SenseCrypt: Sensitivity-guided Selective Homomorphic Encryption for Cross-Device Federated Learning

Borui Li [1], **Li Yan** [1], **Junhao Han** [1], **Jianmin Liu** [1], **Lei Yu** [2]
[1] School of Cyber Science and Engineering
Xi'an Jiaotong University
Shaanxi, China 710049
[2] Department of Computer Science
Rensselaer Polytechnic Institute
Troy, NY USA 12180
boruili@stu.xjtu.edu.cn

Abstract

Homomorphic Encryption (HE) prevails in securing Federated Learning (FL), but suffers from high overhead and adaptation cost. Selective HE methods, which partially encrypt model parameters by a global mask, are expected to protect privacy with reduced overhead and easy adaptation. However, in cross-device scenarios with heterogeneous data and system capabilities, traditional Selective HE methods deteriorate client straggling, and suffer from degraded HE overhead reduction performance. Accordingly, we propose *SenseCrypt*, a **Sen**sitivity-guided **se**lective Homomorphic En**Crypt**ion framework, to adaptively balance security and HE overhead per cross-device FL client. Given the observation that model parameter sensitivity is effective for measuring clients' data distribution similarity, we first design a privacy-preserving method to respectively cluster the clients with similar data distributions. Then, we develop a scoring mechanism to deduce the straggler-free ratio of model parameters that can be encrypted by each client per cluster. Finally, for each client, we formulate and solve a multi-objective model parameter selection optimization problem, which minimizes HE overhead while maximizing model security without causing straggling. Experiments demonstrate that *SenseCrypt* ensures security against the state-of-the-art inversion attacks, while achieving normal model accuracy as on IID data, and reducing training time by 58.4%∼88.7% as compared to traditional HE methods.

1 Introduction

Federated Learning (FL) is increasingly popular due to its ability to enable collaborative Machine Learning (ML) model training without exposing privacy-sensitive user data (Li et al. 2020; Kairouz et al. 2021). However, multiple studies have shown that the model parameters transmitted during FL training can be exploited via *inversion attacks* to reconstruct users' private data (Hitaj, Ateniese, and Pérez-Cruz 2017; Zhu, Liu, and Han 2019; Geiping et al. 2020). To defend against such attacks, Homomorphic Encryption (HE) has attracted much attention due to its strong privacy guarantee while enabling mathematical operation of encrypted model parameters (Zhang et al. 2020; Roth et al. 2022; Jiang, Wang, and Liu 2021; Hao et al. 2019; Han and Yan 2023; Chen et al. 2021; Xu et al. 2021; Zheng et al. 2023; Jin et al. 2023; Hu and Li 2024).

Nevertheless, applying HE per parameter (*Plain* HE) is computationally and communicatively expensive (Zhang et al. 2020; Jin et al. 2023). To mitigate this, various optimization strategies have been proposed. For instance, **Lightweight HE** reduces cryptographic complexity at the cost of weaker security (Jiang, Wang, and Liu 2021; Hao et al. 2019), while **Batch HE** encrypts multiple parameters together but can introduce accuracy loss from quantization or require non-standard framework adaptations (Zhang et al. 2020; Han and Yan 2023; Chen et al. 2021; Xu et al. 2021; Zheng et al. 2023). In contrast, **Selective HE** draws inspiration from model pruning research (LeCun, Denker, and Solla 1989), which demonstrates that many model parameters are non-essential. This approach encrypts only the most critical parameters identified via a shared *selective encryption mask*. A key advantage is its high compatibility with existing FL frameworks, as it avoids altering the core encryption or aggregation algorithms (Jin et al. 2023; Hu and Li 2024). However, existing Selective HE methods fail to balance security and efficiency in cross-device FL with Non-IID data and heterogeneous device capabilities. Our experimental studies in Appendix Sections C and D demonstrate that 1) the uniform encryption budgets (defined as the upper bound quantity for encrypting model parameters) used in these methods may exacerbate the straggler problem on clients with heterogeneous device capabilities (i.e., *system heterogeneity*), and 2) their unioned encryption masks may incur prohibitive overhead when the clients' data distributions diverge (i.e., *statistical heterogeneity*).

Driven by these limitations, we aim to adaptively balance security and HE overhead for each client in cross-device FL. To achieve this, we need to address the following challenges:

• **How to avoid the negative impact of statistical and system heterogeneity on Selective HE?** Multiple solutions have been proposed to deal with statistical heterogeneity issues in FL via privacy-preserving data similarity measurement and clustering of clients with IID data (Cho, Mathur, and Kawsar 2022; Liu, Guo, and Chen 2021). In the meantime, to mitigate the straggler problem, several methods have been proposed to screen clients by their training performance (Bonawitz et al. 2019; Chai et al. 2020; Zhou et al. 2023; Jiang et al. 2023). However, most of them rely on ad-

ditional components (e.g., sparsity of Convolutional Neural Networks, client scheduler) to measure data similarity or coordinate FL training, which creates extra burden for FL framework adaptation. We observe that in addition to weighing model parameter criticality in defending against inversion attacks, model parameter sensitivity can reflect data distribution as well (Appendix Section E). Given this fact, a non-intrusive while privacy-preserving method for guiding the selective HE of cross-device FL clients is expected.

• **How to adaptively balance security and HE overhead for each client?** Encrypting too many model parameters on clients with low device capability significantly degrades FL performance, while encrypting insufficient parameters increases data leakage risk. However, without a sound measurement of the clients' device capability and security impact caused by selective HE, it is challenging to adaptively tailor the straggler-free encryption budget per client, while ensuring sufficient model security.

Accordingly, we propose *SenseCrypt*, a **Sen**sitivity-guided **se**lective Homomorphic En**Crypt**ion framework, to adaptively achieve the optimal balance between model security and HE overhead for cross-device FL clients. Specifically, by exploiting model parameter sensitivity to represent data distribution and similarity measurement, we first design a privacy-preserving method to cluster the clients with IID data into their respective groups. Then, based on the bandwidth and device processing speed of the clients in FL, we establish a scoring mechanism to deduce the straggler-free encryption budget for each client per cluster. Finally, based on the sensitivity assessment result and encryption budget, we let each client formulate and solve a multi-objective model parameter selection optimization problem that aims to minimize HE overhead while maximizing model security without causing straggling.

2 Preliminaries and Motivations

2.1 Definition of Model Parameter Sensitivity

Suppose $\mathbf{W}$ represents the model parameters of a neural network. We denote the model loss function as $\mathcal{L}(\mathbf{W})$, and the gradients of the loss with respect to $\mathbf{W}$ as $\nabla_{\mathbf{W}}\mathcal{L}(\mathbf{W})$. Given a subset of the model parameters $\mathbf{w} \in \mathbf{W}$, the sensitivity of $\mathbf{w}$ is denoted as $\mathbf{\Gamma}(\mathbf{w}) \in \mathbb{R}^{|\mathbf{w}|}$, and the model parameters with $\mathbf{w}$ zeroed-out is denoted as $\mathbf{W}_{-\mathbf{w}}$. The sensitivity of $\mathbf{w}$ is defined as the change in the loss function after zeroing $\mathbf{w}$ (LeCun, Denker, and Solla 1989):

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

From Equation (1), the larger absolute loss change caused by the removal of $\mathbf{w}$, the more contribution $\mathbf{w}$ can make to the loss, and the more sensitive $\mathbf{w}$ is. Thus, $\mathbf{\Gamma}(\mathbf{w})$ also corresponds to the privacy risk on gradients that may be exposed via observing $\mathbf{w}$. Considering that it is computationally infeasible to calculate the sensitivity of any arbitrary subset of model parameters by forward-passing the model every time, we adopt the first-order Taylor expansion of $\mathcal{L}(\cdot)$ with respect to $\mathbf{w}$ at $\mathbf{W}$ to approximate $\mathbf{\Gamma}(\mathbf{w})$ as in (Xu, Koehn, and Murray 2022; Liang et al. 2022):

$$\mathbf{\Gamma}(\mathbf{w}) \approx |\mathbf{w}^\top \nabla_{\mathbf{W}}\mathcal{L}(\mathbf{W})|. \tag{2}$$

Since $\mathbf{w}$ and $\nabla_{\mathbf{W}}\mathcal{L}(\mathbf{W})$ can be easily obtained per client during FL training, the calculation overhead of $\mathbf{\Gamma}(\mathbf{w})$ is trivial as compared to the additional components used by the previous statistical and system heterogeneity solutions (Bonawitz et al. 2019; Chai et al. 2020; Zhou et al. 2023; Jiang et al. 2023).

2.2 Privacy Leakage Evaluation

We employ Mutual Information (MI) to quantify the privacy leakage caused by the selective encryption of model parameters. Given $\mathbf{W}$ and $\mathbf{W}_{-\mathbf{w}}$, their mutual information is defined as the expected value of the point-wise mutual information over their joint distribution:

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

where $p(x, y)$ represents the joint probability distribution, $p(x)$ and $p(y)$ represent the marginal distributions of $\mathbf{W}$ and $\mathbf{W}_{-\mathbf{w}}$ respectively. Considering that we aim to selectively encrypt the most privacy-sensitive model parameters $\mathbf{w}$, thus $I(\mathbf{W}; \mathbf{W}_{-\mathbf{w}})$ reflects the amount of privacy that can be leaked via the unencrypted ones (i.e., $\mathbf{W}_{-\mathbf{w}}$). The larger $I(\mathbf{W}; \mathbf{W}_{-\mathbf{w}})$ is, the higher privacy leakage risk that the selective encryption of $\mathbf{w}$ will result in.

2.3 Threat Model

Following existing Selective HE methods (Hu and Li 2024; Jin et al. 2023), we assume an *honest-but-curious* server that follows the FL protocol, yet attempts to infer sensitive information through inversion attacks (Hitaj, Ateniese, and Pérez-Cruz 2017; Zhu, Liu, and Han 2019; Geiping et al. 2020). We assume that the clients will use their public keys to selectively encrypt model parameters, and will not share their private keys to the server or the other clients as in (Zhang et al. 2020; Jin et al. 2023). We also assume that attackers can only launch inversion attacks via unencrypted model parameters. The protection against other forms of malicious activities is not the focus of this work, and we refer to existing methods for protection (Zheng et al. 2023; Queyrut, Schiavoni, and Felber 2023). To extend our threat model to defend against client collusion attacks, we introduce a dual-server setting in Appendix I. This setting decouples the key distribution and decryption operations, while the rest of the algorithmic procedure remains unchanged. The pseudocode is provided in Algorithm 2 in the Appendix.

3 System Design of *SenseCrypt*

3.1 Privacy-preserving Clustering of Clients

Given that model parameter sensitivity reflects data distribution (Appendix Section E), we aim to exploit it for privacy-preserving clustering of clients, ensuring that those with IID data are clustered together. Considering that both gradients and sensitivity vectors can reveal the change of loss values,

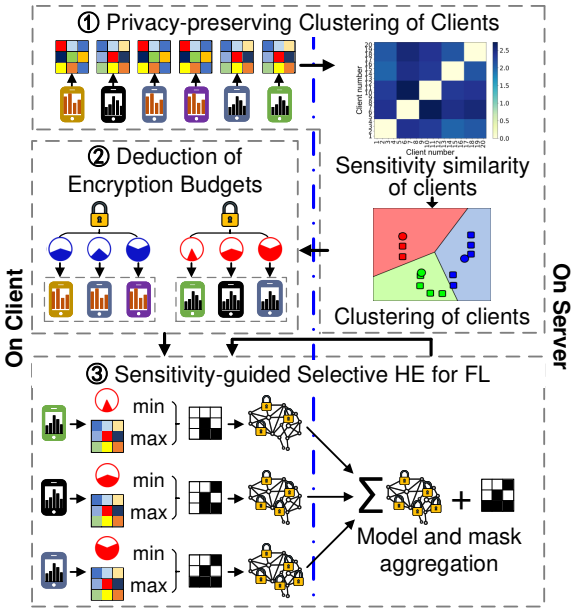


Figure 1: Framework of *SenseCrypt*.

we let each client only upload its sensitivity vector once during the first FL training iteration. Thus, clients' sensitivity vectors cannot be observed (like gradients) for inversion attack as in (Hitaj, Ateniese, and Pérez-Cruz 2017; Zhu, Liu, and Han 2019; Geiping et al. 2020).

