EncCluster: Bringing Functional Encryption in Federated Foundational Models

Vasileios Tsouvalas[1]* **Samaneh Mohammadi**[2,3]* **Ali Balador**[3] **Tanir Ozcelebi**[1]

Francesco Flammini[3] **Nirvana Meratnia**[1]

[1]Eindhoven University of Technology, Eindhoven, Netherlands
[2]RISE Research Institutes of Sweden, Västerås, Sweden
[3]Mälardalen University, Västerås, Sweden

Abstract

Federated Learning (FL) decentralizes model training by transmitting local model updates to a central server, yet it remains vulnerable to inference attacks during these transmissions. Existing solutions, such as Differential Privacy (DP) and Functional Encryption (FE), often degrade performance or impose significant operational burdens on clients. Meanwhile, the advent of Foundation Models (FMs) has transformed FL with their adaptability and high performance across diverse tasks. However, delivering strong privacy guarantees with these highly parameterized FMs in FL using existing privacy-preserving frameworks amplifies existing challenges and further complicates the efficiency-privacy trade-off. We present `EncCluster`†, a novel method that integrates model compression through weight clustering with decentralized FE and privacy-enhancing data encoding using probabilistic filters to deliver strong privacy guarantees in FL without affecting model performance or adding unnecessary burdens to clients. We perform a comprehensive evaluation, spanning 4 datasets and 5 architectures, to demonstrate `EncCluster` scalability across encryption levels. Our findings reveal that `EncCluster` significantly reduces communication costs — below even conventional *FedAvg* — and accelerates encryption up to 1000× over baselines; at the same time, it maintains high model accuracy and enhanced privacy assurances.

1 Introduction

Federated Learning (FL) addresses the challenges of data privacy, regulatory compliance, and centralized data processing by enabling collaborative AI model training directly on edge devices (1). Unlike traditional approaches that aggregate data on centralized servers, FL keeps raw data on devices, reducing privacy risks and helping organizations comply with regulations like GDPR and the AI Act. FL operates through multiple rounds where model updates are sent to clients, trained locally, and then aggregated on a server (e.g., *FedAvg* (2)) until the model converges.

Although FL preserves privacy by keeping raw data on devices, it remains vulnerable to inference attacks during model update communications, potentially exposing sensitive information (3; 4). To address this, privacy-preserving methods such as Differential Privacy (DP) (5) and Secure Multi-Party Computation (SMPC) (6) have been proposed. However, DP often degrades model performance (7),

*Equal contribution. Corresponding author: `samaneh.mohammadi@ri.se`
†Code: `https://github.com/FederatedML/EncCluster`

while SMPC faces scalability challenges due to high computational demands (8). Encryption-based techniques, including Homomorphic Encryption (HE)(9) and Functional Encryption (FE) (10), offer strong privacy without sacrificing performance but depend on fully-trusted Third-Party Authorities (TPAs) for key management, which can be impractical during FL training due to the decentralized nature of FL and the risks associated with relying on fully trusted entities (6; 11). Additionally, these methods involve cryptographic operations across all model parameters, leading to significant increases in computational costs and training times as models grow in complexity.

The need for scalable privacy-preserving solutions becomes increasingly urgent with the rise of Foundation Models (FMs) in FL. These models, exemplified by ViT architectures that scale to billions of parameters (12), have shown exceptional capabilities across various tasks. However, as FMs are more widely adopted in FL (13; 14; 15), the existing trade-offs between privacy and efficiency become even more pronounced. Current encryption-based methods struggle to meet the computational demands of these large-scale models, revealing a critical gap in the literature: the absence of scalable, efficient privacy-preserving techniques that can accommodate the growing complexity and size of FMs within the FL paradigm.

To bridge the gap between the high-performance potential of FMs and the practical constraints of privacy-preserving FL schemes, we introduce `EncCluster`, a framework that offer robust privacy protection against inference attacks while requiring minimal communication and computation overhead for clients participating in FL. To achieve this, we design our framework with the following three building blocks, i.e. (i) model compression via weight clustering, (ii) decentralized FE, allowing cryptographic encryption without fully-trusted *TPA*s, and (iii) encoding via probabilistic data structure (i.e., Binary Fuse (BF) filters) to further enhance privacy without introducing excessive computational burdens. Specifically, we apply weight clustering locally on clients' models and subsequently encrypt the resulting set of cluster centroids via FE. Cluster-weight mapping, which signifies associations between positions in the weight matrix and respective centroids, is then injected into BF filters through computationally efficient hashing operations. To fuse all model updates, the server reconstructs this mapping via a membership query in the BF filters and performs a secure aggregation without decrypting the clients' model updates. In doing so, `EncCluster` restricts the computationally "*heavy*" encryption operations to a small set of centroid values —irrespective of the model's parameters — while mappings to model weights are encoded through cost-effective hashing, striking a balance between privacy preservation and the practical computational and communication demands of FMs in FL. Concisely, our contributions are as follows:

- We introduce `EncCluster`, a *lightweight* privacy-preserving FL framework that maintains model performance with low operational costs, enabling FMs in FL where strict privacy is key.
- We combine weight clustering with decentralized FE and BF filter-based encoding for secure, efficient transmission, and aggregation of compressed model updates, eliminating reliance on trusted external entities.
- Our comprehensive evaluation across 4 datasets and 5 neural architectures demonstrates `EncCluster`'s efficiency gains over traditional FE schemes, achieving 13× reduction in communication costs and 1000-fold speedup in computational demand, alongside a mere 1.15% accuracy loss compared to *FedAvg*.
- We showcase `EncCluster` scalability across various encryption levels revealing *near-constant* communication costs in FL with minimal increases to clients' encryption times, all while maintaining robust privacy guarantees.

2 Preliminaries

Weight Clustering. Weight clustering compresses neural networks by grouping similar weights into clusters using algorithms like K-means (16). This can be done per layer or across the entire model. Given a neural network f with weights $\theta = (\theta_1, \dots, \theta_d) \in \mathbb{R}^d$, the goal is to form κ clusters $C = \{c_1, \dots, c_\kappa\}$ by minimizing:

$$\mathcal{L}_{wc}(\theta, \mathcal{Z}) = \sum_{j=1}^{\kappa} \sum_{i=1}^{d} u_{ij} \cdot ||\theta_i - z_j||^2 , \tag{1}$$

where $\mathcal{Z} = \{z_1, \ldots, z_\kappa\}$ represents the κ centroids, and u_{ij} is a binary indicator for the assignment of weights to clusters. Minimizing $\mathcal{L}_{wc}$ yields the centroids $\mathcal{Z}$ and cluster mappings, assigning each weight to its nearest centroid.

Probabilistic Filters. Probabilistic filters are data structures that map a universe of keys $\mathcal{U}$ to fixed-size bit values, compacting data using hash functions to create a uniformly distributed array of fingerprints $\mathcal{H}$. These structures allow efficient membership checking with adjustable false positive rates while ensuring zero false negatives.

Among various filters, *Binary Fuse* (BF) filters (17) stand out for their space efficiency (up to 1.08 bits per entry) and low false positive rates (2^{-bpe}). A μ-wise BF filter uses μ distinct hash functions $h_j \colon \{1, \ldots, 2^n\} \to \{1, \ldots, t\}$, where t is the size of the fingerprint array $\mathcal{H}$. Let $g \colon \mathbb{N} \to \{1, \ldots, 2^n\}$ generate ξ-bit fingerprints. For each key $i \in \mathcal{U}$, the fingerprint array $\mathcal{H}$ is computed as:

$$\mathcal{H} = \bigcup_{i \in \mathcal{U}} \phi(i) = \bigcup_{i \in \mathcal{U}} \left(\bigcup_{j=1}^{\mu} \{h_j(g(i))\} \right) \tag{2}$$

Here, $\phi(i)$ computes the set of μ locations in $\mathcal{H}$ for each key i in $\mathcal{U}$. Once $\mathcal{H}$ is constructed, we can perform a membership check as:

$$\text{Member}(x) = \begin{cases} \text{true}, & \bigoplus_{j=1}^{m} \mathcal{H}\left[h_j(g(x))\right] = g(x) \\ \text{false}, & \text{otherwise} \end{cases} \tag{3}$$

where $\bigoplus_{j=1}^{m} \mathcal{H}[\cdot]$ represents a bitwise *XOR* over the hash-indexed values in $\mathcal{H}$. If the result matches the fingerprint $g(x)$, the filter likely identifies x as a member. BF filters use efficient hash functions like MurmurHash3 (18), ensuring low computational cost and high-quality hash distributions.

Decentralized Functional Encryption. Functional encryption (FE) enables operations on encrypted data to yield plaintext results without decrypting individual inputs (19). Compared to traditional HE, FE is more efficient, especially in secure multi-party aggregation (10). We focus on Decentralized Multi-Client Functional Encryption (DMCFE) (20), which supports inner product computations on encrypted data. DMCFE is notable for (i) allowing participants to hold partial keys, eliminating the need for a trusted *TPA*, and (ii) incorporating a labeling mechanism to bind functional keys to specific ciphertexts, ensuring exclusivity.

