MingledPie: A Cluster Mingling Approach for Mitigating Preference Profiling in CFL

Cheng Zhang, Yang Xu*, Jianghao Tan, Jiajie An, Wenqiang Jin
Hunan University
(zhangchengcs, xuyangcs, tanjianghao, anjiajie7, wqjin)@hnu.edu.cn

Abstract*—Clustered federated learning (CFL) serves as a promising framework to address the challenges of non-IID (non-Independent and Identically Distributed) data and heterogeneity in federated learning. It involves grouping clients into clusters based on the similarity of their data distributions or model updates. However, classic CFL frameworks pose severe threats to clients' privacy since the honest-but-curious server can easily know the bias of clients' data distributions (its preferences). In this work, we propose a privacy-enhanced clustered federated learning framework, MingledPie, aiming to resist against servers' preference profiling capabilities by allowing clients to be grouped into multiple clusters spontaneously. Specifically, within a given cluster, we mingled two types of clients in which a major type of clients share similar data distributions while a small portion of them do not (false positive clients). Such that, the CFL server fails to link clients' data preferences based on their belonged cluster categories. To achieve this, we design an indistinguishable cluster identity generation approach to enable clients to form clusters with a certain proportion of false positive members without the assistance of a CFL server. Meanwhile, training with mingled false positive clients will inevitably degrade the performances of the cluster's global model. To rebuild an accurate cluster model, we represent the mingled cluster models as a system of linear equations consisting of the accurate models and solve it. Rigid theoretical analyses are conducted to evaluate the usability and security of the proposed designs. In addition, extensive evaluations of MingledPie on six open-sourced datasets show that it defends against preference profiling attacks with an accuracy of 69.4% on average. Besides, the model accuracy loss is limited to between 0.02% and 3.00%.

I. Introduction

Federated Learning (FL) is a distributed machine learning paradigm that enables multiple clients to collaboratively train a machine learning model under the coordination of a server. In each iteration, each client trains a local model using its private dataset and shares only the local updates with the server. The server aggregates all local model updates to update the global model [1]. However, a major challenge of FL in practice is the clients' data statistical heterogeneity [2], [3], [4]. Since data is generated by each client, the distribution varies depending on the clients' personal preferences. Localized training on heterogeneity data results in a local optimum bias, where the client's model updates drift further away from each other [5]. Aggregating these model updates could counteract each other and result in poor performances [6]. Clustered federated learning (CFL) frameworks [7], [8], [9], [10], [11], [12] are proposed to conquer the data heterogeneity in FL, in which clients with similar data distributions are formed as one cluster to perform the FL training within the cluster. Eventually, CFL trains a global model for each cluster, separately.

However, CFL frameworks naturally enable the model aggregation server to profile clients' data distribution biases. For example, in an image classification task, the CFL server publishes a group of training clusters with specific training purposes of recognizing images with different labels, i.e., cars, buildings, foods, and etc. The clients whose data distribution is biased to the car images are grouped and joined to one CFL cluster. Thus, the honest-but-curious CFL server can easily infer that all the clients within the same training cluster have preferences for the cars.

Prior FL efforts have been made to protect clients' data preferences from the model aggregation server by leveraging the anonymous communication systems [13], [14], homographic encryption [15], [16], and trust executed environment [17], [18]. However, these designs cannot be applied to the CFL framework. In particular, some methods based on anonymous communication systems [13], [14], [19] try to break the relationship between client identity and cluster. But iterative model update communication makes client anonymity vulnerable to traffic analysis attacks [20], [21], [22], [23]. Other defenses are based on some secure aggregation techniques [24], [25] to prevent the server from inferring clients' data distribution biases from their model parameters. However, the server can bypass these defenses and analyze preferences based on client cluster identity.

In this work, we propose MingledPie, a privacy-preserving CFL framework that breaks the strong connections between the clients and training clusters by allowing clients to join multiple clusters spontaneously. As a result, clients uniquely belonged to one specific cluster and are now all mingled together, such that the CFL server cannot directly infer clients' preferences based on their belonged clusters. Though the core idea is straightforward, such a design leaves the CFL malfunctioning as the mingled clients will significantly lower the performances of the trained cluster models. Recall that, the original CFL cluster only groups clients with similar

*Yang Xu is the corresponding author.