Specifically, during the first FL training iteration, we apply Equation (2) to obtain the sensitivity vector of each client, and adopt the Euclidean Distance (Danielsson 1980) as the metric of similarity between sensitivity vectors for the sake of simplicity and generality. For the sensitivity vectors of the i-th client (denoted as $\mathbf{\Gamma}_i$) and the j-th client (denoted as $\mathbf{\Gamma}_j$), their similarity is measured as: $\text{sim}(\mathbf{\Gamma}_i, \mathbf{\Gamma}_j) = \sqrt{\sum_{k=1}^{N^{\mathbf{w}}}(\gamma_k^i - \gamma_k^j)^2}$, where γ_k^i and γ_k^j denote the k-th element in $\mathbf{\Gamma}_i$ and $\mathbf{\Gamma}_j$, respectively, and $N^{\mathbf{w}}$ denotes the total number of model parameters. Considering that the aggregation server has no prior knowledge on client data distribution (e.g., how many IID clusters the clients belong to), we use the Affinity Propagation (AP) method (Frey and Dueck 2007) to cluster the clients. Different from the K-means method (MacQueen 1967), AP automatically selects cluster centers "exemplars" by iteratively passing similarity and responsibility messages between data points, eliminating the need for predefined cluster numbers. Finally, the clients will be clustered into their respective groups, within which the clients' data distribution is IID to each other.

To preemptively mitigate potential inference attacks targeting sensitivity vectors, we introduce an optional Differential Privacy (DP) noise injection mechanism over sensitivity values. Appendix Section G empirically demonstrates the plausibility of privacy-preserving client clustering with DP-noised sensitivity vectors and the inherent robustness of AP to noise perturbations. The workflow of this component is illustrated as ① in Figure 1.

3.2 Deduction of Encryption Budgets

To avoid the negative impact of system heterogeneity, the encryption budgets of different clients must be set according to their system capabilities. Considering that device computation speed and communication bandwidth are the most decisive factors in determining the device's system capability in FL (Chai et al. 2020; Liu, He, and Cao 2023), we use them to deduce the clients' respective encryption budgets.

Specifically, suppose that in Section 3.1, the server has measured the bandwidth of each client per cluster (denoted as r_i for client i) based on the training delay and the amount of data transmitted during the iteration. Meanwhile, from the device specification of each client, the server can obtain client i's CPU clock speed v_i (e.g., 2.4 GHz×4 cores). Since the client's system capability in FL will always be bottlenecked by its most limited resource, we propose to use the lesser of r_i and v_i to deduce each client's encryption budget. However, r_i and v_i are measured on different scales and are independent of each other. Therefore, to score the clients' relative system capabilities, the server first normalizes them into the same scale (denoted as $\overline{r}_i \in [0,1]$ and $\overline{v}_i \in [0,1]$) via the Max Absolute Scaling: $\overline{u}_i = \frac{u_i}{\max\{|u_i|\}^{N^c}}$, where $u_i \in \{r_i, v_i\}$ is the original value of either r_i or v_i, and $\max\{|u_i|\}^{N^c}$ is the maximum absolute value of r_i or v_i over all the N^c clients in the cluster. Following this normalization, the system capability of client i can be represented as the minimum of $\overline{r}_i$ and $\overline{v}_i$, i.e., $\min\{\overline{r}_i, \overline{v}_i\}$. Finally, the server further normalizes each client's system capability, and sends it back to the client as its encryption budget $\alpha_i = \frac{\min\{\overline{r}_i, \overline{v}_i\}}{\max\{|\min\{\overline{r}_i, \overline{v}_i\}|\}^{N^c}}$. It is obvious that the fastest client's encryption budget will be 1, and α_i corresponds to the ratio of model parameters that the client can timely encrypt without straggling. The workflow of this component is illustrated as ② in Figure 1.

3.3 Sensitivity-guided Selective HE and Model Aggregation

Given the calculated model parameter sensitivity and encryption budget of each client, we elaborate: 1) the selection of model parameters for encryption on each client, which optimally balances model security and HE overhead , and 2) model aggregation without the straggler problem.

Model Parameter Selection Optimization Problem We use a binary mask vector $\mathbf{X}_i = [x_k^i | k = 1, \ldots, N^{\mathbf{w}}] \in \{0,1\}^{N^{\mathbf{w}}}$ to represent the respective encryption decision of model parameters. Specifically, if x_k^i is 1, it means the model parameter corresponding to x_k^i will be encrypted, or remain unencrypted if otherwise. On the one hand, we expect to minimize the HE overhead through properly selecting the model parameters for encryption. Thus, the first optimization objective can be represented as:

$$\min \quad f_1(\mathbf{X}_i) = \sum_{k=1}^{N^{\mathbf{w}}} x_k^i. \tag{4}$$

On the other hand, we expect to maximize the protection of the model parameters against privacy risks, i.e., select masks

that maximize the sum of sensitivity values corresponding to the masked parameters. Thus, the other objective can be represented as:

$$\max \quad f_2(\mathbf{X}_i) = \sum_{k=1}^{N^{\mathbf{w}}} x_k^i \gamma_k^i. \tag{5}$$

To minimize the straggler problem, the ratio of encrypted model parameters should be lower than the client's encryption budget α_i. Specifically, the straggler-free number of model parameters that the client can encrypt is estimated as $\lfloor \alpha_i N^{\mathbf{w}} \rfloor$, where $\lfloor \cdot \rfloor$ is the floor function. Thus, the constraint can be represented as:

$$\sum_{k=1}^{N^{\mathbf{w}}} x_k^i \leqslant \lfloor \alpha_i N^{\mathbf{w}} \rfloor. \tag{6}$$

Meanwhile, we must ensure that the total sensitivity measure of protected model parameters is sufficiently large to defend against inversion attacks. On the one hand, we expect the security protection level of the clients generally follows their encryption budgets, as the more model parameters a client can encrypt without straggling, the higher the level of security protection it can achieve (Chai et al. 2020). On the other hand, we don't expect the fast clients to excessively encrypt model parameters for the sake of HE overhead reduction. Therefore, we use an exponential function of the encryption budget to scale the security protection level, which is denoted as $1 - Ce^{-B\alpha_i}$, where C is the constant controlling the lower bound of security protection level, and B is the constant controlling the scaling step size of the exponential function. Compared to linear functions, the exponential function can more effectively prevent clients with excessive computational power from encrypting too many parameters, and renders a more gradual increase in the encryption budget as the client's computational capacity grows. Thus, the security protection level constraint can be represented as:

$$\frac{1}{\sum_{k=1}^{N^{\mathbf{w}}} \gamma_k^i} \sum_{k=1}^{N^{\mathbf{w}}} x_k^i \gamma_k^i \geqslant 1 - Ce^{-B\alpha_i}, \tag{7}$$

where γ_k^i is normalized by the sum of all the model parameters' sensitivity values $\sum_{k=1}^{N^{\mathbf{w}}} \gamma_k^i$ to differentiate the model parameters' relative privacy risk. A detailed analysis for the selection of B and C is provided in Appendix Section F. Following the definition in Section 2.2, we add another constraint to limit the privacy leakage caused by selectively encrypting $\mathbf{W}$ into $\mathbf{W}_{-\mathbf{w}} = (1 - \mathbf{X}_i) \odot \mathbf{W}$, where $\odot$ indicates element-wise multiplication:

$$I(\mathbf{W};\ (1 - \mathbf{X}_i) \odot \mathbf{W}) \leq \eta_{\text{MI}}. \tag{8}$$

where η_{MI} is the MI threshold, and can be tuned based on dataset complexity and privacy requirements. A lower η_{MI} value enforces stricter privacy guarantee at the cost of higher computational overhead. A detailed analysis of the relation between MI, attack success rate and encryption overhead is provided in Appendix Section H.

Finally, through combining the objectives and the constraints, the multi-objective model parameter selection optimization problem can be formulated as:

$$\begin{aligned} \min \quad & f_1(\mathbf{X}_i), \\ \max \quad & f_2(\mathbf{X}_i), \\ \text{s.t.} \quad & x_k^i \in \{0, 1\}, \\ & \text{Constraints}(6) \text{ and } (7) \text{ and } (8). \end{aligned} \tag{9}$$

To solve the Multi-Objective Binary Integer Programming (MOBIP) problem, it is necessary to make the following objective transformation: $F(\mathbf{X}_i) = [\min f_1(\mathbf{X}_i), \max f_2(\mathbf{X}_i)]^\top = \min[f_1(\mathbf{X}_i), -f_2(\mathbf{X}_i)]^\top$. As $f_l (l = 1, 2)$ are separately scaled, we normalize them by $\overline{f}_l = \frac{f_l - f_l^{\min}}{f_l^{\max} - f_l^{\min}}$, where $f_l^{\min}$ and $f_l^{\max}$ are the maximum and minimum values of f_l, respectively.

Then, we scalarize the MOBIP into a single-objective optimization problem: $\min\ \beta_1 \overline{f}_1(\mathbf{X}_i) - \beta_2 \overline{f}_2(\mathbf{X}_i)$, where $\beta_l > 0\ (l = 1, 2)$ are the weights of respective objective functions. Thus, the MOBIP problem is transformed to a variant of the Knapsack problem with $N^{\mathbf{w}}$ decision variables. Here, we assume equal importance for the objectives by setting $\beta_1 = \beta_2 = 1$, while noting that these weights can be readily customized for other scenarios without altering the overall framework. By applying Linear Programming (LP) relaxation (Cormen et al. 2022) on the problem, we can obtain the fractional solutions with $O(N^{\mathbf{w}} \log N^{\mathbf{w}})$ time complexity, which provides a lower bound of the optimal value. Then, through using algorithms such as rounding or branch and bound, we can further obtain the binary integer solutions to the problem (i.e., the optimal encryption mask vector $\mathbf{X}_i$). Finally, the client selectively encrypts its model parameters by $\mathbf{X}_i$, and uploads $\mathbf{X}_i$ to the server. Since the sensitivity of model parameters changes along with the progress of FL training, the sensitivity vector $\mathbf{\Gamma}_i$ and mask vector $\mathbf{X}_i$ will be updated per training iteration.

Model Aggregation without Straggling Considering that *FedAvg* (McMahan et al. 2017) has been proved as still one of the most robust FL aggregation strategies while maintaining computational simplicity (Jin et al. 2023), especially when the data is IID, we utilize it for model aggregation per cluster without losing generality. Moreover, since the Paillier HE supports the addition of ciphertext to plaintext (Paillier 1999), unlike the traditional Selective HE methods that aggregate models by the union of clients' mask vectors, *SenseCrypt* directly aggregates the clients' models, which are separately encrypted by their respective $\mathbf{X}_i$, without modifying *FedAvg*'s aggregation strategy. Thus, only the decryption of the aggregation result needs the union of the clients' mask vectors (denoted as $\widehat{\mathbf{X}} = \cup_{i=1}^{N^c} \mathbf{X}_i$).

Finally, the workflow of *SenseCrypt* is summarized as Algorithm 1 and illustrated in Figure 1. At Lines 1-2, each client calculates its sensitivity vector and encryption budget as in Section 2.1 and Section 3.2, respectively. At Line 3, the server applies the clustering of clients as in Section 3.1 to split the clients into respective clusters with IID data

Algorithm 1: Workflow of *SenseCrypt*.