Let $\mathcal{F}$ be a family of sets of functions $f \colon \mathcal{X}_1 \times \ldots \times \mathcal{X}_n \to \mathcal{Y}$, $\ell = \{0,1\}^* \cup \{\perp\}$ be a set of labels, and $\mathcal{N}$ be a set of clients. A DMCFE scheme for the function family $\mathcal{F}$ and the label set ℓ is a tuple of six algorithms $\mathcal{E}_{DMCFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{dKeyShare}, \mathsf{dKeyComb}, \mathsf{Enc}, \mathsf{Dec})$:

- $\mathsf{Setup}(\lambda, n)$: Takes as input a security parameter λ and the number of clients n and generates public parameters pp. We assume that all the remaining algorithms implicitly contain pp.
- $\mathsf{KeyGen}(\mathsf{id}_i)$: Takes as input a client-specific identifier, id_i, and outputs a secret key sk_i and an encryption key ek_i, unique to client i.
- $\mathsf{dKeyShare}(\mathsf{sk}_i, f)$: Takes as input a secret key sk_i and a function $f \in \mathcal{F}$ to computes a partial functional decryption key dk_i.
- $\mathsf{dKeyComb}(\{\mathsf{dk}_i\}_{i \in \mathcal{N}})$: Takes as input a set of n partial functional decryption keys $\{\mathsf{dk}_i\}_{i \in \mathcal{N}}$ and outputs the functional decryption key dk_f.
- $\mathsf{Enc}(\mathsf{ek}_i, x_{i,l})$: Takes as input an encryption key ek_i and a message x_i to encrypt under label $l \in \ell$ and outputs ciphertext $\mathsf{ct}_{i,l}$.
- $\mathsf{Dec}(\mathsf{dk}_f, \{\mathsf{ct}_{i,l}\}_{i \in \mathcal{N}})$: Takes as input a functional decryption key dk_f, and n ciphertexts under the same label l, and computes value $y \in \mathcal{Y}$.

3 `EncCluster` Framework

We present `EncCluster`, a novel approach that combines model compression via weight clustering, decentralized FE, and BF filter-based efficient encoding to enhance FL's privacy against inference attacks, while simultaneously reducing communication costs and the computational load on clients.

3.1 Notations

Let $\mathcal{A}$ represent the aggregation server and $\mathcal{N}$ the set of n clients, each with a local dataset $\mathcal{D}_n$, participating over R federated rounds. The model is f_θ, with weight parameters $\theta = (\theta_1, \ldots, \theta_d)$, while

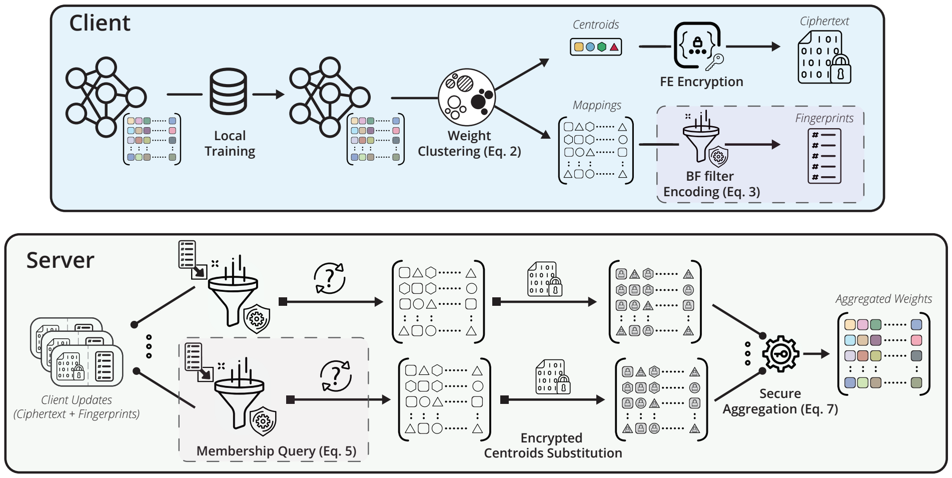

Client
Local Training
Weight Clustering (Eq. 2)
Centroids
FE Encryption
Ciphertext
Mappings
BF filter Encoding (Eq. 3)
Fingerprints
Server
Client Updates (Ciphertext + Fingerprints)
Membership Query (Eq. 5)
Encrypted Centroids Substitution
Secure Aggregation (Eq. 7)
Aggregated Weights

Figure 1: Overview of `EncCluster`'s training process. Clients train locally, cluster weights, encrypt centroids using DMCFE, and encode cluster-weight *mappings* into BF filter fingerprints. Server reconstructs mappings via BF filter queries, derives encrypted clustered weights, and aggregates them to update the model.

optimized parameters after local training are denoted by θ^*. For client n and round r, C_n^r represents the κ clusters formed from the model weights θ_n^r, with centroids $\mathcal{Z}_n^r$ and mapping $\mathcal{P}_n^r$. $\mathcal{H}_n^r$ is the fingerprint array for client n in round r, and $\hat{x}$ represents the encrypted value of x using DMCFE (20). We use r as the label l in DMCFE, as in (10; 21).

3.2 Threat Model and Assumptions

In `EncCluster`, we consider the following threat model:

- [*]***HbC Server***: Follows protocols but seeks private information from clients' encrypted model updates, possibly colluding with clients.
- ***Dishonest clients***: Collude with the server to access private data from other clients without altering model updates.
- ***HbC TPA***: Follows the DMCFE protocol but may seek private information from clients.

To ensure key confidentiality, we assume a secure key-provisioning process, such as Diffie-Hellman (22). Denial-of-service (DoS) and poisoning attacks are considered out of scope. With these assumptions, we analyze our system's resilience against the threat model in Section 4. A quantitative evaluation of privacy leakage is provided in Appx. D.

3.3 Delivering Scalable FE in FL with `EncCluster`

`EncCluster` framework starts with the *TPA* executing $\mathsf{Setup}(\lambda, n)$ to generate public parameters pp. Each client, identified by id_i, independently generates secret and encryption key pairs $(\mathsf{sk}_i, \mathsf{ek}_i)$, allowing them to create partial decryption keys via $\mathsf{dKeyShare}(\mathsf{sk}_i)$. Clients send their partial decryption keys and model updates to the server at each round, facilitating FL.

Training Overview. Training begins with the server initializing and distributing a neural network f_θ. In round r, clients locally train on their datasets $\mathcal{D}_n$, updating model parameters θ_n^{r*}. Next, they perform weight clustering, obtaining clusters $C_n^r = \langle \mathcal{Z}_n^r, \mathcal{P}_n^r \rangle$. The centroids $\mathcal{Z}_n^r$ are encrypted using ek_n to get $\hat{\mathcal{Z}}_n^r$, while $\mathcal{P}_n^r$ is encoded into a μ-wise BF filter, resulting in fingerprint array $\mathcal{H}_n^r$. The local training completes with each clients sending $\hat{\mathcal{Z}}_n^r$, $\mathcal{H}_n^r$, and their partial decryption key dk_n to the server. The server estimates $\mathcal{P}_n^r$ using $\mathcal{H}_n^r$ via querying, and forms encrypted weights $\hat{\theta}_n^r$ by replacing the positions indexed by $\mathcal{P}_n$ with the encrypted centroids from $\hat{\mathcal{Z}}_n^r$. It then combines the partial decryption keys to compute dk_f and securely aggregates the encrypted weights. Fig. 1 illustrates `EncCluster`'s training process, and Algorithm 1 outlines the `EncCluster` algorithm.

[*]Honest but Curious.

Efficient Client-side Encryption. Standard cryptographic encryption of model updates imposes high computational and communication costs due to the large prime numbers involved. As security levels increase, these costs rise exponentially. To address this, `EncCluster` applies weight clustering before encryption, reducing model weights θ to a compact set of centroids $\mathcal{Z}_n$. This minimizes the encryption workload and data transmission during FL by requiring only the centroids to be encrypted. Formally, client n minimizes the following loss:

$$\min_{\theta} \mathcal{L}_n(\theta) = \mathcal{L}_{wc}(\theta^*_n, \mathcal{Z}_n) = \sum_{j=1}^{\kappa} \sum_{i=1}^{d} u_{ij} \cdot ||\theta^*_{n,i} - z_{n,j}||^2 \ , \qquad \text{where } \theta^*_n = \min_{\theta_n} \mathcal{L}_{ce}(\mathrm{f}_{\theta_n}(\mathcal{D}_n)) \tag{4}$$

Here, $\mathcal{L}_{ce}$ is the cross-entropy loss on the dataset $\mathcal{D}_n$, $\mathcal{Z}_n$ represents the centroids, and u_{ij} indicates whether weight θ_i belongs to cluster j. The client first optimizes the model parameters θ for accuracy on D_n, then minimizes the clustering loss $\mathcal{L}_{wc}$ on the post-training parameters θ^*.