Network and Distributed System Security (NDSS) Symposium 2025
24-28 February 2025, San Diego, CA, USA
ISBN 979-8-9894372-8-3
https://dx.doi.org/10.14722/ndss.2025.240195
www.ndss-symposium.org

data distributions and thus achieves high performances in a specialized training task. To conquer this challenge, we assume that every other cluster has a certain number of false positive clients obfuscated into the given cluster. The mingled cluster model is then treated as a composition of all accurate cluster models weighted by the number of false positive clients. This formulation leads to a solvable linear system of equations with accurate cluster models as variables. We leverage homomorphic encryption to privately aggregate the actual count of false positive clients, enabling clients to solve the linear system and rebuild the accurate global cluster models. Furthermore, to ensure privacy protection and usability of the model rebuilding method, we aim to maintain the proportion of false positive clients in each cluster close to a fixed ratio p. To achieve this without assistance from the CFL server, we take inspiration from public-key message detection designs [26]. Specifically, by mapping the client's target cluster address and the public keys of the other clusters to binary bits bitwise and comparing them, respectively, they should indicate a false-positive match with a probability of approximately p. From the server side, every client from other clusters has a probability p of joining this cluster, resulting in a fixed ratio of false positive clients.

The contributions of this paper are summarized as follows.

- We propose a privacy-enhanced CFL framework, MingledPie, which effectively protects client preference privacy while maintaining the accuracy of the CFL task.
- We propose an indistinguishable model update mingling method without additional communication infrastructure, to mingle the client's cluster identity, and devise an optimization method to rebuild cluster models.
- Theoretical analysis proves the correctness and security of the framework. Extensive experiments show that MingledPie can effectively protect privacy with almost negligible accuracy loss [1].

II. Related Work

In the following, we discuss the current CFL frameworks, and review the privacy inference attacks and defenses in FL. The differences between existing defenses and our approach are highlighted in Table I.

A. Clustered Federated Learning

One of the main challenges in traditional FL is statistical heterogeneity, which causes it to fail to produce good generalization models. Recently, CFL has been proposed to circumvent this issue by clustering clients into clusters based on the similarity of their data distributions or model updates, and training personalized cluster models for each cluster.

Most CFL frameworks adopt the client clustering strategy. Sattler *et al.* [28] presented the first CFL framework FedCluster, which recursively separates the two groups of clients with incongruent descent directions based on the cosine similarity of gradients. In order to achieve better efficiency and usability, some studies have optimized the clustering computation cost and clustering indicators of CFL [29], [28], [30], [31]. For instance, FedSEM [32] uses a l_2 distance-based stochastic expectation maximization instead of the distance-based neighborhood methods to optimize the cluster assignments. Besides, researchers in [33], [34], [35], [36] identified that efficient CFL could be achieved by only one-shot clustering based on clients' latest computed gradients when the global model stabilizes. There are also CFL frameworks that use the client clustering strategy, where the client estimates the closest cluster and joins in [7], [37], [38], [39]. Ghosh *et al.* [7] proposed the Iterative Federated Clustering Algorithm (IFCA), in which clients select the one with the lowest loss on their dataset from all the cluster models in each round and train the local model on it. On this basis, Ruan *et al.* [38] and Li *et al.* [37] respectively proposed soft clustering strategies on the client side, in which they made full use of the diversity of client data in different clusters to obtain higher model accuracy.

Nonetheless, the servers in all of the above CFL frameworks are able to get the client's cluster identity, because they rely on the server clustering clients or aggregating cluster models., which may result in preference profiling attacks.

B. Privacy-Preserving Federated Learning

Privacy Attacks. Although clients in FL do not need to share their private data, their private information can still be leaked by inference attacks. Specifically, in a member inference attack [40], [41], the attacker inferences whether a particular data is in the client's dataset. In a property inference attack [42], [43], the attacker inferences whether the target attribute is in the client's data set. In a model inversion attack [44], [45], the attacker attempts to get the data of the training dataset from the model. In a preference profile attack [46], [47], the attacker inferences the preferences of the client. The above attacks are all based on the client's local model parameters. In this work, we expose a new preference inference attack in CFL, which does not rely on model parameters and only needs to know the cluster preferences and the cluster identity of the client to infer the preferences of the client.

Preserving model privacy. The widely used techniques to protect model parameters from inference in FL include homomorphic encryption [15], [16], [48], [49], TEE [17], [18], [50], multi-party computation [51], [52], differential privacy [27], [53], etc. For instance, Zhang *et al.* [16] proposed an efficient homomorphic encryption-based FL framework BatchCrypt. By encoding gradients into long integers and aggregating them in the ciphertext domain, this method effectively prevents inference attacks on servers and saves up to 99% cost compared with classical homomorphic encryption-based schemes. Rieger *et al.* [18] proposed CrowdGuard, which protects the original performance of the model while resisting privacy inference attacks and adaptive attacks by deploying TEE on the client and server and aggregating the global model within the TEE. In addition, some studies [27], [53] introduced differential privacy to protect clients from the privacy threats of gradient exposure, but these schemes still sustain a certain degree of potential privacy disclosure. Although these efforts effectively

[1]Code is available at: https://github.com/CHENGZ03/MingledPie.