1 $\mathbf{\Gamma}_i \leftarrow$ calculate sensitivity vector as in Section 2.1;
2 $\alpha_i \leftarrow$ calculate encryption budget as in Section 3.2;
3 $\{G_{\text{IID}}\} \leftarrow$ clustering of clients as in Section 3.1;
4 **for** *each iteration* $e = 1, 2, ...$ **do**
5 **for** *each group* G_{IID} **in parallel do**
6 **for** *client* $i \in G_{\text{IID}}$ **in parallel do**
7 $\mathbf{W}_i^e \leftarrow$ local model update as in *FedAvg*;
8 $\mathbf{\Gamma}_i \leftarrow$ update sensitivity vector as in Section 2.1;
9 $\mathbf{X}_i \leftarrow$ calculate mask vector as in Section 3.3;
10 $\mathbf{W}_i^{e,*} \leftarrow$ selective HE of $\mathbf{W}_i^e$ by $\mathbf{X}_i$;
11 $\mathbf{W}^{e+1,*} \leftarrow$ aggregate $\mathbf{W}_i^{e,*}$ as in *FedAvg*;
12 $\widehat{\mathbf{X}} \leftarrow$ union of the clients' uploaded $\mathbf{X}_i$;
13 **for** *client* $i \in G_{\text{IID}}$ **in parallel do**
14 $\mathbf{W}^{e+1} \leftarrow$ decryption of $\mathbf{W}^{e+1,*}$ by $\widehat{\mathbf{X}}$;

(denoted as $\{G_{\text{IID}}\}$). At Lines 6-10, each client updates its sensitivity vector and mask vector to selectively encrypt its model parameters. At Lines 11-12, the server aggregates the selectively encrypted models from the clients, and determines the mask vector $\widehat{\mathbf{X}}$ for decryption per cluster. At Lines 13-14, each client uses $\widehat{\mathbf{X}}$ to decrypt the global model.

4 Performance Evaluation

4.1 Experimental Settings

FL Datasets. We use the `CIFAR10`, `CIFAR100`, `MNIST` and `FMNIST` datasets for evaluations.

Models. For evaluations on the `FMNIST` and `CIFAR10` datasets, we train a Fully-Connected Neural (FCN) network and an AlexNet (Krizhevsky, Sutskever, and Hinton 2012), respectively. To evaluate effectiveness against inversion attacks, we train a ResNet-50 (He et al. 2016) on the `CIFAR100` dataset, and a LeNet-5 (LeCun et al. 1998) on the `MNIST` dataset.

Implementation. We use PyTorch to implement *SenseCrypt* on a server with 2 NVIDIA RTX 4090 GPUs, 104 Intel Xeon CPUs, and 256 GB memory. The total number of clients is 20. We adopt the python-paillier (Pyt 2013) as the HE implementation. The HE key size of each client is 2048. The privacy leakage threshold η_{MI}=2.0. For `FMNIST` and `CIFAR10`, we set {C=0.7, B=1.3} and {C=0.5, B=2}, respectively. For details on the selection of hyperparameters B and C, please refer to Appendix section F.

Experiment Scenarios. We design three scenarios:

• **Statistical Heterogeneity Scenario.** We evenly split the clients into 4 categories and ensure the clients' data is IID within the same category, but Non-IID across different categories. To exclusively evaluate the impact of statistical heterogeneity, each client has 5% of overall data, a bandwidth of 50 MBps and 32 CPUs.

• **System Heterogeneity Scenario.** We first evenly split the clients into 4 categories. Then, 1) for bandwidth heterogeneity, we assign 50, 45, 40, 35 and 30 MBps to the 5 clients per category, respectively, 2) for computation speed heterogeneity, we assign 24, 16, 12, 10 and 8 CPUs to the 5 clients per category, respectively. To exclusive evaluation of system heterogeneity, all the clients hold IID training data.

• **Statistical & System Heterogeneity Scenario.** We divide the clients into 4 categories, each containing 5 clients, and adopt the same settings as in the above scenarios.

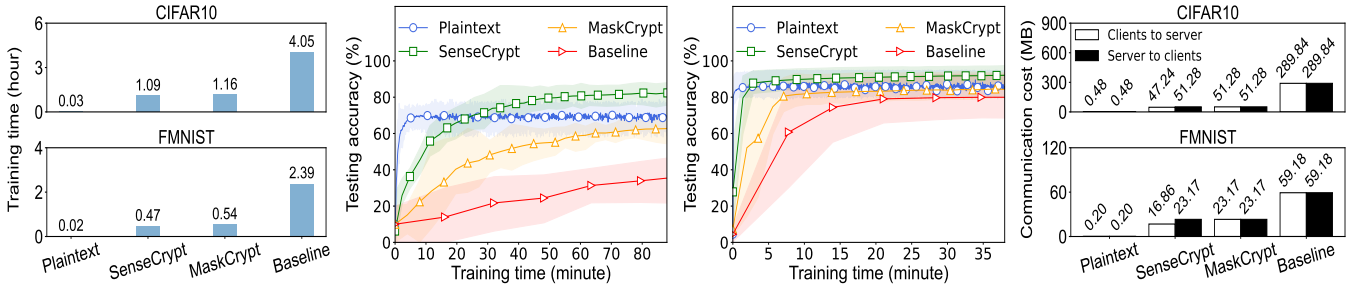
Comparison Methods. We compare *SenseCrypt* with a representative Selective HE method (denoted as *MaskCrypt*) (Hu and Li 2024), *FedAvg* with ciphertext encrypted by the Paillier HE method (denoted as *Baseline*) (Pyt 2013), and *FedAvg* with plaintext (denoted as *Plaintext*) (McMahan et al. 2017). Specifically, *MaskCrypt* utilizes the change of gradient value to select the model parameters that need encryption. *Baseline* utilizes the stock implementation of python-paillier (Pyt 2013) to encrypt each model parameter.

4.2 Experimental Results

Performance in Heterogeneity Scenarios Figures 2, 3 and 4 show the evaluation results of different methods in the three scenarios. For fairness, the union of masks in *SenseCrypt* has the same size as that in *MaskCrypt*.

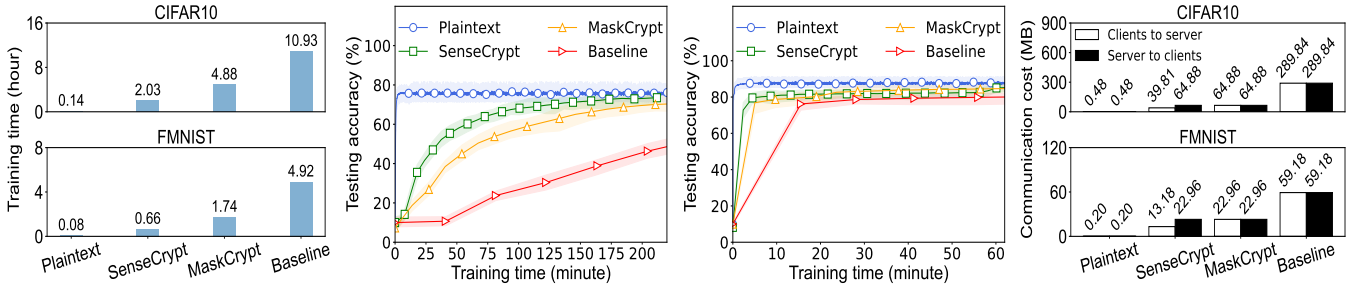
Figures 2a, 3a and 4a show the training time of the methods over 30 iterations in the three scenarios. We can see that *Plaintext* consistently results in the shortest training time as it spends no time on HE, whereas *Baseline* consistently results in the longest training time as it encrypts each model parameter. In comparison, *MaskCrypt* greatly shortens the training time. This is because that *MaskCrypt* only selectively encrypts partial model parameters by sensitivity. However, *MaskCrypt* still consumes $>33\times$ and $>20\times$ more time than that of *Plaintext* on `CIFAR10` and `FMNIST`, respectively, in both system heterogeneity and statistical & system heterogeneity scenarios. This is because that *MaskCrypt* cannot assign encryption masks in accordance with the clients' system capabilities, thereby deteriorates the straggler problem in these scenarios. In contrast, *SenseCrypt* reduces the training time by 58.4%$\sim$62.7% and 81.4%$\sim$88.7% as compared to *MaskCrypt* and *Baseline*, respectively, in Figures 3a and 4a. Even in the statistical heterogeneity scenario where clients share the same system capabilities (Figure 2a), *SenseCrypt* requires shorter training time than *MaskCrypt*. This is primarily due to the adaptive selection of model parameters for encryption in *SenseCrypt*, which not only considers the clients' system capabilities, but also adapts to their Non-IID data via client clustering. We also notice that the training time of *MaskCrypt* and *SenseCrypt* in Figure 4a is slightly shorter than that in Figure 3a. The primary reason is that, in *SenseCrypt*, the number of clients per cluster (i.e., 5 clients sharing IID data) in the statistical & system heterogeneity scenario is smaller than that in the system heterogeneity scenario (i.e., 20 clients sharing IID data), which causes the union of encryption masks per cluster shrinks in size. For fairness, the mask size in *MaskCrypt* also shrinks correspondingly.

Figures 2b, 2c, 3b, 3c, 4b and 4c show the testing accuracy of each client's trained model of different methods in the three scenarios. Each curve represents the mean testing accuracy of all the clients, and the shaded areas surrounding the curve represent the fluctuation range of testing ac-



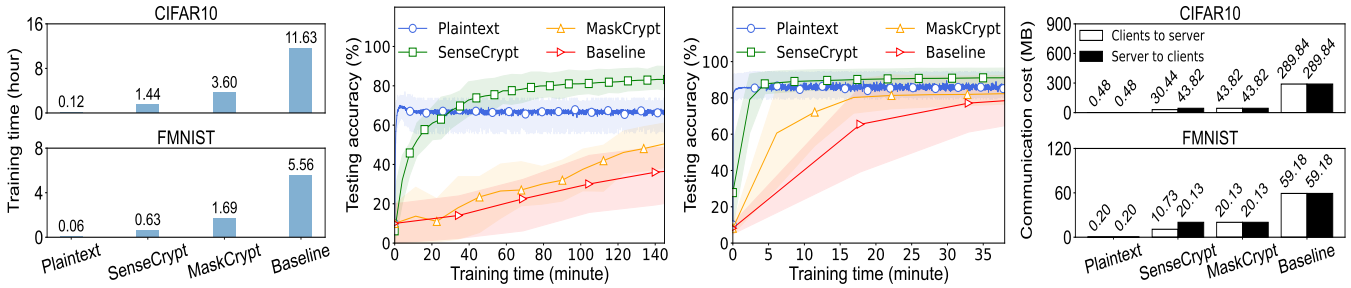
(a) Training time. (b) Performance on CIFAR10. (c) Performance on FMNIST. (d) Communication cost.

Figure 2: Performance comparison in statistical heterogeneity scenario.



(a) Training time. (b) Performance on CIFAR10. (c) Performance on FMNIST. (d) Communication cost.

Figure 3: Performance comparison in system heterogeneity scenario.



(a) Training time. (b) Performance on CIFAR10. (c) Performance on FMNIST. (d) Communication cost.

Figure 4: Performance comparison in statistical & system heterogeneity scenario.

curacies among the clients. We can see that despite the fast convergence rate of *Plaintext*, the clients' testing accuracy after convergence is significantly lower than that of *SenseCrypt* in the statistical heterogeneity scenario and the statistical & system heterogeneity scenario. Similarly, such model performance degradation in scenarios with high statistical heterogeneity can also be observed in *MaskCrypt* and *Baseline*. The reason is that these methods cannot adapt to the clients' Non-IID data, causing the global models trained via FL to fail to converge to satisfactory performance levels. In contrast, *SenseCrypt* can always achieve normal model performance levels as on IID data, while enjoying approximate convergence rate as *Plaintext*, especially on FMNIST. This is majorly due to the client clustering, which ensures that the clients per group have IID data, and the adaptive selection of model parameters for encryption, which prevents convergence delay caused by straggling.

Figures 2d, 3d and 4d show the communication cost of different methods incurred in one iteration in the three scenarios. The upstream communication cost (i.e., clients $\rightarrow$ server) and the downstream communication cost (i.e., server $\rightarrow$ clients) are separately illustrated. Given that the ciphertext generated by the Paillier HE method increases in size by more than $64\times$ as compared to the plaintext (Zhang et al. 2020), the fewer model parameters are encrypted in a method, the lower communication cost it will render. We can see that the results are generally consistent with the training time due to the same aforementioned reasons. Note that unlike the comparison methods, the upstream communication cost of *SenseCrypt* is always much lower than its downstream communication cost. This is because that *SenseCrypt* uses the adaptively determined mask to encrypt par-

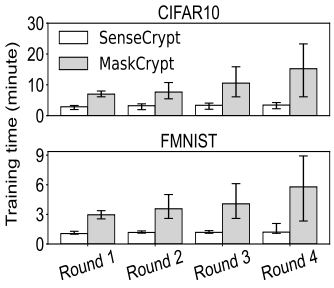


Figure 5: Training time under system heterogeneity.