To protect against HbC *TPA* and dishonest clients, `EncCluster` encodes the cluster-weights mapping $\mathcal{P}_n$ into a 4-wise BF filter using 8 bits per parameter ($\xi = 8$). Each key in $\mathcal{U}_n = \{(i, \mathcal{P}_{n,i}) \mid i \in \{1, \ldots, d\}\}$, consisting of a weight position and its corresponding cluster, is inserted into the filter to produce the fingerprint array $\mathcal{H}_n$. The hashing operation ensures that $\mathcal{U}_n$ cannot be deduced from $\mathcal{H}_n$, protecting against preimage attacks. Reconstruction of $\mathcal{P}_n$ from $\mathcal{H}_n$ relies on a shared seed s_n between clients and the server, ensuring secure and accurate estimation. While this mechanism does not offer the same security as cryptographic FE, it effectively protects sensitive data while avoiding the computational complexity of encryption.

Secure Aggregation. After local training in round r, the server $\mathcal{A}$ synthesizes a new global model from clients' fingerprints $\mathcal{H}^r_n$ and encrypted centroids $\hat{\mathcal{Z}}^r_n$. The server reconstructs each client's BF filter using $\mathcal{H}^r_n$ and their unique seed s_n, estimating the cluster-weights mappings $\mathcal{P'}^r_n$ through membership queries:

$$\mathcal{P'}^r_n = \{j \mid \text{Member}\,(i, j) = \text{true}\}_{i \in \{d\},\ j \in \{\kappa\}} \ , \tag{5}$$

where Member$(\cdot)$ operates as in Eq. 3. Due to the low false positive rate of BF filters, $\mathcal{P'}^r_n \approx \mathcal{P}^r_n$. The server then replaces positions in $\mathcal{P'}^r_n$ with the corresponding encrypted centroids from $\hat{\mathcal{Z}}^r_n$, yielding the encrypted updated model weights $\hat{\theta}^r_n$. By combining clients' partial decryption keys, the server derives the functional decryption key dk_f and computes the aggregated global model:

$$\theta^{r+1} = \left\{ \mathsf{Dec}\left(\{\hat{\theta}^r_{n,i}\}_{n \in \mathcal{N}}, \mathsf{dk}_f \right) \right\}_{i \in \{d\}} \ , \tag{6}$$

where $\mathsf{Dec}(\cdot)$ decrypts the aggregated weights, forming the updated global model for the next federated round. In `EncCluster`, aggregation can be plain or weighted (e.g., FedAvg (2)). We use the latter, scaling centroids by sample size before encryption and normalizing θ^{r+1} by the total number of samples after aggregation. Algorithm 2 provides an overview of the secure aggregation mechanism.

4 Security and Privacy Analysis

Both server $\mathcal{A}$ and the *TPA* operate as HbC entities, with dishonest clients raising concerns about passive attacks that could extract client information (e.g., model updates) during FL training. Our analysis evaluates how these entities could attempt to access unauthorized information and how `EncCluster` counters these threats.

Inference Attacks I (*Reconstruction Attack*): Here, we assess the privacy risks associated with weight clustering, focusing on potential reconstruction attacks aimed at deriving a client's weights, θ_n, from the aggregated weights, $\theta_{\mathcal{A}}$, using data from any aggregation round, r. Despite DMCFE protection, which secures client centroids from external access, attackers may attempt to infer client's centroids by clustering $\theta_{\mathcal{A}}$ into $C_{\mathcal{A}}$ and reconstruct θ_n, assuming $C_{\mathcal{A}}$ mirrors C_n. The attack strategy involves optimizing the placement of $C_{\mathcal{A}}$ within θ_n to reduce the discrepancy between the estimated weights, $\hat{\theta}_n$, and the actual weights, θ_n, typically by minimizing MSE$\left(\theta_{\mathcal{A}},\ \hat{\theta}_n\right)$. In `EncCluster`, such attacks can be initiated by:

- **Dishonest Clients**: Without direct access to $\mathcal{P}n$, clients must infer cluster-weight mappings to approximate θ_n, with the objective of minimizing MSE $\left(\theta_{\mathcal{A}},\ \hat{\theta}_n\right)$. The computational complexity of this process is $O(\kappa^d)$, where d represents the dimensionality of the weights, rendering the task increasingly infeasible as d and κ grow.
- **HbC Server**: Utilizing the estimated cluster-weight mappings $\mathcal{P}'n$, the server $\mathcal{A}$ can attempt to deduce client weights $\hat{\theta}n$ using $\mathcal{C}_{\mathcal{A}}$ and $\mathcal{P}'n$. This process has a computational complexity of $O(\kappa!)$, rendering attacks impractical for $\kappa > 32$. Even if reconstruction is successful, the privacy leakage remains bounded (see Eq. 11). It is important to note that this analysis assumes $\mathcal{C}_{\mathcal{A}} \approx \mathcal{C}_n$, an assumption often not true in pragmatic federated settings (see Appx. C). We provide A quantitative evaluation of this attack in Appx. D.

Inference Attack II (*Compromised TPA*): The *TPA*'s role in initializing DMCFE and coordinating client-specific key generation (sk, ek) using unique identifiers (id) is critical. However, being an HbC entity it could misuse id to access clients' sk and ek. By intercepting communications from client j, a compromised TPA might decrypt Z_j^r using "*dummy*" updates and perform secure aggregation. Yet, access to j's raw centroid values does not facilitate the accurate reconstruction of θ_j^r. The complexity involved in deducing the cluster-to-weights mapping $\mathcal{U}_n$ from $\mathcal{H}_j^r$, compounded by the BF filters' sensitivity to unknown seeding (s_j), safeguards data integrity against a compromised TPA. Furthermore, *TPA* could aim to estimate clients' model weights using Z_j^r and the aggregated parameters θ^r by replacing each weight in θ^r with its nearest centroid in Z_j^r. However, variability in weight clustering optimization (Eq. 1) across clients leads to flawed estimations, underscoring `EncCluster`'s defense against compromised TPA scenarios.

5 Experiments

Datasets and FL Settings. We conduct experiments on 4 diverse image classification datasets. Note that we focus solely on classification tasks; yet `EncCluster` poses no restriction on the underlying downstream task. We train 4 neural architectures from scratch designed for on-device learning, namely ResNet-20 (23), ConvMixer-256/8 (24), ConvNeXt (25), and MobileNet (26). Additionally, to demonstrate `EncCluster`'s efficacy with vision FMs in FL, we employ popular ViT architectures pre-trained in self-supervised manner, such as CLIP (27), DINOv2 (28), where, we fine-tune the last 5 transformer blocks, similar to (29; 13). Our FL simulations are performed using Flower (30), with key parameters: number of clients (M), rounds (R), local epochs (E), clients' participation rate (ρ), class concentration (γ), and number of clusters (κ). For a detailed experimental setup, refer to Appx. B.1.

Baselines. We evaluate `EncCluster` in terms of accuracy, encryption complexity, and total volume of communicated data relative to *FedAvg* (×*times FedAvg*). We introduce FedAvg$_{wc}$, which incorporates weight clustering and Huffman encoding before transmitting models to the server, to analyze the impact of weight clustering. Additionally, we consider SEFL (31) as a baseline owing to its analogous use of HE with pruning for scalable privacy-preserving FL. We also compare `EncCluster` with DeTrust-FL (21) to demonstrate its advantages over other FE-based FL adaptations. To ensure a fair comparison, all baselines share the same weight initialization, while we use a fixed R across experiments to facilitate a direct comparison of required data transfer volumes across baselines (lower is better).

5.1 Privacy-Efficiency Trade-Off

Self-Supervised Pretrained FMs. Here, we evaluate the effectiveness of `EncCluster` with vision FMs in FL, where strong privacy guarantees are paramount. We use CLIP ViT-B/32 with 30 clients ($N = 30$) under both *IID* and *Non-IID* data distributions to compare `EncCluster`'s efficiency against traditional FE-based approaches. Table 1 shows that `EncCluster` maintains stringent privacy protections and delivers significant reductions in computation and communication, even with FMs integrated into FL. Compared to DeTrust-FL, `EncCluster` reduces communication demands by up to 13 times and slashes encryption time from potentially over an hour per round to just 2.34 seconds, all while keeping model performance within 1% of *FedAvg*. These results underscore `EncCluster`'s ability to effectively balance privacy and efficiency, enabling the use of FE with FMs in scenarios where it was previously infeasible.

