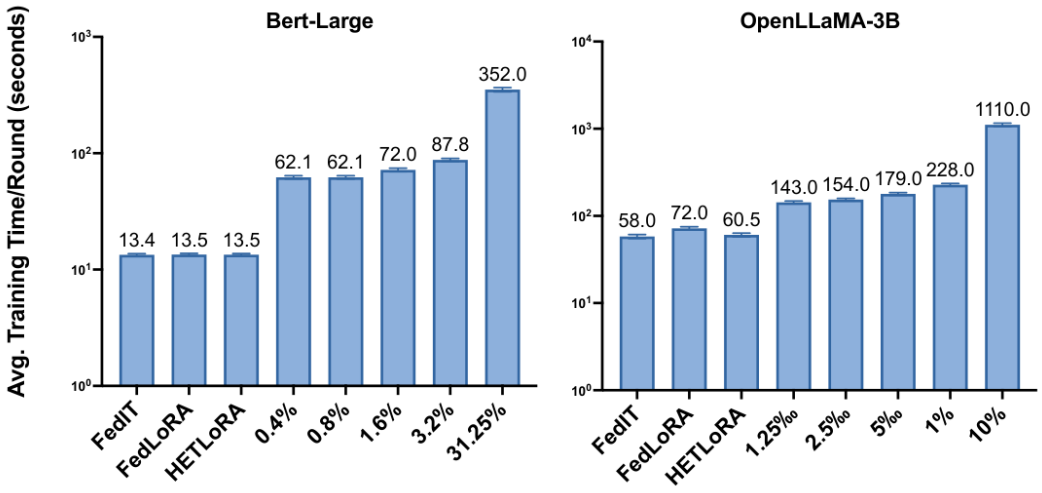


Figure 11: The Avg. training time comparison.

We set up four selection strategies to show the impact of negotiation results under different strategies as shown in the Fig. 12.

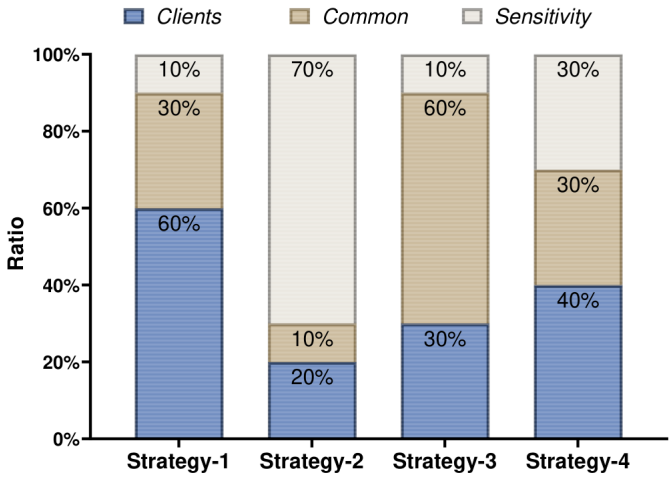


Figure 12: Distribution of three hyperparameters under different strategies.

- Strategy 1 selects 60% from *Clients*, 30% from *Common*, and 10% from *Sensitivity*, indicating that the selection results are more inclined to client personalization.
- Strategy 2 selects 20%,10%,70% respectively, which takes another opposite choice and selects more sensitive parameters.
- Strategy 3 chooses 30%,60%,10%,respectively, focusing more on public privacy sensitive parameter locations.
- And strategy 4 uses 40%,30%,30% a more balanced choice.

The negotiation results generated under the four selection strategies are shown in Fig. 13. It shows the sensitivity score of different clients (A, B, C) at each parameter position (location index) under four different negotiation strategies. The horizontal axis is the parameter location index (Index), and the vertical axis is the sensitivity score (Score) of the parameter. Color distinguishes different clients: red is Client A, blue is Client B, and green is Client C. The “Checked" and “Unchecked" states of

each client are represented in dark and light colors, respectively, indicating whether the parameters are selected. For example, the sensitive parameters of client 1 are centrally distributed on the left (index 1-5), the sensitive parameters of client 2 are relatively centered (index 3-7), and the sensitive parameters of client 3 are relatively concentrated on the right (index 5-10).