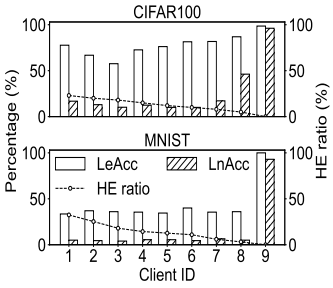


Figure 6: Attack results under extreme system heterogeneity.

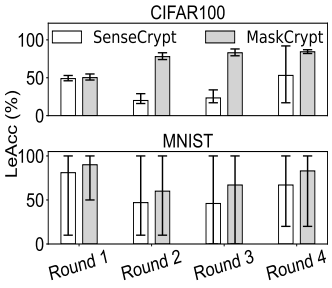


Figure 7: LeAcc under statistical heterogeneity.

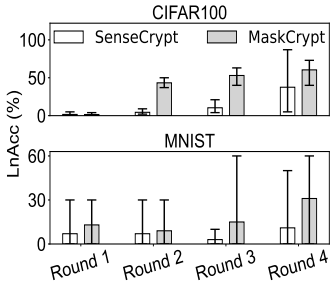


Figure 8: LnAcc under statistical heterogeneity.

tial model parameters of respective clients, thereby further reducing the communication cost.

Cost Efficiency under Varying Heterogeneity Degrees We further evaluate cost efficiency in terms of training time under varying system heterogeneity degrees. Specifically, we conduct FL training on 5 clients for 4 rounds and gradually increase the variance in the number of CPUs as: Round 1 ([32, 30, 28, 26, 24]), Round 2 ([32, 28, 24, 20, 16]), Round 3 ([32, 24, 18, 16, 12]) and Round 4 ([32, 16, 12, 10, 8]). To avoid the impact of statistical heterogeneity, we set that the clients have IID training data, 20% of overall data samples, and a bandwidth of 50 MBps. For fairness, the size of the union of masks is the same in *SenseCrypt* and *MaskCrypt*. Then, we measure each client's training time per iteration.

Figure 5 shows the minimum, mean and maximum values of all the measured results under each system heterogeneity degree on `CIFAR10` and `FMNIST`. We can see that as the system heterogeneity degree increases, the mean and fluctuation of the clients' training time in *MaskCrypt* also significantly rise on both datasets. In contrast, the training time of *SenseCrypt* remains nearly static across different system heterogeneity degrees, primarily due to the adaptive partial encryption of model parameters in accordance with the clients' system capabilities. We hence conclude that *SenseCrypt* can tolerate varying system heterogeneity degrees.

Effectiveness against Inversion Attacks Intuitively, under extremely high system heterogeneity, the slowest client's encryption mask determined by *SenseCrypt* might be very small, which may affect its security. To evaluate the effectiveness of *SenseCrypt* against inversion attacks in such cases, we further increase the system heterogeneity degree and launch inversion attacks to reconstruct each client's private data. Specifically, we conduct another round of FL training on 9 clients. The numbers of CPUs of Clients 1-8 follow [32, 24, 16, 12, 10, 8, 2, 1]. For reference, Client 9 trains without HE. The other settings are the same as in Section 4.2. Then, we launch the instance-wise Labels Restoration from Gradients (iLRG) attack (Ma et al. 2023) on each client, which utilizes the batch-averaged gradients to reconstruct each data sample and its label per batch. To evaluate the general effectiveness, the 1st attack is launched on ResNet-50 with a batch size of 200, a model depth (i.e., number of model layers) of 50, and the `CIFAR100` dataset. The 2nd attack is launched on LeNet-5 with a batch size of 30, a model depth of 7, and the `MNIST` dataset. Finally, we calculate the Label existence Accuracy (LeAcc), which measures the accuracy score for predicting label existences, and the Label number Accuracy (LnAcc), which measures the accuracy score for predicting the number of instances per class, of the reconstructed results as in (Ma et al. 2023).

Figure 6 shows the measured results of each client. We find that among Clients 1-8, the HE ratio decreases along with the decrement in the number of CPUs to avoid straggling. Interestingly, no matter how low the HE ratio is (as low as 5.7% on `CIFAR100` and 3.1% on `MNIST`), LeAcc remains almost static and LnAcc is significantly lower than that of Client 9 (no encryption). This confirms that the redundancy of model parameters allows the adaptive selective HE of model parameters while ensuring each client's security under extremely high system heterogeneity.

We additionally launch iLRG attacks under varying statistical heterogeneity degrees. Specifically, we conduct FL training on 10 clients for 4 rounds: 1) in the 1st round, each client has equal splits of all data classes (i.e., strictly IID); 2) in the 2nd, 3rd and 4th rounds, each client has all the data samples of 5, 2 and 1 classes, respectively. For the rest classes, each client has one data sample per class. For fairness, both *SenseCrypt* and *MaskCrypt* encrypt 5.0% and 3.1% of the model parameters on `CIFAR100` and `MNIST`, respectively.

Figures 7 and 8 show the minimum, mean and maximum values of the measured results of different methods per round. Note that the exceptionally high LeAcc in Round 1 of Figure 7 is caused by the label prediction logic of iLRG, which infers that all labels exist by default (Ma et al. 2023), thereby favoring the case with strictly IID data. We can see that along with the increase of Non-IID degree, the LeAcc and LnAcc of *MaskCrypt* significantly rise, while those of *SenseCrypt* almost remain static. This is primarily because that under a more Non-IID degree, the union of sensitive (important) model parameters that need encryption becomes larger, which cannot be fully covered by the shared encryption mask of *MaskCrypt*. While in *SenseCrypt*, the clustering of clients with IID data avoids the expansion of important model parameters that need encryption, thereby ensuring sufficient encryption of the important model parameters per cluster under varying Non-IID degrees.

5 Conclusion

We propose *SenseCrypt*, a Selective HE framework that clusters clients with IID data by model parameter sensitivity, assigns straggler-free encryption budget, and adaptively balances model security and HE overhead for cross-device FL clients with heterogeneous data and system capabilities. Our experiments conducted in multiple heterogeneity scenarios showed that compared with the state-of-the-art, *SenseCrypt* achieves normal model accuracy as on IID data, while reducing training time by 58.4%∼88.7%, and ensuring model security against the state-of-the-art inversion attacks.

References

2013. Python Paillier Library. https://github.com/data61/python-paillier. Accessed in June, 2024.

Abadi, M.; Chu, A.; Goodfellow, I.; McMahan, H. B.; Mironov, I.; Talwar, K.; and Zhang, L. 2016. Deep Learning with Differential Privacy. In *Proc. of CCS*.

Bonawitz, K.; Eichner, H.; Grieskamp, W.; Huba, D.; Ingerman, A.; Ivanov, V.; Kiddon, C.; Konečný, J.; Mazzocchi, S.; McMahan, B.; Van Overveldt, T.; Petrou, D.; Ramage, D.; and Roselander, J. 2019. Towards Federated Learning at Scale: System Design. In *Proc. of MLSys*.

Chai, Z.; Ali, A.; Zawad, S.; Truex, S.; Anwar, A.; Baracaldo, N.; Zhou, Y.; Ludwig, H.; Yan, F.; and Cheng, Y. 2020. TiFL: A Tier-Based Federated Learning System. In *Proc. of HPDC*.

Chen, W.; Ma, G.; Fan, T.; Kang, Y.; Xu, Q.; and Yang, Q. 2021. Secureboost+: A High Performance Gradient Boosting Tree Framework for Large Scale Vertical Federated Learning. *arXiv preprint arXiv:2110.10927*.

Cho, H.; Mathur, A.; and Kawsar, F. 2022. FLAME: Federated Learning across Multi-Device Environments. *ACM IMWUT*, 6(3).

Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; and Stein, C. 2022. *Introduction to Algorithms*. MIT press.

Danielsson, P.-E. 1980. Euclidean Distance Mapping. *Computer Graphics and Image Processing*, 14(3).

Ding, X.; ding, g.; Zhou, X.; Guo, Y.; Han, J.; and Liu, J. 2019. Global Sparse Momentum SGD for Pruning Very Deep Neural Networks. In *Proc. of NeurIPS*.

Fan, J.; Wu, K.; Tang, G.; Zhou, Y.; and Huang, S. 2024. Taking advantage of the mistakes: Rethinking clustered federated learning for iot anomaly detection. *IEEE TPDS*, 35(6).

Frey, B. J.; and Dueck, D. 2007. Clustering by passing messages between data points. *science*, 315(5814).

Geiping, J.; Bauermeister, H.; Dröge, H.; and Moeller, M. 2020. Inverting gradients-how easy is it to break privacy in federated learning? In *Proc. of NeurIPS*.

Han, J.; and Yan, L. 2023. Adaptive Batch Homomorphic Encryption for Joint Federated Learning in Cross-Device Scenarios. *IEEE IoT-J*, 11(6).

Hao, M.; Li, H.; Xu, G.; Liu, S.; and Yang, H. 2019. Towards Efficient and Privacy-Preserving Federated Deep Learning. In *Proc. of ICC*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. In *Proc. of CVPR*.

Hitaj, B.; Ateniese, G.; and Pérez-Cruz, F. 2017. Deep Models Under the GAN: Information Leakage from Collaborative Deep Learning. In *Proc. of CCS*.

Hu, C.; and Li, B. 2024. MASKCRYPT: Federated Learning With Selective Homomorphic Encryption. *IEEE TDSC*, Early Access.

Jiang, Z.; Wang, W.; and Liu, Y. 2021. Flashe: Additively symmetric homomorphic encryption for cross-silo federated learning. *arXiv preprint arXiv:2109.00675*.

Jiang, Z.; Xu, Y.; Xu, H.; Wang, Z.; and Qian, C. 2023. Heterogeneity-Aware Federated Learning With Adaptive Client Selection and Gradient Compression. In *Proc. of INFOCOM*.

Jin, W.; Yao, Y.; Han, S.; Joe-Wong, C.; Ravi, S.; Avestimehr, S.; and He, C. 2023. FedML-HE: An Efficient Homomorphic-Encryption-Based Privacy-Preserving Federated Learning System. In *Proc. of NeurIPS*.

Kairouz, P.; McMahan, H. B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A. N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; et al. 2021. Advances and Open Problems in Federated Learning. *Foundations and Trends® in Machine Learning*, 14(1–2).

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet Classification With Deep Convolutional Neural Networks. In *Proc. of NeurIPS*.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-Based Learning Applied to Document Recognition. *Proc. of the IEEE*, 86(11).

LeCun, Y.; Denker, J.; and Solla, S. 1989. Optimal Brain Damage. *Proc. of NeurIPS*.

Li, T.; Sahu, A. K.; Talwalkar, A.; and Smith, V. 2020. Federated Learning: Challenges, Methods, and Future Directions. *IEEE SPM*, 37(3).

Liang, C.; Jiang, H.; Zuo, S.; He, P.; Liu, X.; Gao, J.; Chen, W.; and Zhao, T. 2022. No Parameters Left Behind: Sensitivity Guided Adaptive Learning Rate for Training Large Transformer Models. In *Proc. of ICLR*.

Liu, B.; Guo, Y.; and Chen, X. 2021. PFA: Privacy-preserving Federated Adaptation for Effective Model Personalization. In *Proc. of WWW*.

Liu, H.; He, F.; and Cao, G. 2023. Communication-Efficient Federated Learning for Heterogeneous Edge Devices Based on Adaptive Gradient Quantization. In *Proc. of INFOCOM*.

Lubana, E. S.; and Dick, R. P. 2021. A Gradient Flow Framework For Analyzing Network Pruning. In *Proc. of ICLR*.

Ma, K.; Sun, Y.; Cui, J.; Li, D.; Guan, Z.; and Liu, J. 2023. Instance-Wise Batch Label Restoration via Gradients in Federated Learning. In *Proc. of ICLR*.

MacQueen, J. 1967. Some Methods for Classification and Analysis of Multivariate Observations. In *Proc. of BSMSP*.

McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; and y Arcas, B. A. 2017. Communication-Efficient Learning of

Deep Networks From Decentralized Data. In *Proc. of AISTATS*.

Molchanov, P.; Mallya, A.; Tyree, S.; Frosio, I.; and Kautz, J. 2019. Importance Estimation for Neural Network Pruning. In *Proc. of CVPR*.