Training Models from Scratch. We train the "*lightweight*" ResNet-20 (23) from scratch with 30 clients ($N = 30$) across both *IID* and *Non-IID* data distributions to evaluate

Table 1: Evaluation of `EncCluster` in terms of accuracy loss (δ-Acc), data transmission and clients' encryption times versus *FedAvg* for CLIP ViT-B/32 (27). Federated parameters are set to N=30, ρ=1, and E=1.

Dataset	Approach	IID ($\gamma \approx 1.0$)		non-IID ($\gamma \approx 0.2$)		*Data Transmitted (× times FedAvg)*	*Client-side Encryption (s)*
		Accuracy	*δ-Acc.*	*Accuracy*	*δ-Acc.*		
CIFAR-100	FedAvg$_{wc}$	*77.35*	-0.21	*75.49*	-0.56	0.043	—
	DeTrust-FL		-0.15		-0.14	3.758	3720.04
	EncCluster (**Ours**)		-0.34		-0.67	0.255	**2.34**
EMNIST	FedAvg$_{wc}$	*94.89*	-0.18	*93.13*	-0.37	0.041	—
	DeTrust-FL		-0.09		-0.12	3.754	3594.88
	EncCluster (**Ours**)		-0.29		-0.45	0.254	**2.33**
Food-101	FedAvg$_{wc}$	*86.72*	-0.26	*84.71*	-0.49	0.043	—
	DeTrust-FL		-0.14		-0.17	3.759	3723.09
	EncCluster (**Ours**)		-0.32		-0.67	0.255	**2.34**

Table 2: Evaluation of `EncCluster` in terms of accuracy loss (δ-Acc), data transmission and clients' encryption times versus *FedAvg* for ResNet-20 (23). Federated parameters are set to N=30, ρ=1, and E=1. Partial client participation experiments are in Table 4.

Dataset	Approach	IID ($\gamma \approx 1.0$)		non-IID ($\gamma \approx 0.2$)		*Data Transmitted (× times FedAvg)*	*Client-side Encryption (s)*
		Accuracy	*δ-Acc.*	*Accuracy*	*δ-Acc.*		
CIFAR-10	FedAvg$_{wc}$	*89.07*	-0.12	*83.12*	-0.71	0.034	—
	DeTrust-FL		-0.09		-0.10	3.743	329.53
	SEFL		-0.98		-1.74	2.581	7.64
	EncCluster (***Ours***)		-0.32		-0.79	0.284	**2.04**
CIFAR-100	FedAvg$_{wc}$	*61.33*	-1.07	*54.37*	-1.64	0.035	—
	DeTrust-FL		-0.11		-0.09	3.757	338.61
	SEFL		-2.42		-3.88	2.656	8.01
	EncCluster (***Ours***)		-1.21		-1.67	0.285	**2.03**

whether `EncCluster` maintains its efficiency with smaller models as observed with FMs. Table 2 demonstrates that `EncCluster` significantly mitigates accuracy loss, boosts communication efficiency, and speeds up client-side computations, even when training models from scratch in FL. Compared to DeTrust-FL, `EncCluster` slashes communication demands by up to 13× and cuts encryption time from 325 seconds to just over 2 seconds. Against SEFL, `EncCluster` particularly shines in *Non-IID* scenarios, where SEFL faces up to a 4% greater accuracy drop. Meanwhile, `EncCluster` maintains a δ-Acc of 2%, achieves a 9-fold increase in encryption speed, and reduces communication costs by 4×. Due to limited space, we provide additional experiments under partial client participation in Appx. B.2.

5.2 `EncCluster`'s Scalability

Encryption Key Sizes. To assess `EncCluster`'s scalability with different encryption levels, we experimented with various key sizes (KS) of DMCFE on ResNet-20 with CIFAR-10 for N=10 in IID settings ($\gamma \approx 1.0$). We tracked encryption time and communication costs, measuring how KS influences computations and the size of encrypted messages. Additionally, we tested `EncCluster`$_{noBF}$, a variant substituting BF filter-based with Huffman encoding to assess the overhead of BF filters. Our findings, shown in Fig. 2, reveal that `EncCluster` maintains low encryption times across all key sizes, peaking at 6.44 seconds for KS=521. In contrast, DeTrust-FL's encryption time increases with larger KS, and SEFL is consistently about 4× slower than `EncCluster`. The overhead from BF filters is minimal, showcasing efficient scalability and added privacy benefits. Further details on the BF filters' overhead are available in Appx.B.3. Communication costs, as depicted in Fig. 2b, remain nearly constant and significantly lower for `EncCluster` compared to DeTrust-FL, SEFL, and even *FedAvg*. Weight clustering reduces data to just κ values, minimizing overhead, while `EncCluster`$_{noBF}$ achieves even lower communication costs, proving ideal for bandwidth-constrained FL environments.

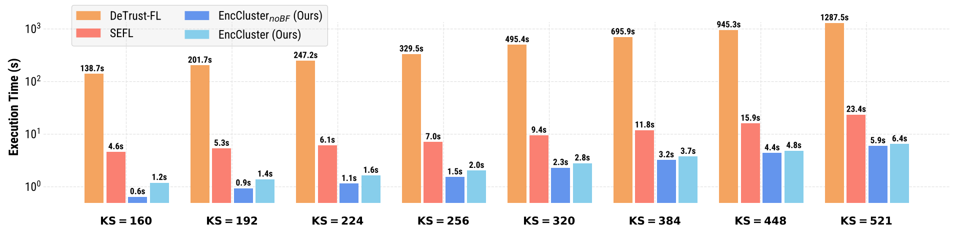
Cluster Sizes. We now investigated the impact of cluster size κ on model performance using CIFAR-10 with N=10, full client participation, and IID settings ($\gamma \approx 1.0$). Experiments varied κ from 16 to 512, alongside key size (KS) of DMCFE to explore the relationship between cluster size and communication overhead under different security levels. As shown in Fig. 3a, model accuracy improves significantly with increasing κ, rising from 68% at κ=16 to over 89% at κ=512, highlighting the effectiveness of weight clustering in capturing clients' post-training parameters (see Eq. 1). Accuracy improvements are consistent across various KS values (Fig. 3b), indicating that performance is mainly influenced by κ rather than encryption level. However, accuracy plateaus at $\kappa = 128$, where further increases yield diminishing returns while escalating communication costs and computational overhead ($\mathcal{O}(\kappa \cdot d)$). KS has minimally impacts on data volume, fluctuating slightly between 0.285 and 0.303, as only κ values are encrypted regardless of KS. Thus, `EncCluster` scales efficiently to higher encryption levels without increasing client-side overhead, providing scalable FE in FL.

Neural Architectures. We evaluate `EncCluster`'s performance across various neural architectures, focusing on accuracy loss and encryption time, excluding communication results as they mirror previous findings tied to weight clustering (see Fig. 2b). Encryption time accounts for the injection of cluster-weight mappings into BF filters, which depends on model size (from 0.16M in ConvNeXt-Tiny to 3.4M in MobileNet). As shown in Table 3, `EncCluster` consistently achieves minimal accuracy loss across all architectures. For larger models like MobileNet and ConvMixer-256/6, performance impact is even lower, with δ-Acc as low as -0.19 for CIFAR-10 and -0.97 for CIFAR-100. Encryption times confirm `EncCluster`'s computational efficiency, with only a 0.1-second increase across models due to low-cost BF filter hashing. These results demonstrate `EncCluster`'s scalability and minimal performance impact, making it suitable for secure, efficient FL, even for large-scale FMs.

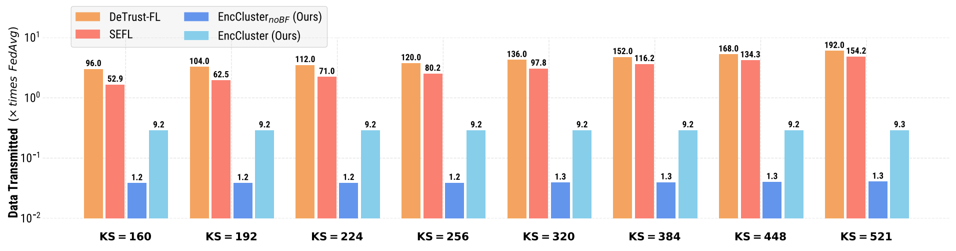
6 Related Work

Privacy-preserving FL. FL is vulnerable to inference attacks during model updates (3; 4). DP has been used to mitigate this (5; 32) but often degrades model performance (7). Cryptographic alternatives like HE (9) and SMPC (33) preserve privacy without affecting accuracy but impose high computational and communication costs (34). Optimizations such as pruning and quantization (31; 35) help but struggle with balancing security and resource constraints. FE has emerged as a lower-overhead alternative, supporting complex computations like weighted averaging (36; 10; 21). Decentralized FE (20; 37) improves privacy by using unique keys for each client, avoiding the risks of HE's single-shared key. However, many FE-based approaches, like HybridAlpha (10) and CryptoFE (36), require a trusted third party, limiting practicality. DeTrust-FL (21) mitigates this by enabling clients to collaboratively generate decryption keys, but it requires fixed client participation. These approaches still encrypt all model parameters, leading to significant computational and communication overhead, especially for resource-limited devices. `EncCluster` integrates FE with weight clustering and probabilistic filters, improving efficiency while maintaining strong privacy.