Strategy 1 tends to choose the more important parameter at index 3,5,7 across clients, and selects the parameter at index 1 when selecting the most sensitive parameter position, and then selects 6 randomly from 4 and 6 according to the number of occurrences. Strategy 2 selects index 9 from the perspective of client personalization, and selects others based on sensitive and common parameters. The choice of strategy 3 seems to be relatively balanced, with the most sensitive parameters and the most frequent parameters of all clients. The results of Strategy 4 show the diversity of sources, including not only the most sensitive parameters for each client, but also the parameters that are common and important and those (i.e., index 1) that are available for personalization.

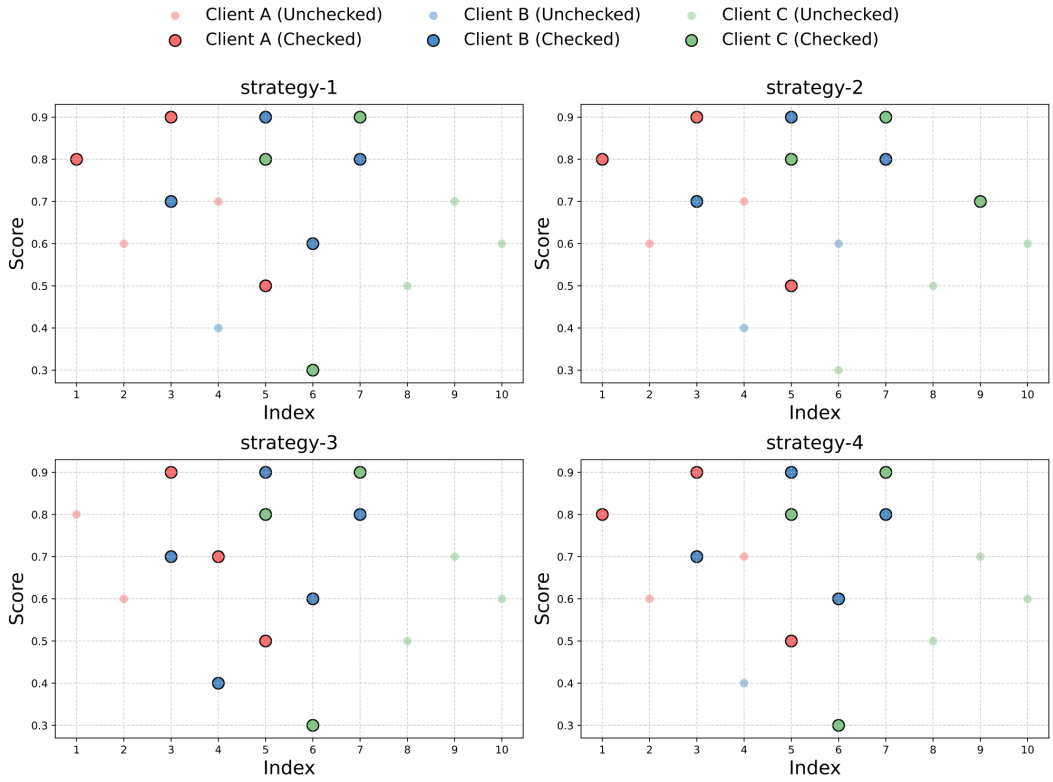


Figure 13: The impact of hyperparameter of negotiation.

F.4 The Impact of the Number of Clients on Encryption Time.

In SHE-LoRA, we assume that the clients dynamicly adjust the encryption subset to ensure that encryption and decryption are synchronized with the FL system without causing the client to lag behind. Therefore, the number of clients does not significantly increase the time for homomorphic encryption, but if each client adaptively selects the best encryption subset, the encryption time is limited to the encryption time of the longest client.

F.5 Performance

We employed FedIT [52] and FedSA [19] as baseline methods under homogeneous LoRA. FedIT averages LoRA weights across clients, limiting the rank to the capacity of the weakest device. FedSA trains matrix B locally while aggregating matrix A globally, leveraging federated learning to enhance the representational capacity of LoRA. Besides, we took FLoRA [48], HeterLoRA [13], and FlexLoRA [5] as baselines under heterogeneous settings. FLoRA employs stacking to reduce full

weight computations and achieve precise averaging across heterogeneous LoRA updates, but it comes with the trade-off of an expanded parameter space. HeterLoRA zero-pads all LoRA matrices to the global maximum rank, applies average aggregation similar to FedAvg, and subsequently truncates the aggregated weights to align with the local rank of each client. However, zero-padding introduces additional dilution in the aggregated parameters, which in turn leads to degraded model performance. Flex-LoRA reconstructs the full parameter matrix for each client by computing $B \times A$ and performs aggregation. Subsequently, the aggregated matrix is decomposed using singular value decomposition (SVD) and truncated according to the client's LoRA rank, producing a low-rank parameter matrix.