Molchanov, P.; Tyree, S.; Karras, T.; Aila, T.; and Kautz, J. 2017. Pruning Convolutional Neural Networks for Resource Efficient Inference. In *Proc. of ICLR*.

Paillier, P. 1999. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *Proc. of EUROCRYPT*.

Queyrut, S.; Schiavoni, V.; and Felber, P. 2023. Mitigating Adversarial Attacks in Federated Learning with Trusted Execution Environments. In *Proc. of ICDCS*.

Roth, H. R.; Cheng, Y.; Wen, Y.; Yang, I.; Xu, Z.; Hsieh, Y.-T.; Kersten, K.; Harouni, A.; Zhao, C.; Lu, K.; et al. 2022. Nvidia Flare: Federated Learning From Simulation to Real-World. *arXiv preprint arXiv:2210.13291*.

Shahapure, K. R.; and Nicholas, C. 2020. Cluster quality analysis using silhouette score. In *Proc. of DSAA*.

Theis, L.; Korshunova, I.; Tejani, A.; and Huszár, F. 2018. Faster Gaze Prediction With Dense Networks and Fisher Pruning. *arXiv preprint arXiv:1801.05787*.

Xu, H.; Koehn, P.; and Murray, K. 2022. The Importance of Being Parameters: An Intra-Distillation Method for Serious Gains. In *Proc. of EMNLP*.

Xu, W.; Fan, H.; Li, K.; and Yang, K. 2021. Efficient Batch Homomorphic Encryption for Vertically Federated Xgboost. *arXiv preprint arXiv:2112.04261*.

Zhang, C.; Li, S.; Xia, J.; Wang, W.; Yan, F.; and Liu, Y. 2020. BatchCrypt: Efficient Homomorphic Encryption for Cross-Silo Federated Learning. In *Proc. of USENIX ATC*.

Zhang, S.; Li, Z.; Chen, Q.; Zheng, W.; Leng, J.; and Guo, M. 2021. Dubhe: Towards Data Unbiasedness With Homomorphic Encryption in Federated Learning Client Selection. In *Proc. of ICPP*.

Zhang, T.; Lam, K.-Y.; Zhao, J.; Li, F.; Han, H.; and Jamil, N. 2023. Enhancing federated learning with spectrum allocation optimization and device selection. *IEEE/ACM TON*, 31(5).

Zheng, Y.; Lai, S.; Liu, Y.; Yuan, X.; Yi, X.; and Wang, C. 2023. Aggregation Service for Federated Learning: An Efficient, Secure, and More Resilient Realization. *IEEE TDSC*, 20(2).

Zhou, R.; Yu, J.; Wang, R.; Li, B.; Jiang, J.; and Wu, L. 2023. A Reinforcement Learning Approach for Minimizing Job Completion Time in Clustered Federated Learning. In *Proc. of INFOCOM*.

Zhu, L.; Liu, Z.; and Han, S. 2019. Deep Leakage from Gradients. In *Proc. of NeurIPS*.

A Notations

Notation	Description
$\mathbf{W}$	Model parameters of a neural network
$\mathbf{W}_{-\mathbf{w}}$	Model parameters with $\mathbf{w}$ zeroed-out
$\mathcal{L}(\mathbf{W})$	Model loss function
$\nabla_{\mathbf{W}}\mathcal{L}(\mathbf{W})$	Gradients of the loss function with respect to $\mathbf{W}$
$\mathbf{\Gamma}(\mathbf{w})$	Sensitivity of a subset of model parameters $\mathbf{w}$
$\mathbf{\Gamma}_i$	Sensitivity vector of the i-th client
$I(\mathbf{W};\mathbf{W}_{-\mathbf{w}})$	The mutual information between $\mathbf{W}$ and $\mathbf{W}_{-\mathbf{w}}$
γ_k^i	The k-th element in $\mathbf{\Gamma}_i$
α_i	Encryption budget of the i-th client
r_i	Bandwidth of the i-th client
v_i	CPU clock speed of the i-th client
N^c	Number of clients in a cluster
$N^{\mathbf{w}}$	Total number of model parameters
N^p	Number of input pixels in a local batch of training data
N^b	Batch size
n	Number of pixels in a data sample
β_l	Weight of objective function l in the optimization problem
$\mathbf{X}_i$	Encryption mask vector of the i-th client
$\widehat{\mathbf{X}}$	Union of clients' encryption mask vectors in a cluster
G_{IID}	Set of clusters with IID data

Table 1: Table of main notations.

B Paillier Homomorphic Encryption

The Paillier cryptosystem (Paillier 1999) is a probabilistic asymmetric encryption scheme classified as a Partial Homomorphic Encryption (HE) system. Its security relies on the decisional composite residuosity assumption (DCRA), which posits the computational infeasibility of distinguishing random elements modulo n^2 from the n-th residues, where $n = pq$ is an RSA modulus with large primes p and q. Paillier satisfies the additive homomorphic property:

$$\text{Enc}(m_1) \cdot \text{Enc}(m_2) \equiv \text{Enc}(m_1 + m_2 \mod n) \mod n^2,$$

which makes it particularly suitable for Federated Learning scenarios where encrypted local model updates need to be aggregated without decryption.

C Straggler Problem of Existing Selective HE Methods in System Heterogeneity Scenarios

In existing Selective HE methods (Jin et al. 2023; Hu and Li 2024), all clients use the same encryption mask to encrypt model parameters. However, in scenarios with high system capability heterogeneity, the clients with the lowest capability may need substantially more time to encrypt the masked model parameters per training iteration, and always delay the completion of FL training (i.e., the *straggler problem* (Chai et al. 2020)). To confirm this conjecture, we vary the encryption budget of the clients, and measure their HE time costs (encryption plus decryption time).

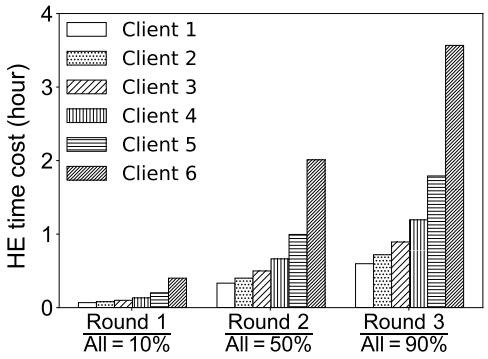


Figure 9: Imbalanced client HE time costs of existing Selective HE methods in system heterogeneity scenarios.

Specifically, we set up 6 clients to conduct the FL training of an AlexNet on the `CIFAR10` dataset. To emulate system heterogeneity, we allocate 6, 5, 4, 3, 2, 1 CPUs to the clients (denoted by *Client 1-6*), respectively. We conduct the FL training for 3 rounds, within which the model parameters are selectively encrypted by *MaskCrypt* (Hu and Li 2024) with the encryption budgets of 10%, 50% and 90%, respectively. Each round of FL training lasts for 3 iterations. Then, for each client, we measure its mean HE time cost over all the iterations per round, which is illustrated in Figure 9. We can see that when only 10% of the model parameters are encrypted, the clients' HE time costs are not significantly different. This means that the overhead caused by FL training is not the major cause of straggling. Along with the increase of the encryption budget, the imbalance of the clients' HE time costs deteriorates significantly. For example, when the encryption budget increases from 10% to 90%, the HE time cost of Client 6 increases by $>10\times$, while the HE time cost of Client 1 only increases by $<5\times$. Thus, Clients 1-5 have to wait for Client 6 per model aggregation, which significantly delays the completion of FL training. Therefore, we must maximally align the clients' HE time costs by adaptively tailoring the clients' encryption budgets according to their respective device capabilities.

D Performance Degradation of Existing Selective HE Methods in Statistical Heterogeneity Scenarios

To ensure the collaborative aggregation of encrypted model parameters, the existing Selective HE methods utilize the union of clients' selected model parameter subsets to represent the global selective encryption mask (Jin et al. 2023; Hu and Li 2024). However, in statistical heterogeneity scenarios where clients have Non-IID training data, the encryption mask may be much larger than any clients' encryption subsets since the clients' data distributions may have little overlap. In this case, the performance of HE overhead reduction of these methods may significantly degrade. To confirm this conjecture, we vary the degree of statistical heterogeneity in the clients' training data and measure the corresponding HE ratios of model parameters.

Specifically, to exclude the impact of system heterogeneity and data quantity difference, we set up 10 clients, each of

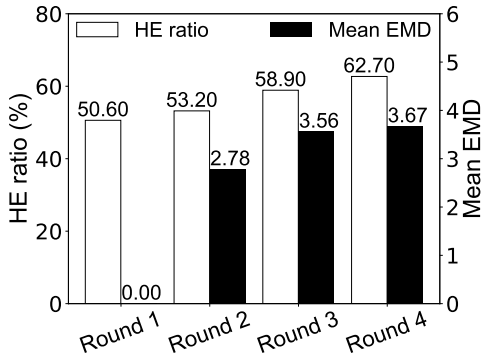


Figure 10: HE ratios of existing Selective HE methods under different degrees of statistical heterogeneity.

which has 10% data samples of `CIFAR10`, 1 CPU and a HE key size of 2048. We conduct the FL training for 4 rounds, with 3 iterations per round: 1) in the 1st round, each client has all the 10 classes of data samples (i.e., strictly IID); 2) in the 2nd and 3rd rounds, each client has 5 and 2 classes of data samples, respectively; 3) in the 4th round, each client only has 1 class of data samples (i.e., strictly Non-IID). To visualize the statistical heterogeneity, we calculate the mean Earth Mover's Distance (EMD) between the training data of every two clients in each round, which measures the similarity between two statistical distributions (Zhang et al. 2021). The larger mean EMD that the clients result in, the more statistically heterogeneous they are in a round. As illustrated with black bars in Figure 10, the measured mean EMDs are: 0 (Round 1), 2.78 (Round 2), 3.56 (Round 3) and 3.67 (Round 4), respectively. The white bars in Figure 10 illustrate the measured mean HE ratios of model parameters across varying degrees of statistical heterogeneity. We can see that as the mean EMD increases, the HE ratio also significantly rises. This indicates that the existing Selective HE methods become less cost-efficient in the presence of statistical heterogeneity. To effectively minimize HE overhead through selective encryption, it is better to determine the HE mask over clients with IID data. This motivates us to develop methods for measuring data distribution similarity, which can be utilized to cluster clients with IID data and determine cost-efficient HE masks for each cluster.

E Relation between Model Parameter Sensitivity and Data Distribution

We know that model parameter sensitivity is intrinsically the same as gradients, both of which represent the change of loss values. Since previous inversion attacks (Hitaj, Ateniese, and Pérez-Cruz 2017; Zhu, Liu, and Han 2019; Geiping et al. 2020) have demonstrated that the gradients transmitted during FL training can reveal clients' data distribution information, we can intuitively infer that the clients' model parameter sensitivity is also consistent with their data distribution. To confirm this, we continue to analyze the relationship between model parameter sensitivity and data distribution.

Specifically, we first set up 20 clients, each of which has 1 CPU and a HE key size of 2048. Then, we classify the clients into 5 categories with 4 clients per category, and ensure that every 2 classes of `CIFAR10` data samples are equally split to every 4 clients within the same category. Figure 11 shows the heat map matrix of the pairwise EMD between every two clients. The darker the color of the square corresponding to two clients, the more statistically heterogeneous they are. As expected, the clients are clearly split into 5 categories, within which the clients' data distributions are IID. Subsequently, we conduct the training of an AlexNet (Krizhevsky, Sutskever, and Hinton 2012) for one iteration on each client, and obtain the gradients $\nabla_{\mathbf{W}}\mathcal{L}(\mathbf{W})$ and model parameter values $\mathbf{w}$. Next, we calculate the Euclidean distance (Danielsson 1980) between the sensitivity vectors of every two clients as their similarity, which is also illustrated as a heat map matrix in Figure 12. We can see that although Figure 12 does not completely follow Figure 11, the pairwise similarities between the sensitivity vectors of the clients within the same category (i.e., antidiagonal of Figure 12) are highly consistent with those in Figure 11. This means that model parameter sensitivity can provide an alternative for data distribution representation and similarity measurement.