Communication-efficient FL. FL communication efficiency can be enhanced through methods like adaptive optimizers (38) and client sampling (39), which speed up convergence and reduce transmission. Compression techniques like sparsification (40; 41), quantization (42; 43), and low-rank approximation (44; 45) also reduce data transmission. Weight clustering (46) compresses model

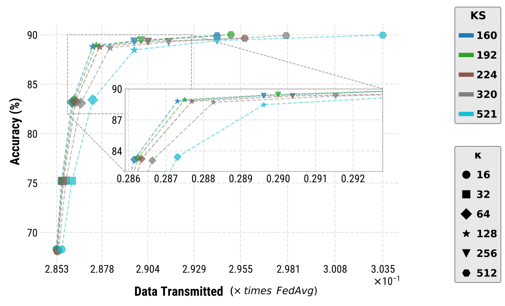


(a) Execution time of encryption process.



(b) Volume of transmitted data compared to standard *FedAvg*.

Figure 2: Efficiency of `EncCluster` on ResNet-20 in IID settings ($\gamma \approx 1.0$) with CIFAR-10, $N = 10$, $\rho = 1$. (a) Encryption time (log scale); (b) Transmitted data volume and bits-per-parameter (*bpp*) compared to *FedAvg*.


(a) Impact of cluster size (κ) and key sizes (KS).

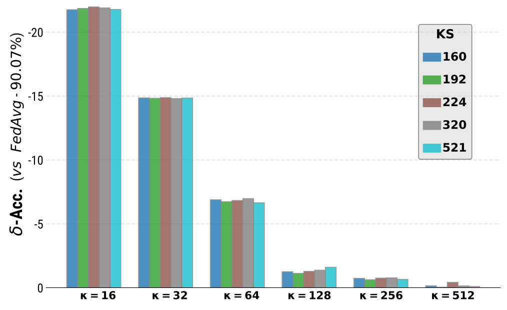

(b) Accuracy loss (δ-Acc.) across cluster sizes (κ).

Figure 3: Evaluation of EncCluster with varying cluster sizes (κ) and key sizes (KS) using ResNet-20 on CIFAR-10 (IID, $\gamma \approx 1.0$). Subfigure (a) compares test accuracy and data transmitted to *FedAvg*; (b) shows accuracy loss for different κ and KS. Federated parameters are $N = 10$, $R = 100$, and $\rho = 1.0$.

parameters with minimal accuracy loss. Recent work (47; 48) incorporating weight clustering into FL has significantly reduced communication costs, making it ideal for low-bandwidth environments. SEFL (31) combines HE with gradient pruning, and BatchCrypt (35) uses quantization and batching; yet both face scalability and communication challenges. EncCluster addresses these limitations by combining weight clustering with decentralized FE, increasing encrypted parameters per operation, reducing communication overhead, and enhancing privacy with minimal computational cost through probabilistic filter-based encoding.

7 Conclusions

We introduce EncCluster, a framework that bridges the gap between upholding privacy guarantees against inference attacks on model updates and delivering operational efficiency and scalability within FL. EncCluster uses model compression via weight clustering to transmit compressed model updates during training, secured by combining decentralized FE with BF filter-based encoding. Through extensive testing on 4 datasets and 5 architectures, EncCluster substantially reduces communication costs (>13× reduction) and computational demands (>1000-fold speedup) with minimal accuracy loss; thereby delivering robust privacy without reliance on trusted TPAs. Moreover, our work pioneers the integration of strong FE privacy guarantees with FMs in FL, marking a significant advancement in developing FL systems where privacy and efficiency coexist as complementary rather than conflicting objectives.

Limitations. While EncCluster, combined with DMCFE, significantly improves client privacy with near-constant communication overhead and minimal impact on training times, its adaptation to other cryptographic frameworks has not been explored. Future research could integrate EncCluster with emerging decentralized FE schemes (37; 21), eliminating the need for a TPA and seamlessly integrating FE into existing FL systems. Additionally, pre-setting cluster sizes in weight clustering, which depends on model and task complexity, limits adaptability across diverse FL systems. Notably, the integration of recent adaptive weight clustering schemes (48) within EncCluster— capable of dynamically adjusting cluster size based on model and task needs — offers a promising path to overcome such challenges.

Broader Impacts. Our evaluation across widely utilized deep learning architectures in FL highlights a significant gap in recent research (10; 21; 36; 31; 35), emphasizing the need to assess performance under complex models and challenging tasks. By open-sourcing our code, we aim to foster further exploration into approaches that simultaneously prioritize privacy and efficiency rather than treating

Table 3: EncCluster across various neural architectures on CIFAR-10 (IID, $\gamma \approx 1.0$), reporting *FedAvg* accuracy, EncCluster's accuracy loss (δ-Acc), and client-side encryption time with $N = 10$, $R = 100$, $\rho = 1.0$.

Dataset	ConvNeXt-Tiny		ResNet-20		MobileNet		ConvMixer-256/8	
	Accuracy	*δ-Acc.*	*Accuracy*	*δ-Acc.*	*Accuracy*	*δ-Acc.*	*Accuracy*	*δ-Acc.*
CIFAR-10	86.47	-0.36	89.07	-0.32	91.48	-0.24	92.35	-0.19
CIFAR-100	60.35	-1.29	61.33	-1.19	70.02	-0.97	72.64	-1.02
Encryption time (s)	2.02		2.03		2.08		2.12	

them as separate challenges in FL. While `EncCluster` does not explicitly address eavesdropping threats, BF filter-based encoding, which relies on a unique seed value for data reconstruction, naturally safeguards against such threats. Additionally, our evaluations primarily focused on accuracy as a performance metric. However, as highlighted in (49), model compression techniques may disproportionately impact different subgroups of data, raising fairness concerns in communication-efficient FL frameworks that warrant further attention from the community.

Acknowledgements

The work presented in this paper is partially performed in the context of the Distributed Artificial Intelligent Systems (DAIS) project supported by the ECSEL Joint Undertaking (JU) under grant agreement No.101007273.

References

[1] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, D. Bacon, Federated learning: Strategies for improving communication efficiency, arXiv preprint arXiv:1610.05492 (2016).

[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, B. A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: Artificial intelligence and statistics, PMLR, 2017, pp. 1273–1282.

[3] M. Nasr, R. Shokri, A. Houmansadr, Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning, in: 2019 IEEE symposium on security and privacy (SP), IEEE, 2019, pp. 739–753.

[4] R. Shokri, M. Stronati, C. Song, V. Shmatikov, Membership inference attacks against machine learning models, in: 2017 IEEE symposium on security and privacy (SP), IEEE, 2017, pp. 3–18.

[5] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, L. Zhang, Deep learning with differential privacy, in: Proceedings of the 2016 ACM SIGSAC conference on computer and communications security, 2016, pp. 308–318.

[6] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, K. Seth, Practical secure aggregation for privacy-preserving machine learning, in: proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 2017, pp. 1175–1191.

[7] Y. Yang, B. Hui, H. Yuan, N. Gong, Y. Cao, {PrivateFL}: Accurate, differentially private federated learning via personalized data transformation, in: 32nd USENIX Security Symposium (USENIX Security 23), 2023, pp. 1595–1612.

[8] M. S. Riazi, K. Laine, B. Pelton, W. Dai, Heax: An architecture for computing on encrypted data, in: Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems, 2020, pp. 1295–1309.

[9] C. Fang, Y. Guo, Y. Hu, B. Ma, L. Feng, A. Yin, Privacy-preserving and communication-efficient federated learning in internet of things, Computers & Security 103 (2021) 102199.

[10] R. Xu, N. Baracaldo, Y. Zhou, A. Anwar, H. Ludwig, Hybridalpha: An efficient approach for privacy-preserving federated learning, in: Proceedings of the 12th ACM workshop on artificial intelligence and security, 2019, pp. 13–23.

[11] S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, R. Zhang, Y. Zhou, A hybrid approach to privacy-preserving federated learning, in: Proceedings of the 12th ACM workshop on artificial intelligence and security, 2019, pp. 1–11.

[12] M. Dehghani, J. Djolonga, B. Mustafa, P. Padlewski, J. Heek, J. Gilmer, A. Steiner, M. Caron, R. Geirhos, I. Alabdulmohsin, R. Jenatton, L. Beyer, M. Tschannen, A. Arnab, X. Wang, C. Riquelme, M. Minderer, J. Puigcerver, U. Evci, M. Kumar, S. Van Steenkiste, G. F. Elsayed, A. Mahendran, F. Yu, A. Oliver, F. Huot, J. Bastings, M. P. Collier, A. A. Gritsenko, V. Birodkar, C. Vasconcelos, Y. Tay, T. Mensink, A. Kolesnikov, F. Pavetić, D. Tran, T. Kipf, M. Lučić, X. Zhai, D. Keysers, J. Harmsen, N. Houlsby, Scaling vision transformers to 22 billion parameters, ICML'23, JMLR.org, 2023.