Table 3: MMLU Benchmark

Method	STEM	SS	Humanities	Average
FedIT [52]	21.5	21.3	20.4	21.2
FedSA [19]	21.8	21.4	19.7	20.1
HeterLoRA [13]	24.7	25.4	25.8	26
Flex-LoRA [5]	26.2	27.9	**26.6**	27.4
SHE-LoRA	**28.1**	**29.2**	26.5	**28.2**

Table 4: GLUE Benchmark

Method	SST2	MRPC	QQP	RTE	WNLI	QNLI
FedIT [52]	47.41	31.62	64.71	43.07	46.34	48.87
FedSA [19]	48.23	33.71	66.32	43.56	48.27	48.26
HeterLoRA [13]	55.73	68.38	72.17	44.72	48.86	49.14
Flex-LoRA [5]	52.29	**74.81**	**75.31**	46.93	49.66	49.51
SHE-LoRA	**57.11**	70.88	72.52	**50.18**	**57.75**	**59.63**

Natural Language Generation: According to the results in Table 3, FedIT and FedSA perform the worst on the MMLU Benchmark, obtaining scores of 21.2 and 20.1, respectively. These results indicate the limitations of traditional homogeneous approaches in heterogeneous LoRA settings, where the inability to effectively utilize client-specific information hinders overall performance. While HeterLoRA integrates parameters from heterogeneous devices to improve performance, its reliance on zero-padding leads to parameter dilution, resulting in inferior performance compared to Flex-LoRA. SHE-LoRA achieves the highest scores on STEM, Social Sciences(SS) and the overall Average, and matches Flex-LoRA's performance on Humanities. Both methods outperform all other baselines by a significant margin. These results indicate that SHE-LoRA better preserves informative updates in heterogeneous generative tasks, leading to improved generalization and performance.

Natural Language Understanding: Similarly, we reviewed on the six datasets of GLUE Benchmark in the Table 4, and the performances of FedIT and FedSA reaffirmed the limitations of traditional aggregation methods in heterogeneous scenarios. Flex-LoRA and SHE-LoRA, on the other hand, outperform the other methods, demonstrating that SHE-LoRA can more effectively update model parameters in heterogeneous environments while achieving performance comparable to non-private methods. Unsurprisingly, HeterLoRA achieves better performance than homogeneous baselines. However, it lags behind Flex-LoRA and SHE-LoRA, primarily due to the performance degradation caused by parameter dilution.

SHE-LoRA demonstrates strong performance across both benchmarks, achieving state-of-the-art results in heterogeneous settings while maintaining optimal performance despite the integration of privacy-preserving mechanisms.

G Limitations

As described in Section 2.3 and Appendix C, SHE-LoRA operates under the assumption of an honest-but-curious server, where all clients share the same HE key. Although secure communication channels can be used to defend against malicious clients or collusion between the server and clients, such mechanisms incur higher encryption costs. A promising direction for future work is to explore more efficient distributed parameter protection using techniques such as threshold homomorphic encryption, multi-key homomorphic encryption, or proxy re-encryption.

H Broader Impact

In this work, we leverage parameter sensitivity and SHE to ensure the secure aggregation of federated LoRA against inversion attacks such as DAGER. Such attacks are able to recover the original client data from clients' updates uploaded during federated PEFT, exacerbating privacy concerns and hindering the possibility of FL to extract value from distributed data. Our work offers adaptive and sufficient privacy preservation, while minimizing HE overhead per client in cross-device federated PEFT with LoRA.

Importantly, we find that with more sensitive model parameters being encrypted, the mutual information that can be leaked from the model updates drops dramatically, indicating that it is possible to effectively reduce the risk of privacy leakage in terms of privacy information as long as the sensitive model parameters are correctly encrypted. Our work implies that critical information within the model parameters can be soundly protected against the state-of-the-art attacks by merely encrypting less than 1% of the model parameters. Furthermore, we take into account the heterogeneity of the parameter sensitivity and encryption capabilities across clients, and broadly adapt the cost-effective SHE-LoRA to accommodate clients with diverse data distributions and device capabilities. With these observations, we highlight the feasibility and effectiveness of applying tailored and secure privacy protection for cross-device federated PEFT at much lower overhead compared to existing off-the-shelf privacy protection techniques.