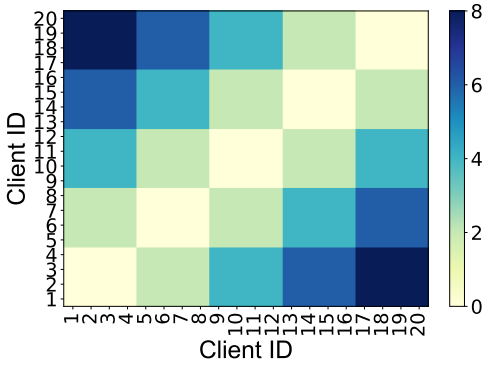


Figure 11: Heat map matrix of pairwise statistical heterogeneity in the training data of 20 clients.

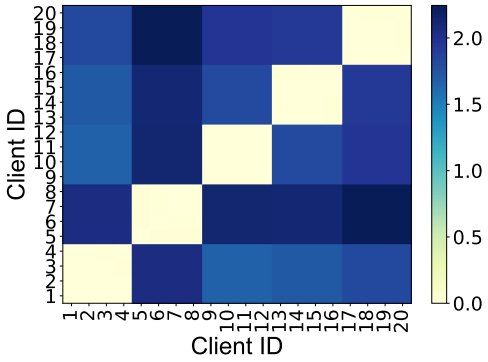


Figure 12: Heat map matrix of pairwise model parameter sensitivity similarity between the 20 clients.

F Selection of B and C

The values of B and C are scenario-dependent and must balance security, computational efficiency, and system heterogeneity. To guide their selection, we propose an empirical

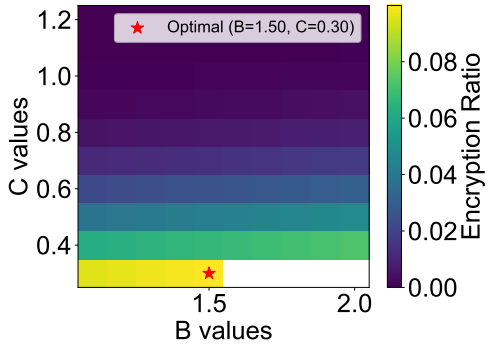


Figure 13: Encryption ratios under different B and C values.

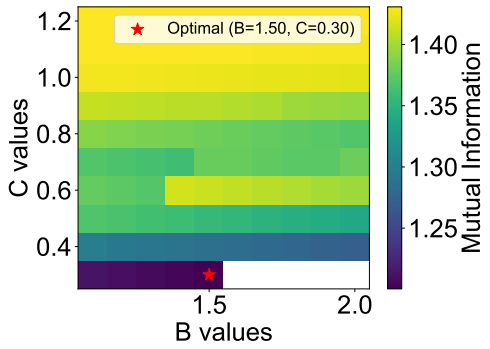


Figure 14: Mutual information under different B and C values.

methodology based on the following objectives:

(1) Ensure the model parameter selection optimization problem has a valid solution under system heterogeneity.

(2) Reduce Mutual Information (MI) between the original model parameters $\mathbf{W}$ and the ones after selective encryption $\mathbf{W}_{-\mathbf{w}}$, thereby limiting privacy leakage.

Figures 13 and 14 illustrate the heat map matrix of encryption ratio and MI $I(\mathbf{W}, \mathbf{W}_{-\mathbf{w}})$ under different combinations of B and C values for a resource-constrained client (encryption budget $\alpha_i = 0.1$) on the `CIFAR-10` dataset, respectively. We can see that with the increase of encryption ratio, MI generally decreases. The white squares in the heat map matrix represent the combinations of B and C values for which no feasible solutions can be obtained for the optimization problem.

Based on the empirical analysis results, we know that $(B = 1.5, C = 0.3)$ is the best combination that can achieve the maximum encryption ratio with the minimum MI. Note that the selection of B and C values should adapt to the actual statistical and system heterogeneity condition of participating clients, which may shift or fluctuate during FL. To improve the robustness of the encryption strategy, the selection of B and C values should tolerate certain variation margins (e.g., suboptimal combinations such as $(B = 1.4, C = 0.4)$).

Algorithm 2: Privacy-Preserving Clustering of Clients with DP-noised Sensitivity Vectors.

Input : Set of sensitivity vectors $\{\mathbf{\Gamma}_k\}$, noise scale σ, norm bound G
Output: Set of clusters $\{c_n\}$

1 Client:
2 Clip sensitivity vector: $\bar{\mathbf{\Gamma}}_k \leftarrow \mathbf{\Gamma}_k / \max(1, \|\mathbf{\Gamma}_k\|_2 / G)$;
3 Add DP noise: $\tilde{\mathbf{\Gamma}}_k \leftarrow \bar{\mathbf{\Gamma}}_k + \mathcal{N}(0, \sigma^2 G^2 \mathbf{I})$;
4 **return** $\tilde{\mathbf{\Gamma}}_k$
5 Server:
6 Initialize the set for storing sensitivity vectors $\{\tilde{\mathbf{\Gamma}}_k\}$;
7 **for** *the k-th client* **do**
8 $\{\tilde{\mathbf{\Gamma}}_k\} \leftarrow$ received DP-noised sensitivity vector $\tilde{\mathbf{\Gamma}}_k$;
9 $\{c_n\} \leftarrow \text{AffinityPropagation}(\{\tilde{\mathbf{\Gamma}}_k\})$;
10 **return** $\{c_n\}$

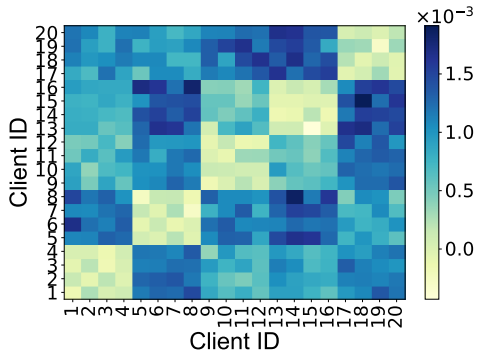
G Clustering with DP-noised Sensitivity Vectors

Although no existing attack methods can directly reconstruct user data from sensitivity vectors, the inherent privacy risks cannot be entirely neglected. To preemptively mitigate potential inference attacks, we adopt the noise injection methodology in DP-SGD (Abadi et al. 2016) for perturbing sensitivity vectors.

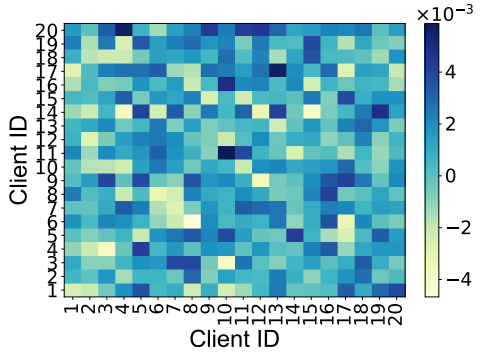
The noise magnitude σ is calibrated to satisfy (ϵ, δ)-differential privacy through the Gaussian mechanism $\sigma = \frac{\sqrt{2\ln(1.25/\delta)}}{\epsilon}$, where ϵ denotes the privacy budget. The lower the value of ϵ, the stronger privacy protection that the sensitivity value will enjoy, but the greater error it will suffer. δ represents the probabilistic relaxation parameter (fixed at 10^{-5}), and σ quantifies the standard deviation of the Gaussian noise injected into sensitivity vectors.

Algorithm 2 presents the details of our proposed method for privacy-preserving clustering of clients based on their DP-noised sensitivity vectors. The inputs include the set of clients' sensitivity vectors $\{\mathbf{\Gamma}_k\}$, the noise scale σ, and the norm bound G. At line 2, each client performs l_2-norm clipping on its sensitivity vector, with the clipping bound G. At line 3, each client adds the DP noise to its clipped sensitivity vector $\bar{\mathbf{\Gamma}}_k$. The noise follows the Gaussian distribution $\mathcal{N}(0, \sigma^2 G^2 \mathbf{I})$, with zero mean and covariance matrix $\sigma^2 G^2 \mathbf{I}$. At lines 6-9, the FL server receives all clients' DP-noised sensitivity vectors, and applies the Affinity Propagation (AP) method to cluster the clients into their respective groups c_n, within which the clients' data distribution is IID to each other.

Following the experimental setup in Section D, we further evaluate the effectiveness of client clustering based on DP-noised sensitivity vectors. As shown in Figure 15a, when $\epsilon = 0.1$, although most pairwise similarity values are greatly perturbed, the similarity between the clients with IID data is still significantly higher than the others. However, as shown



(a) Higher privacy budget ($\epsilon = 0.1$)



(b) Lower privacy budget ($\epsilon = 0.01$)

Figure 15: Heat map matrix of pairwise similarity between the DP-noised sensitivity vectors of the 20 clients.

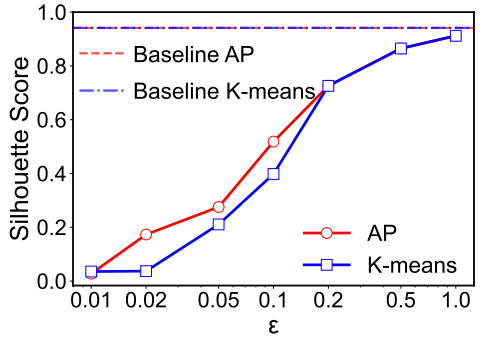


Figure 16: Impact of ϵ on Silhouette Score.

in Figure 15b, excessive noise injection over sensitivity values ($\epsilon = 0.01$) significantly hinders the effective measurement of data similarity.

Driven by these observations, we vary ϵ from 0.01 to 1.0 and apply the AP method and the K-means method to cluster the clients based on their DP-noised sensitivity vectors. We use the Silhouette Score (Shahapure and Nicholas 2020; Fan et al. 2024) and the Adjusted Rand Index (ARI) (Zhang et al. 2023) to evaluate clustering quality and clustering accuracy, respectively. Specifically, the Silhouette Score (S) is an intrinsic metric for evaluating cluster quality based on

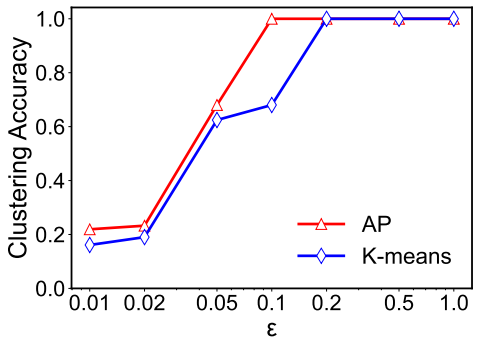


Figure 17: Impact of ϵ on Clustering Accuracy.

cohesion and separation. It is computed as:

$$S(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}, \tag{10}$$

where $a(i)$ is the mean distance between sample i and all other data points in the same cluster, and $b(i)$ is the mean distance between sample i and all the data points in its nearest cluster. $S(i)$ ranges from -1 to 1, with higher scores indicating better clustering quality.

ARI reflects clustering accuracy by comparing the DP-noised clustering result with the baseline (i.e., ground truth clusters). It is computed as:

$$\text{ARI}(C_n^p, C_n^g) = \frac{2(\beta_{00}\beta_{11} - \beta_{01}\beta_{10})}{(\beta_{00} + \beta_{01})(\beta_{01} + \beta_{11}) + (\beta_{00} + \beta_{10})(\beta_{10} + \beta_{11})}, \tag{11}$$

where C_n^p is the predicted clustering result, C_n^g is the ground truth clusters, β_{11} is the number of pairs that are in the same cluster in both C_n^p and C_n^g, β_{00} is the number of pairs that are in different clusters in both C_n^p and C_n^g, β_{01} is the number of pairs that are in the same cluster in C_n^p but in different clusters in C_n^g, and β_{10} is the number of pairs that are in different clusters in C_n^p but in the same cluster in C_n^g. The ARI will be close to 0 if two clusters do not have any overlapped pair of data samples and exactly 1 if the clusters are the same.