[13] V. Tsouvalas, Y. M. Asano, A. Saeed, Federated fine-tuning of vision foundation models via probabilistic masking, in: ICML 2024 Workshop on Foundation Models in the Wild, 2024.
URL `https://openreview.net/forum?id=VDgx8JLtid`

[14] Z. Peng, X. Fan, Y. Chen, Z. Wang, S. Pan, C. Wen, R. Zhang, C. Wang, Fedpft: Federated proxy fine-tuning of foundation models (2024). `arXiv:2404.11536`.
URL `https://arxiv.org/abs/2404.11536`

[15] D. P. Nguyen, J. P. Munoz, A. Jannesari, Flora: Enhancing vision-language models with parameter-efficient federated learning (2024). `arXiv:2404.15182`.
URL `https://arxiv.org/abs/2404.15182`

[16] S. Lloyd, Least squares quantization in pcm, IEEE transactions on information theory 28 (2) (1982) 129–137.

[17] T. M. Graf, D. Lemire, Binary fuse filters: Fast and smaller than xor filters, Journal of Experimental Algorithmics (JEA) 27 (1) (2022) 1–15.

[18] A. Appleby, Murmurhash3.(2016), URL: https://github. com/aappleby/smhasher/wiki/MurmurHash3 (2016).

[19] D. Boneh, A. Sahai, B. Waters, Functional encryption: Definitions and challenges, in: Theory of Cryptography: 8th Theory of Cryptography Conference, TCC 2011, Providence, RI, USA, March 28-30, 2011. Proceedings 8, Springer, 2011, pp. 253–273.

[20] J. Chotard, E. Dufour Sans, R. Gay, D. H. Phan, D. Pointcheval, Decentralized multi-client functional encryption for inner product, in: Advances in Cryptology–ASIACRYPT 2018: 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2–6, 2018, Proceedings, Part II 24, Springer, 2018, pp. 703–732.

[21] R. Xu, N. Baracaldo, Y. Zhou, A. Anwar, S. Kadhe, H. Ludwig, Detrust-fl: Privacy-preserving federated learning in decentralized trust setting, in: 2022 IEEE 15th International Conference on Cloud Computing (CLOUD), IEEE, 2022, pp. 417–426.

[22] R. Canetti, H. Krawczyk, Analysis of key-exchange protocols and their use for building secure channels, in: International conference on the theory and applications of cryptographic techniques, Springer, 2001, pp. 453–474.

[23] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[24] A. Trockman, J. Z. Kolter, Patches are all you need?, arXiv preprint arXiv:2201.09792 (2022).

[25] S. Woo, S. Debnath, R. Hu, X. Chen, Z. Liu, I. S. Kweon, S. Xie, Convnext v2: Co-designing and scaling convnets with masked autoencoders (2023). arXiv:2301.00808.

[26] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, Mobilenetv2: Inverted residuals and linear bottlenecks (2019). arXiv:1801.04381.

[27] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, I. Sutskever, Learning transferable visual models from natural language supervision, in: ICML, 2021.

[28] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, M. Assran, N. Ballas, W. Galuba, R. Howes, P.-Y. Huang, S.-W. Li, I. Misra, M. Rabbat, V. Sharma, G. Synnaeve, H. Xu, H. Jegou, J. Mairal, P. Labatut, A. Joulin, P. Bojanowski, Dinov2: Learning robust visual features without supervision (2023). arXiv:2304.07193.

[29] M. Zhao, T. Lin, F. Mi, M. Jaggi, H. Schütze, Masking as an efficient alternative to finetuning for pretrained language models, in: B. Webber, T. Cohn, Y. He, Y. Liu (Eds.), Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Online, 2020, pp. 2226–2241. doi:10.18653/v1/2020.emnlp-main.174.
URL https://aclanthology.org/2020.emnlp-main.174

[30] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, J. Fernandez-Marques, Y. Gao, L. Sani, K. H. Li, T. Parcollet, P. P. B. de Gusmão, et al., Flower: A friendly federated learning research framework, arXiv preprint arXiv:2007.14390 (2020).

[31] S. Mohammadi, S. Sinaei, A. Balador, F. Flammini, Secure and efficient federated learning by combining homomorphic encryption and gradient pruning in speech emotion recognition, in: International Conference on Information Security Practice and Experience, Springer, 2023, pp. 1–16.

[32] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. Quek, H. V. Poor, Federated learning with differential privacy: Algorithms and performance analysis, IEEE Transactions on Information Forensics and Security 15 (2020) 3454–3469.

[33] V. Mugunthan, A. Polychroniadou, D. Byrd, T. H. Balch, Smpai: Secure multi-party computation for federated learning, in: Proceedings of the NeurIPS 2019 Workshop on Robust AI in Financial Services, MIT Press Cambridge, MA, USA, 2019, pp. 1–9.

[34] S. Mohammadi, A. Balador, S. Sinaei, F. Flammini, Balancing privacy and performance in federated learning: A systematic literature review on methods and metrics, Journal of Parallel and Distributed Computing (2024) 104918.

[35] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, Y. Liu, {BatchCrypt}: Efficient homomorphic encryption for {Cross-Silo} federated learning, in: 2020 USENIX annual technical conference (USENIX ATC 20), 2020, pp. 493–506.

[36] X. Qian, H. Li, M. Hao, S. Yuan, X. Zhang, S. Guo, Cryptofe: Practical and privacy-preserving federated learning via functional encryption, in: GLOBECOM 2022-2022 IEEE Global Communications Conference, IEEE, 2022, pp. 2999–3004.

[37] J. Chotard, E. Dufour-Sans, R. Gay, D. H. Phan, D. Pointcheval, Dynamic decentralized functional encryption, in: Annual International Cryptology Conference, Springer, 2020, pp. 747–775.

[38] S. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečný, S. Kumar, H. B. McMahan, Adaptive federated optimization, arXiv preprint arXiv:2003.00295 (2020).

[39] W. Chen, S. Horvath, P. Richtarik, Optimal client sampling for federated learning, Transactions on Machine Learning Research (2022).

[40] A. F. Aji, K. Heafield, Sparse communication for distributed gradient descent, arXiv preprint arXiv:1704.05021 (2017).

[41] Y. Lin, S. Han, H. Mao, Y. Wang, W. J. Dally, Deep gradient compression: Reducing the communication bandwidth for distributed training, arXiv preprint arXiv:1712.01887 (2017).

[42] H. Xu, K. Kostopoulou, A. Dutta, X. Li, A. Ntoulas, P. Kalnis, Deepreduce: A sparse-tensor communication framework for federated deep learning, Advances in Neural Information Processing Systems 34 (2021) 21150–21163.

[43] S. Vargaftik, R. B. Basat, A. Portnoy, G. Mendelson, Y. B. Itzhak, M. Mitzenmacher, Eden: Communication-efficient and robust distributed mean estimation for federated learning, in: International Conference on Machine Learning, PMLR, 2022, pp. 21984–22014.

[44] H. Mozaffari, V. Shejwalkar, A. Houmansadr, Frl: Federated rank learning, arXiv preprint arXiv:2110.04350 (2021).

[45] A. Mohtashami, M. Jaggi, S. Stich, Masked training of neural networks with partial gradients, in: International Conference on Artificial Intelligence and Statistics, PMLR, 2022, pp. 5876–5890.

[46] S. Han, H. Mao, W. J. Dally, Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, International Conference on Learning Representations (ICLR) (2016).

[47] S. Khalilian, V. Tsouvalas, T. Ozcelebi, N. Meratnia, Fedcode: Communication-efficient federated learning via transferring codebooks, arXiv preprint arXiv:2311.09270 (2023).

[48] V. Tsouvalas, A. Saeed, T. Ozcelebi, N. Meratnia, Communication-efficient federated learning through adaptive weight clustering and server-side distillation, in: ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2024, pp. 5805–5809. doi:10.1109/ICASSP48485.2024.10447174.

[49] S. Hooker, N. Moorosi, G. Clark, S. Bengio, E. Denton, Characterising bias in compressed models (2020). arXiv:2010.03058.

[50] Q. Li, Y. Diao, Q. Chen, B. He, Federated learning on non-iid data silos: An experimental study, in: 2022 IEEE 38th International Conference on Data Engineering (ICDE), IEEE, 2022, pp. 965–978.

[51] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al., Advances and open problems in federated learning, Foundations and Trends® in Machine Learning 14 (1–2) (2021) 1–210.

[52] W. Wei, L. Liu, M. Loper, K.-H. Chow, M. E. Gursoy, S. Truex, Y. Wu, A framework for evaluating gradient leakage attacks in federated learning, arXiv preprint arXiv:2004.10397 (2020).

A EncCluster Algorithm

For completeness, this section includes the `EncCluster` algorithm detailed in Algorithm 1 and the secure aggregation process outlined in Algorithm 2.

Algorithm 1 `EncCluster`: Efficient Functional Encryption in FL through weight clustering and probabilistic filters. Here, η refers to the learning rate, while *SecureAggr* refers to the secure aggregation process on server-side presented in Algorithm 2.

```
1: $DMCFE\_Init(\lambda, \mathcal{N})$
2: Server $\mathcal{A}$ initializes model parameters $\theta$, and computes the total number of samples, $|D| = \sum_{i \in N} |D_i|$.
3: for $r = 1$ to $R$ do
4:     for $n \in \mathcal{N}$ in parallel do
5:         $\left(\mathcal{Z}_n^r, \mathcal{P}_n^r\right) \leftarrow ClientUpdate(\theta^r)$
6:         for $z \in \mathcal{Z}_n^r$ do
7:             $\hat{z} \leftarrow \mathsf{Enc}(\mathsf{ek}_n, z)$
8:         end for
9:         $\mathcal{V}_n^r \leftarrow \left\{\left(i, \mathcal{P}_{n,i}^r\right)\right\}_{i \in \{d\}}$
10:        $\mathcal{H}_n^r \leftarrow \bigcup_{i \in \mathcal{V}_n^r} \phi(i)$                     ▷ // See Equation 2
11:        $\mathsf{dk}_n^r \leftarrow \mathsf{dKeyShare}\left(\mathsf{sk}_n, |D_n|\right)$
12:    end for
13:    $\mathsf{dk}^r \leftarrow \mathsf{dKeyComb}(\{\mathsf{dk}_n^r\}_{n \in \mathcal{N}})$
14:    $\theta^{r+1} \leftarrow SecureAggr\left(\left\{\left(\hat{\mathcal{Z}}_n^r, \mathcal{H}_n^r\right)\right\}_{i \in \mathcal{N}}, \mathsf{dk}^r\right)$
15: end for
16: procedure $DMCFE\_Init(\lambda, \mathcal{N})$
17:     $\mathsf{pp} \leftarrow \mathsf{Setup}\left(\lambda, |\mathcal{N}|\right)$
18:     for $n \in \mathcal{N}$ clients in parallel do
19:         $\left(\mathsf{ek}_n, \mathsf{sk}_n\right) \leftarrow \mathsf{KeyGen}(\mathsf{id}_n)$
20:     end for
21: end procedure
22: procedure $ClientUpdate(\theta)$
23:     for epoch $e = 1, 2, \ldots, E$ do
24:         for batch $b \in \mathcal{D}_n$ do
25:             $\theta^* \leftarrow \theta - \eta \cdot \nabla_\theta \left(\mathcal{L}_{ce}\left(\mathrm{f}_\theta(b)\right)\right)$
26:         end for
27:     end for
28:     $\mathcal{Z} \leftarrow \{x \mid x \in \mathrm{rand}(\theta^*)\}_{|x| = \kappa}$                ▷ // Cluster initialization
29:     $(\mathcal{Z}, \mathcal{P}) \leftarrow \mathcal{L}_{wc}(\theta^*, \mathcal{Z})$
30:     return $(\mathcal{Z}, \mathcal{P})$
31: end procedure
```

Algorithm 2 *SecureAggr*: Server-side secure weighted aggregation directly on encrypted client updates.

```
1: Inputs: Clients' encrypted centroids and fingerprints, $\left\{\hat{\mathcal{Z}}_n, \mathcal{H}_n\right\}_{n \in \mathcal{N}}$, functional decryption key $\mathsf{dk}$, and total number of training samples, $|D|$.
2: Output: Aggregated model parameters $\theta^{\mathrm{agg}}$.
3: for $n \in N$ do
4:     $\mathcal{P}'_n \leftarrow \{j \mid \mathrm{Member}\,(i, j)\}_{i \in \{d\},\, j \in \{\kappa\}}$          ▷ // Equation 5
5:     $\hat{\theta}_n \leftarrow \left\{\hat{\mathcal{Z}}_{n,i}\right\}_{i \in \mathcal{P}'_n}$               ▷ // Enc. Cluster Substitution
6: end for
7: $\theta^{\mathrm{agg}} \leftarrow \frac{1}{|D|}\left\{\mathsf{Dec}\left(\{\hat{\theta}_{n,i}\}_{n \in \mathcal{N}}, \mathsf{dk}\right)\right\}_{i \in \{d\}}$
8: return $\theta^{\mathrm{agg}}$
```

B Additional Experiments

B.1 Experimental Details

Training Parameters: For our experiments, we simulate a federated environment using Flower (30) with key parameters: number of clients ($M = 30$), rounds (R), local training epochs ($E = 1$), client participation rate (ρ), class concentration (γ), and clusters ($\kappa = 128$) with exceptions of experiments detailed in Figure 3. Clients train with a batch size of 64, using the Adam optimizer with a learning rate of $1e-3$. For $\rho = 1$ (IID and non-IID), we use 100 rounds. For $\rho << 1$, we extend to 300 rounds for IID and 500 for non-IID. In scenarios where $\rho < 1.0$, client selection in each round was randomized. Unless specified otherwise, a key size of 256 was used for DMCFE, indicated by the parameter KS. All experiments run on NVIDIA A10 GPUs in an internal cluster server with 96 CPU cores and one GPU per run.

Data Splits: Data is split across clients using a Dirichlet distribution $Dir(a)$ (50), where a controls class distribution. In IID settings, $a = 10$ ($\gamma \approx 1.0$), ensuring all classes are distributed evenly. In non-IID settings,

$a = 0.1$ ($\gamma \approx 0.2$), leading to skewed label distributions typical in FL scenarios(51). We fix the seed in data partitioning to ensure consistent data splits across experiments for direct comparison.

B.2 Additional Experiments with Partial Client Participation

In this section, we provide additional experiments performed under partial client participation (ρ=0.2) with ResNet-20 (23) on CIFAR-10/100. We report our findings in Table 4, where we also report the accuracy degradation and communication cost compared to *FedAvg*, as well as encryption times for clients.

Table 4: Evaluation of `EncCluster` in terms of accuracy loss (δ-Acc), data transmission and clients' encryption times versus *FedAvg* for ResNet-20 (23). Federated parameters are set to N=30, ρ=0.2, and E=1.

Dataset	Approach	IID ($\gamma \approx 1.0$)		non-IID ($\gamma \approx 0.2$)		*Data Transmitted ($\times$ times FedAvg)*	*Client-side Encryption (s)*
		Accuracy	*δ-Acc.*	*Accuracy*	*δ-Acc.*		
CIFAR-10	**FedAvg$_{wc}$**	*88.42*	-0.28	*81.91*	-0.89	0.032	—
	DeTrust-FL		-0.06		-0.12	3.744	326.15
	SEFL		-1.14		-1.97	2.569	7.63
	EncCluster (Ours)		-0.44		-1.05	**0.281**	**2.04**
CIFAR-100	**FedAvg$_{wc}$**	*60.07*	-1.32	*47.37*	-1.87	0.033	—
	DeTrust-FL		-0.09		-0.03	3.756	339.12
	SEFL		-2.35		-3.79	2.632	7.98
	EncCluster (Ours)		-1.67		-2.03	**0.279**	**2.03**

From Table 4, we note that `EncCluster`'s efficiency remains unaffected from the number of clients that participate in each FL round. More importantly, we note that `EncCluster` allows FE to work in flexible FL settings (i.e., a subset of clients in each federated round), a benefit of DMCFE cryptosystem.

B.3 Binary Fuse Filter Efficiency

To evaluate the impact of BF filters on computational complexity, communication overhead, and accuracy loss (δ-Acc), we tested `EncCluster` on ResNet-20 with $N = 30$, comparing it to $\texttt{EncCluster}_{noBF}$, a variant that replaces BF filters with Huffman encoding for transmitting cluster-weight mappings. Table 5 shows that BF filters have minimal impact on accuracy due to their near-perfect mapping reconstruction at the server, with a slight increase in encryption time from 1.72 to 2 seconds. In terms of communication, $\texttt{EncCluster}_{noBF}$ requires only 0.032 times the data of *FedAvg*, while adding BF filters increases this to 0.284 due to the bit requirement per filter entry (8.68 bits-per-entry). Despite this, `EncCluster`still significantly reduces communication costs compared to other privacy-preserving methods (see Table 2), while improving privacy. Significantly, $\texttt{EncCluster}_{noBF}$ proves particularly advantageous in FL systems with limited communication resources operating under less "*stringent*" threat models — such as when a fully-trusted authority exists — delivering computational efficiency while substantially reducing communication overhead to over 100× less than DeTrust-FL's.