Figures 16 and 17 illustrate the measured Silhouette Scores and accuracy of the clustering results under the two methods, respectively. *Baseline AP* and *Baseline K-means* represent the cases without DP-noise injection, *AP* and *K-means* represent the cases on DP-noised sensitivity vectors. We can see that as the Silhouette Score is dependent on the distances between data samples, it is sensitive to subtle perturbation caused by DP noise. In comparison, the clustering accuracy of both methods suffers no drop until ϵ decreases below 0.2. This is mainly because that sensitivity vectors, as long as not excessively perturbed by noise, can effectively capture the underlying data distribution patterns, which validates the effectiveness of exploiting DP-noised sensitivity vectors for high-quality clustering of clients while preserving privacy. What's more, we also notice that compared with the K-means method, which requires predefined number of clusters in advance, the AP method can automatically determine the number of clusters, and even tolerate the noise level at $\epsilon = 0.1$. This indicates that the AP method is more robust to DP noise perturbation than the K-means method

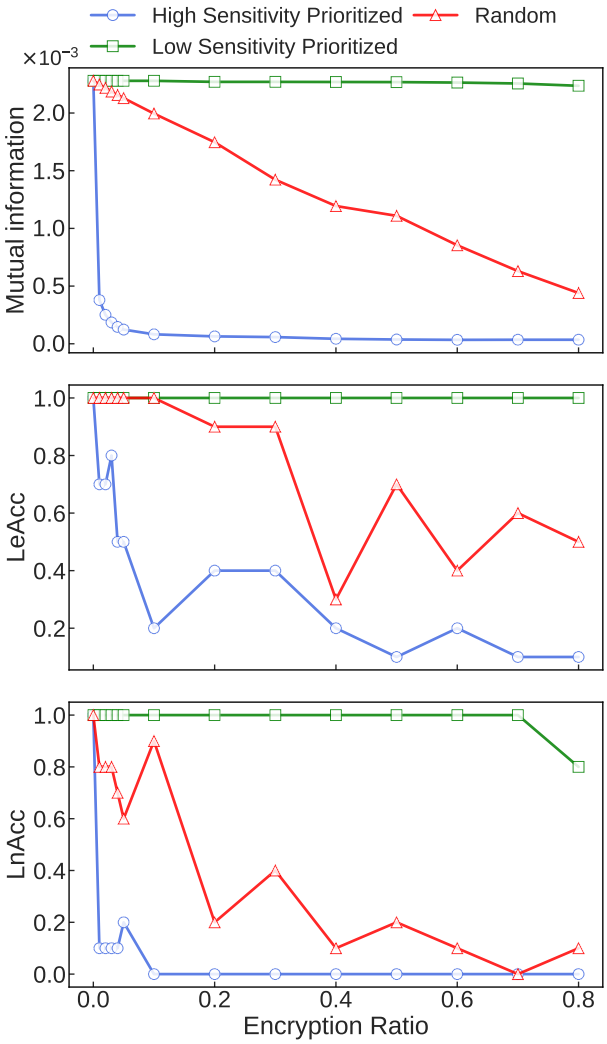


Figure 18: Impact of Encryption Ratio and Strategy on Mutual Information and Attack Success Rate.

and more effective for client clustering in scenarios where the server cannot predetermine the number of clusters.

H Effectiveness of Sensitivity-guided Selective HE

To validate the effectiveness of the sensitivity-guided selective HE strategy, we make a detailed analysis of MI and attack success rate under varying encryption ratios using three different encryption strategies:

- **High Sensitivity Prioritized.** Given a specified encryption ratio, we prioritize the selection of model parameters with the highest sensitivity for encryption.
- **Low Sensitivity Prioritized.** Given a specified encryption ratio, we prioritize the selection of model parameters with the lowest sensitivity for encryption.
- **Random.** Given a specified encryption ratio, we randomly select model parameters for encryption.

Specifically, we launch the iLRG attack under varying encryption ratios ranging from 0 to 0.8, and measure the changes of MI, Label existence Accuracy (LeAcc) and Label number Accuracy (LnAcc). The experiments are conducted on ResNet-50 with a batch size of 30, and the `MNIST` dataset. This set of experimental settings is vulnerable to iLRG attacks on plaintext, thereby excluding the influence of settings other than the encryption strategies.

Figure 18 illustrates the measured results of each strategy. We can see that along with the increasing of encryption ratio, the MI of **Random** almost decreases linearly. While for the case of **Low Sensitivity Prioritized**, the MI almost remains unchanged, and for the case of **High Sensitivity Prioritized**, the MI sharply drops to almost 0 when the encryption ratio is above 0.1. Correspondingly, LeAcc and LnAcc follow similar change patterns. These results confirm the reliability of employing MI to quantify the privacy leakage caused by the selective encryption of model parameters. What's more, we can also notice that **Low Sensitivity Prioritized** and **High Sensitivity Prioritized** demonstrate stark contrast in privacy protection in terms of LeAcc and LnAcc. Although **Random** can finally reach a high privacy protection level, it consumes significantly more encryption overhead than **High Sensitivity Prioritized**.

In summary, these observations strongly advocate the effectiveness and necessity of applying sensitivity-guided selective HE strategy, which validates the theoretical foundation of our proposed framework that adaptively balances security and HE overhead for each client in cross-device FL.

I HE Key Management and Distribution

In the standard threat model, the server is honest-but-curious, and clients are honest. However, a more realistic threat model considers the presence of malicious clients who might collude with the server or other clients. If all clients share a single private key, a malicious client could intercept and decrypt the model updates from any other client, completely compromising their privacy.

To address this expanded threat, we propose a framework that decouples the aggregation server from a trusted **Decryption and Key Management Server (DKMS)**. In our framework, the DKMS assumes the critical role of generating and distributing public/private key pairs, as well as decrypting the final aggregated model. This design does not introduce additional physical infrastructure; instead, it formally delegates key management and decryption operations to a trusted third party - a function that is frequently implied but rarely explicitly defined in most homomorphic encryption-based federated learning systems. Maintaining the private key either at the aggregation server or with participating clients would compromise security, as possession of the private key would enable decryption of all sensitive communications. The modified complete algorithm pseudocode can be referred to in Algorithm 3.

This decoupled architecture offers two significant advantages:

1. **Enhanced Security:** By isolating the private key on a dedicated DKMS, we prevent the aggregation server and any potentially malicious clients from decrypting individual client updates. The aggregation server only per-

forms homomorphic additions on ciphertexts, and clients only receive the final, decrypted global model from the DKMS.

2. **Improved Efficiency in Heterogeneous Environments:** In our system, clients only perform encryption. The final aggregated model is decrypted once by the DKMS and then distributed to all clients. This eliminates the need for each client to perform decryption locally, which can be a significant bottleneck, especially for resource-constrained devices. Since the final aggregated model requires a union of all clients' encryption masks for decryption, the decryption time would be dictated by the slowest device, exacerbating the straggler problem. Centralizing decryption at the DKMS effectively mitigates this issue.

J Proof of Collusion Resistance

Theorem 1: If the aggregation server is honest-but-curious, and the server colludes with at most $|G_{IID}|-2$ clients within a cluster G_{IID}, *SenseCrypt* achieves model confidentiality for the honest clients in that cluster.

Proof: Let C and H denote the set of corrupted and honest clients respectively, within a specific cluster G_{IID}, where $|C|+|H| = |G_{IID}|$. Since the homomorphic encryption is IND-CPA secure, adversaries cannot infer an honest client's plaintext model $\mathbf{W}_i$ directly from its encrypted version $\mathbf{W}_i^*$.

However, adversaries can access the final decrypted aggregation result $\mathbf{W}^{e+1}$ for their cluster. Following the aggregation protocol, the server and the colluding clients in C can compute the aggregated model of the honest clients by subtracting their own contributions:

$$\sum_{i\in H} p_i \mathbf{W}_i = \mathbf{W}^{e+1} - \sum_{j\in C} p_j \mathbf{W}_j$$

If there is only one honest client (i.e., $|H| = 1$), the adversary can recover the full model update $\mathbf{W}_i$ of that client. However, as long as there are at least two honest clients (i.e., $|H| \geq 2$), adversaries can only obtain the aggregated sum of their models, but cannot disentangle this sum to recover the individual model update $\mathbf{W}_i$ of any specific honest client.

Therefore, given the server colludes with at most $|G_{IID}| - 2$ clients, ensuring $|H| \geq 2$, *SenseCrypt* protects the model confidentiality of individual honest clients within the cluster.

K Choice of Encryption Method: Paillier vs. CKKS

While both Paillier and schemes like CKKS (Cheon-Kim-Kim-Song) are popular homomorphic encryption solutions, Paillier is better suited for the *SenseCrypt* framework. The primary reason lies in the incompatibility of our selective encryption strategy with the batching (or packing) technique that makes CKKS highly efficient.

In *SenseCrypt*, the decision to encrypt a parameter is based on its individual sensitivity score. This means that sensitive and non-sensitive parameters can be adjacent to one

Algorithm 3: Workflow of *SenseCrypt* with Dual-Server Architecture.

Input : Set of clients $\{1, \ldots, K\}$, parameters B, C, η_{MI};
Output: Converged global model $\mathbf{W}^{\text{final}}$;

Phase 1: Initialization;
DKMS():
  Generate HE key pair: (pk, sk) ← Paillier.keygen();
  Distribute pk to all clients;
  for *each client* $i = 1, \ldots, K$ **in parallel do**
    Client(i):
      Local training to get initial $\mathbf{W}_i^1$ and $\nabla_{\mathbf{W}}\mathcal{L}(\mathbf{W}_i^1)$;
      Calculate sensitivity vector: $\mathbf{\Gamma}_i \leftarrow |(\mathbf{W}_i^1)^\top \nabla_{\mathbf{W}}\mathcal{L}(\mathbf{W}_i^1)|$;
      Send $\mathbf{\Gamma}_i$ and (r_i, v_i) to AggregationServer();
  AggregationServer():
    Receive $\mathbf{\Gamma}_i$ and (r_i, v_i) from all clients;
    for *each client* $i = 1, \ldots, K$ **do**
      Calculate encryption budget:

$$\alpha_i \leftarrow \frac{\min\{\overline{r}_i, \overline{v}_i\}}{\max\{|\min\{\overline{r}_j, \overline{v}_j\}|\}_{j=1}^K}$$

      Send α_i back to client i;
    Cluster clients based on sensitivity similarity:
    $\{G_{\text{IID}}\} \leftarrow \text{AffinityPropagation}(\{\mathbf{\Gamma}_1, \ldots, \mathbf{\Gamma}_K\})$;

Phase 2: Federated Training Loop;
for *each iteration* $e = 1, 2, \ldots$ **do**
  for *each group* $G_{\text{IID}} \in \{G_{IID}\}$ **in parallel do**
    for *each client* $i \in G_{\text{IID}}$ **in parallel do**
      Client(i):
        Receive global model $\mathbf{W}^e$;
        Update local model: $\mathbf{W}_i^e \leftarrow$ local training on $\mathbf{W}^e$;
        Update sensitivity vector: $\mathbf{\Gamma}_i \leftarrow |(\mathbf{W}_i^e)^\top \nabla_{\mathbf{W}}\mathcal{L}(\mathbf{W}_i^e)|$;
        Solve for optimal mask $\mathbf{X}_i = [x_k^i] \in \{0,1\}^{N^{\mathbf{W}}}$ by:
        $\min \quad [\sum_{k=1}^{N^{\mathbf{W}}} x_k^i, -\sum_{k=1}^{N^{\mathbf{W}}} x_k^i \gamma_k^i]^\top$
        s.t.:
        1. $\sum_{k=1}^{N^{\mathbf{W}}} x_k^i \leqslant \lfloor \alpha_i N^{\mathbf{W}} \rfloor$ *(Budget Constraint)*
        2. $\frac{\sum x_k^i \gamma_k^i}{\sum \gamma_k^i} \geqslant 1 - Ce^{-B\alpha_i}$ *(Security Constraint)*
        3. $I(\mathbf{W}; (1 - \mathbf{X}_i) \odot \mathbf{W}) \leq \eta_{\text{MI}}$ *(MI Constraint)*
        Selectively encrypt model:

$$\mathbf{W}_i^{e,*} \leftarrow \text{Encrypt}(\mathbf{W}_i^e, \mathbf{X}_i, \text{pk})$$

        Send $\mathbf{W}_i^{e,*}$ and $\mathbf{X}_i$ to AggregationServer();
      AggregationServer():
        Receive $\mathbf{W}_i^{e,*}$ and $\mathbf{X}_i$ from all clients in G_{IID};
        Aggregate encrypted models:

$$\mathbf{W}^{e+1,*} \leftarrow \sum_{i\in G_{\text{IID}}} \frac{n_i}{N_G} \mathbf{W}_i^{e,*}$$

        Create union mask for decryption: $\widehat{\mathbf{X}} \leftarrow \bigcup_{i\in G_{\text{IID}}} \mathbf{X}_i$;
        Send $\mathbf{W}^{e+1,*}$ and $\widehat{\mathbf{X}}$ to DKMS();
      DKMS():
        Receive $\mathbf{W}^{e+1,*}$ and $\widehat{\mathbf{X}}$ from AggregationServer();
        Decrypt aggregated model:

$$\mathbf{W}^{e+1} \leftarrow \text{Decrypt}(\mathbf{W}^{e+1,*}, \widehat{\mathbf{X}}, \text{sk})$$

        Distribute plaintext global model $\mathbf{W}^{e+1}$ to all clients;

Quantity	Scheme	Encryption Time (s)	Ciphertext Size (bytes)
1	Paillier	0.3748	768
	CKKS	0.0073	333324 × 1
10	Paillier	3.7613	7680
	CKKS	0.0076	333570 × 10
100	Paillier	40.7887	76800
	CKKS	0.0120	334488 × 100
1000	Paillier	396.1321	767990
	CKKS	0.0079	334366 × 1000
10000	Paillier	3925.9769	7679907
	CKKS	0.0261	1002800 × 10000

Table 2: Comparison of Encryption Performance between Paillier and CKKS Schemes.

another in the model's structure. Furthermore, each client generates a unique encryption mask tailored to its data and system capabilities.