Table 5: Effect of BF filters in `EncCluster` using ResNet-20 with $N = 30$. We report test accuracy loss (δ-Acc), upstream communication data relative to *FedAvg*, and client-side encryption times.

	ρ	CIFAR-10		CIFAR-100	
		EncCluster$_{noBF}$	*EncCluster*	*EncCluster$_{noBF}$*	*EncCluster*
IID	0.2	-0.31	-0.44	-1.31	-1.67
($\gamma \approx 1.0$)	1	-0.14	-0.32	-1.09	-1.21
non-IID	0.2	-0.94	-1.05	-1.85	-2.03
($\gamma \approx 0.2$)	1	-0.73	-0.79	-1.63	-1.67
Encryption time (s)		1.72	2.04	1.72	2.03
Data Transmitted ($\times$ FedAvg)		0.034	0.284	0.035	0.281

C Weight Clustering Convergence Analysis

Estimation Error Analysis due to Weight Clustering. In this section, we analyze the privacy implications due to the weight clustering process in `EncCluster`. Recall that θ^* refers to the original post-trained model weights, while θ denotes the clustered weights. We encode cluster-weight mappings using probabilistic filters, which introduce an error probability of $2^{-\text{bpe}}$ (where p denotes the false positive rate of the filter) leading to the assignment of a weight to an incorrect cluster. Note that the introduced error probability is independent across both clients and cluster dimensions. The estimation error between θ^* and θ can be computed as follows:

$$\mathbb{E}\left[\|\theta^* - \theta\|_2^2\right] = \sum_{i=1}^{d} \mathbb{E}\left[\left(\theta_i^* - \theta_i\right)^2\right] \tag{7}$$

$$= \sum_{i=1}^{d} \left((1-2^{-\mathrm{bpe}}) \cdot \mathbb{E}\left[\left(\theta_i^* - c_i\right)^2\right] + 2^{-\mathrm{bpe}} \cdot \mathbb{E}\left[\left(\theta_i^* - \tilde{c}_i\right)^2\right]\right) \tag{8}$$

$$= \sum_{i=1}^{d} \left((1-2^{-\mathrm{bpe}}) \cdot \left(\sum_{k=1}^{\kappa} \sum_{\theta_i^* \in C_k} \|\theta_i^* - c_k\|^2\right) + 2^{-\mathrm{bpe}} \cdot \left(\frac{1}{\kappa - 1} \sum_{k=1}^{\kappa} \|\theta_i^* - \tilde{c}_i\|^2\right)\right). \tag{9}$$

Here, $\tilde{c}_i$ refers to a randomly chosen centroid (any centroid apart from the correct one) due to the reconstruction error of the cluster-weights mapping. Assuming a uniform distribution of weights and centroids, the expected intra-cluster distance α (distance between weights within a given cluster) is given by $\alpha = \left(\sum_{k=1}^{\kappa} \sum_{\theta_i^* \in C_k} \left\|\theta_i^* - c_k\right\|^2\right)$, while the inter-cluster distance β (error due to the false positive rate of the probabilistic filter) is estimated by the average distance from each given weight belonging to a cluster to all other clusters' centroids, computed as $\beta = \left(\frac{1}{\kappa-1} \sum_{k=1}^{\kappa} \left\|\theta_i^* - \tilde{c}_i\right\|^2\right)$. While exact estimation of α and β are complex and depend on the specific characteristics of the data, we note that both α and β are bounded.

In terms of privacy amplifications due to weight clustering, we can consider the minimum discrepancy between θ^* and θ as worst case scenario, which occurs when both α and β take their minimal values (referred by D_{intra} and D_{inter}) across all d weight values. Thus, we can derive the following:

$$\mathbb{E}\left[\|\theta^* - \theta\|_2^2\right] \geq d\left((1-2^{-\mathrm{bpe}}) \cdot D_{intra} + 2^{-\mathrm{bpe}} \cdot D_{inter}\right) \tag{10}$$

Distributed Mean Estimation Error Analysis. We can now compute the the expected mean estimation error of server-side aggregated model to derive a privacy leakage estimation, similar to Equation 10. For this, we compute the lower bound of the mean estimation error between the true mean is $\bar{\theta^*}^{r+1} = \frac{1}{N}\sum_{i=1}^{N} {\theta^*}_i^r$ and our estimation $\bar{\theta}^{r+1} = \frac{1}{N}\sum_{i=1}^{N} \theta_i^r$. Here, the mean estimation error is as follows:

$$\mathbb{E}\left[\left\|\bar{\theta^*}^{r+1} - \hat{\bar{\theta}}^{r+1}\right\|_2^2\right] = \sum_{i=1}^{d} \mathbb{E}\left[\left(\bar{\theta^*}^{r+1} - \bar{\theta}^{r+1}\right)^2\right] \tag{11}$$

$$= \sum_{i=1}^{d} \mathbb{E}\left[\left(\frac{1}{N} \sum_{i \in \mathcal{N}} \left({\theta^*}_i^r - \theta_i^r\right)\right)^2\right] \tag{12}$$

$$= \frac{1}{N^2} \sum_{i=1}^{d} \mathbb{E}\left[\left(\sum_{i \in \mathcal{N}} \left({\theta^*}_i^r - \theta_i^r\right)\right)^2\right] \tag{13}$$

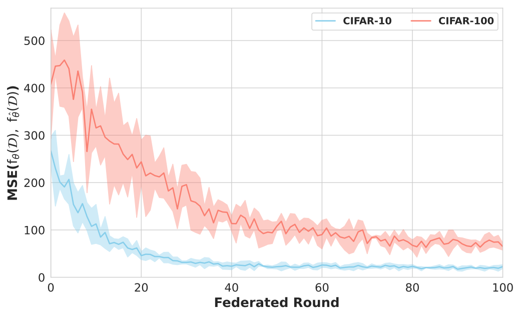
$$= \frac{1}{N^2} \sum_{i=1}^{d} \sum_{i \in \mathcal{N}} \mathbb{E}\left[\left({\theta^*}_i^r - \theta_i^r\right)^2\right] \tag{14}$$

$$\geq \frac{d\left((1-2^{-\mathrm{bpe}}) \cdot \bar{D}_{intra} + 2^{-\mathrm{bpe}} \cdot \bar{D}_{inter}\right)}{K} \tag{15}$$

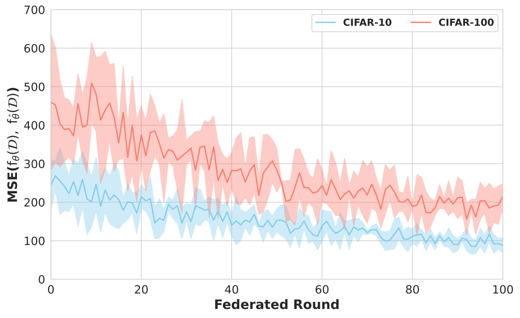
Here, $\bar{D}_{intra}$ and $\bar{D}_{inter}$ refer to the mean intra-cluster and inter-cluster distance across clients. Since server does not have access to clients' centroids, direct re-construction of clients' weights remains infeasible.

D Weight Clustering Privacy Leakage Analysis

We now evaluate the ability of server to re-construct individual clients weights by performing an cluster inference attack as presented in Inference Attacks I. For this, we performed experiments, where we measured the similarity between the "*perfectly*" estimated client weights (e.g. replacing each entry in θ_n with their closest value in $C_{\mathcal{A}}$), $\hat{\theta}_n$, and the true client's weights, θ_n, by measuring the similarity of the two models in the embeddings space. Specifically, we extract embeddings on the client's locally stored data using both $\hat{\theta}_n$ and θ_n, after which we perform a dimensionality reduction through PCA and measure the MSE error, similar to (52).



(a) IID settings



(b) Non-IID settings

Figure 4: Evaluation of Cluster Inference Attacks in `EncCluster`. We report MSE between the client's data embeddings extracted from client's true and estimated weights for both (a) IID and (b) non-IID settings.

In Figure 4, a pronounced contrast emerges between IID and non-IID settings. Specifically, in IID scenarios where there's a significant overlap between $C_{\mathcal{A}}$ and C_n, the attacker achieves notably lower MSE values, particularly in the later stages of training. Conversely, in non-IID settings, the effectiveness of the attack diminishes significantly, marked by high fluctuations across training rounds. Given the prevalence of highly non-IID conditions in most practical FL environments, Figure 4b underscores the potential for privacy breaches stemming from weight clustering in `EncCluster`. Nonetheless, it's critical to acknowledge that these findings are predicated on the assumption of a "*perfect*" estimation scenario, where θ_n entries are precisely matched with the nearest values in $C_{\mathcal{A}}$, necessitating $O(\kappa^d)$ complexity. This scenario underscores that, even when subjected to such potent attack strategies, the server's capacity to accurately reconstruct client models is limited, thereby offering a degree of protection against private data exposure.