CKKS achieves its efficiency by packing multiple plaintext values (a vector) into a single ciphertext and performing SIMD (Single Instruction, Multiple Data) operations. This requires the data to be arranged in a specific order before encryption. In our case, to use batching, clients would need to reorder their model parameters, separating the ones to be encrypted from the ones to be sent in plaintext. Since each client has a different mask, their reordering would be different, and the server would be unable to perform meaningful aggregation because the parameter positions would not align.

Therefore, we cannot leverage the batching capabilities of CKKS. As shown in Table 2, when encrypting a single parameter without batching, Paillier is far more practical. Although CKKS is faster for the encryption operation itself, its resulting ciphertext is designed to hold a large vector and is therefore massive, regardless of the number of encrypted values. Specifically, encrypting a single parameter with CKKS would generate a ciphertext of 333324 bytes, which is over **434 times** larger than Paillier's 768 byte ciphertext. This dramatic increase in data size would lead to an untenable communication overhead. Given our parameter-wise selective strategy, Paillier offers a more efficient solution by minimizing this overhead.

L Time Consumption Analysis of Components

To understand the computational overhead of each component in *SenseCrypt*, we profiled the execution time for a single client during one training iteration on the `CIFAR10` dataset. The results are summarized in Table 3.

As the table shows, the most time-intensive operations on the client side are encryption and solving the optimization problem. Local model training and decryption also contribute significantly to the overall latency. In contrast, the initial one-off cost of clustering on the server is negligible, and the sensitivity calculation on the client side is very fast. The server-side aggregation time is comparable to the client-side encryption time. These results underscore the importance of our approach to adaptively manage the encryption overhead to mitigate the straggler problem.

Step	Min (s)	Max (s)	Avg (s)
Local training	7.1	7.7	7.4
Sensitivity calculation	0.03	0.15	0.1
Optimization problem solving	11.8	17.7	13.2
Encryption	79.7	103.8	87.3
Decryption	24.8	86.4	56.8
FedAvg aggregation (server-side)	-	-	87.9
Clustering (server-side, one-off)	-	-	0.002

Table 3: Average time consumption per iteration for a single client.

M Approximation of Parameter Sensitivity

Our use of a first-order Taylor expansion is a well-established and reasonable method for approximating parameter importance. This technique is standard in machine learning, notably in classic network pruning algorithms. Here is the proof of the approximation expansion. The sensitivity of a parameter subset $\mathbf{w}$ is defined as the change in loss when $\mathbf{w}$ is zeroed-out: $\mathbf{\Gamma}(\mathbf{w}) = |\mathcal{L}(\mathbf{W}) - \mathcal{L}(\mathbf{W}_{-\mathbf{w}})|$. We can approximate the term $\mathcal{L}(\mathbf{W}_{-\mathbf{w}})$ by performing a Taylor series expansion of the loss function $\mathcal{L}$ around the point $\mathbf{W}$:

$$\begin{aligned}\mathcal{L}(\mathbf{W}_{-\mathbf{w}}) = \mathcal{L}(\mathbf{W}) &+ (\mathbf{W}_{-\mathbf{w}} - \mathbf{W})^\top \nabla_{\mathbf{W}} \mathcal{L}(\mathbf{W}) \\ &+ \frac{1}{2}\big((\mathbf{W}_{-\mathbf{w}} - \mathbf{W})^2\big)^\top \mathbf{H} (\mathbf{W}_{-\mathbf{w}} - \mathbf{W}) \\ &+ \cdots\end{aligned}$$

where $\mathbf{H}$ is the Hessian matrix of $\mathcal{L}$ evaluated at $\mathbf{W}$, and the ellipsis represents higher-order terms.

The difference vector $(\mathbf{W}_{-\mathbf{w}} - \mathbf{W})$ is a vector that is equal to $-\mathbf{w}$ at the positions corresponding to the parameters in the subset $\mathbf{w}$, and is zero everywhere else. The first-order approximation is made by truncating the series after the linear term, assuming that the quadratic and higher-order terms are negligible. This is a common assumption when the change (in this case, zeroing out $\mathbf{w}$) is relatively small. This truncation leads to the approximation:

$$\begin{aligned}\mathcal{L}(\mathbf{W}_{-\mathbf{w}}) - \mathcal{L}(\mathbf{W}) &\approx (\mathbf{W}_{-\mathbf{w}} - \mathbf{W})^\top \nabla_{\mathbf{W}} \mathcal{L}(\mathbf{W}) \\ &= -\mathbf{w}^\top \nabla_{\mathbf{W}} \mathcal{L}(\mathbf{W})\end{aligned}$$

Rearranging the terms, we get the change in loss:

$$\mathcal{L}(\mathbf{W}) - \mathcal{L}(\mathbf{W}_{-\mathbf{w}}) \approx \mathbf{w}^\top \nabla_{\mathbf{W}} \mathcal{L}(\mathbf{W})$$

Substituting this approximation back into the sensitivity definition, we obtain the final, computationally tractable formula:

$$\mathbf{\Gamma}(\mathbf{w}) \approx |\mathbf{w}^\top \nabla_{\mathbf{W}} \mathcal{L}(\mathbf{W})|$$

N Related Work

Statistical & system heterogeneity in FL. Cho *et al.* (Cho, Mathur, and Kawsar 2022) proposed to personalize model training for countering statistical heterogeneity in cross-device FL. Liu *et al.* (Liu, Guo, and Chen 2021) proposed to utilize the sparsity of CNNs for privacy-preserving similarity measurement of clients' data. To mitigate system heterogeneity issues, Bonawitz *et al.* (Bonawitz et al. 2019) designed a FL system that filters clients by their response speed in FL training. Chai *et al.* (Chai et al. 2020) proposed

a tier-based FL system that selects clients by training performance to mitigate system heterogeneity problems. Zhou *et al.* (Zhou et al. 2023) proposed to cluster clients by device capabilities, and utilize reinforcement learning to mitigate statistical heterogeneity issues. Jiang *et al.* (Jiang et al. 2023) proposed to utilize adaptive client selection and gradient compression for addressing the straggler problem. However, these methods mostly rely on additional components to measure data similarity or coordinate FL training, which creates extra burden for FL framework adaptation.

Overhead-optimized HE for FL. To make HE more lightweight, Jiang *et al.* (Jiang, Wang, and Liu 2021) proposed to drop the asymmetric-key design to meet the minimum requirements of security and functionality. Hao *et al.* (Hao et al. 2019) proposed to combine lightweight symmetric additive HE with DP technique to improve security protection. However, such methods generally suffer from degraded security since symmetric encryption cannot provide the same level of security as public-key encryption.

Another alternative is to encrypt multiple model parameters in one go. Zhang *et al.* (Zhang et al. 2020) proposed *BatchCrypt*, which utilizes batch quantization and encryption of model parameters to reduce HE overhead. Chen *et al.* (Chen et al. 2021) further migrated *BatchCrypt* to the establishment of gradient boosting decision trees for vertical FL. Xu *et al.* (Xu et al. 2021) indicated that the gradient quantization in *BatchCrypt* may cause frequent overflow errors due to the addition of negative numbers represented in two's complement, and result in accuracy loss. To fix this issue, Zheng *et al.* (Zheng et al. 2023) modified *BatchCrypt* via down-scaling and identification of negative model parameter values. Han *et al.* (Han and Yan 2023) proposed an adaptive accuracy-lossless batch HE method that shifts parameters to non-negative values for the prevention of overflow errors. Nevertheless, these methods cannot easily fit in mainstream FL frameworks without specific adaptation.

Inspired by the model pruning methods that remove redundant model parameters by sensitivity (Molchanov et al. 2017; Theis et al. 2018; Ding et al. 2019; Molchanov et al. 2019; Lubana and Dick 2021), several methods proposed to selectively encrypt relatively more sensitive model parameters to reduce HE overhead. For example, Jin *et al.* (Jin et al. 2023) designed *FedML-HE*, which utilizes the union of clients' sensitive model parameter masks for encryption. Hu *et al.* (Hu and Li 2024) proposed *MaskCrypt*, which lets the clients agree on a consensus mask for selective HE with reduced overhead. However, these methods cannot adaptively ensure the security of each client with low HE overhead in cross-device FL.

O Comparison with FedML-HE

Although FedML-HE (Jin et al. 2023) proposes a selective encryption mechanism based on parameter sensitivity, its efficiency is extremely low. FedML-HE calculates sensitivity via sequentially iterating the gradient value of each parameter and computing its partial derivative with respect to each data label, as formulated in

$$J_m(y_k) = \frac{\partial}{\partial y_k}\left(\frac{\partial \ell(\mathbf{X}, \mathbf{y}, \mathbf{W})}{\partial w_m}\right),$$

where $\ell(\mathbf{X}, \mathbf{y}, \mathbf{W})$ denotes the loss function. Specifically, FedML-HE requires $\mathcal{O}(N^w \cdot N^b)$ complex backpropagation operations, where N^w is the total number of model parameters and N^b is the batch size. In contrast, SenseCrypt only requires $\mathcal{O}(N^w)$ simple multiplication operations, which are significantly less time-consuming than backpropagation.

Consequently, while FedML-HE required **more than 4 hours** to compute the sensitivity of 50,000 parameters on a 5-thread CPU, SenseCrypt and MaskCrypt completed it in **less than 0.1 seconds**. Even with an extreme space-for-time trade-off optimization, FedML-HE's computation time could only be reduced to 340 seconds, at the cost of consuming over **100 GB** of memory. Table 4 summarizes the sensitivity computation time for a single evaluation over 200 samples on different models. Therefore, FedML-HE is unsuitable for selective encryption in heterogeneous device scenarios due to its prohibitive computational overhead.

Model	Params	Dataset	Time (s)
MLP	50,890	MNIST	17214.75
AlexNet	124,534	CIFAR-10	42823.79

Table 4: Sensitivity Computation Time of FedML-HE

P Data Reconstruction Example

Figure 19 illustrates a `MNIST` data sample reconstructed from batch-averaged gradients via iLRG from Round 4 in Fig. 8 of the main text. Compared with *MaskCrypt*, the reconstructed data in *SenseCrypt* exposes much less useful information to attackers.

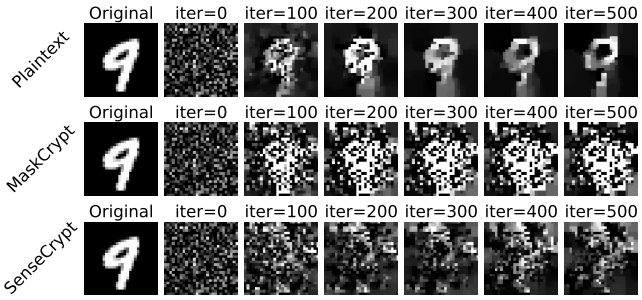


Figure 19: Comparison of data reconstruction results.