TABLE I: A comparison of different privacy-preserving FL schemes, for each represents a different privacy protection way. Property: a brief description of the privacy protection basis, Dependency: techniques or parameters on which the scheme's privacy depends. Resist on traffic analysis only counts when the scheme satisfies the client identity privacy.

Scheme	Property	Dependency	Model Privacy	Client Identity Privacy	Resist on Traffic Analysis	Resist on Preference Profiling Attack
BatchCrypto [16]	Homomorphic Encryption	Crypto Technique	●	○	-	○
CrowdGuard [18]	Trust Executed Environment	Hardware	●	○	-	○
DP-PFL [27]	Differential Privacy	DP Budget	●	○	-	○
FedTor [13]	Anonymous Communication	Tor Network	○	◐	○	◐
Ours	Mingling	False Positive Rate	●	●	●	●

○: do not have the attribute. ◐: holds under weak attackers, but not under strong attackers such as traffic analysis. ●: have the attribute.

defend against existing inference attacks, they cannot defend against preference profiling attacks in CFL because this attack does not depend on model parameters.

Preserving client identity privacy. Some FL frameworks protect privacy by breaking the link between the client's identity and its model update messages, making it impossible for the server to associate inferred privacy information with the client. Wang *et al.* [54] implements anonymity for cluster membership based on ring signatures, but it does not protect the communication metadata of model update messages. There are also FL frameworks that use Mix-net-based anonymous communication systems, such as FedTor [13] (use onion encryption to forward model update messages), shuffle-based FL [14], [55] (use third-party shuffling to mix model update messages). However, these schemes rely on additional communication infrastructure, and they can be insufficient against determined adversaries capable of observing network traffic [21], [23].

III. Problem Setting

In this section, we first describe the system model as well as the adversary model in our clustered federated learning, next clarify the design goals and challenges.

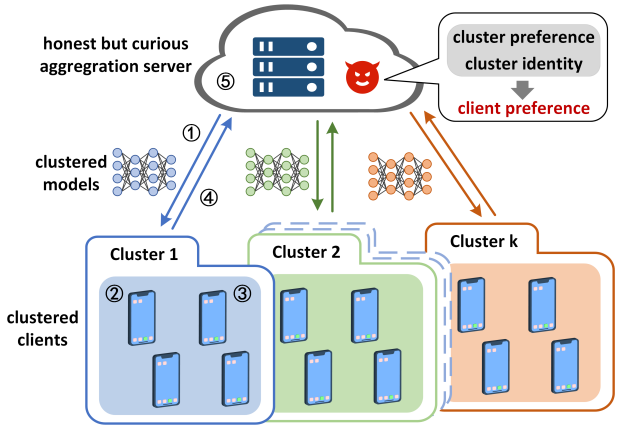


Fig. 1: Workflow of MingledPie

A. System Setting

For training the personalized cluster models, clients are grouped into clusters according to the similarity of their data distributions. Each client trains a local model comprising both shared and personalized layers, which is then transmitted to the server. The server uses FedAvg [1] to aggregate the shared layers from all local models into the shared layers of the cluster model, while also aggregating the personalized layers within each cluster to construct the personalized layers of the cluster model. To emphasize the key point, we omit the part where the server aggregates the shared layers and primarily focus on the training of the personalized part.

Specifically, the clients' dataset D has k different data distributions $\{D_j\}_{j=1}^{k}$. Accordingly, all clients $\{C_i\}_{i=1}^{N}$ are divided into k disjoint clusters $\{{G_j}^*\}_{j=1}^{k}$, with clients in each cluster having the similar data distributions. Since the data distribution is private to the clients, the real cluster identity of each client should not be disclosed to the server. Without loss of generality, the server initializes the cluster model for each cluster based on its prior knowledge and publishes the cluster address to receive model update messages. We define the meaningful description of the skew in the data distribution D_j as the preference of cluster j. Let $f(\theta; x, y)$ be the loss function associated with data point (x, y) in the dataset. The objective of cluster j is to minimize the loss function $F_j(\theta) = \mathbb{E}_{z\sim\mathcal{D}_j}[f(\theta; x, y)],\ j \in [k]$. To achieve this in practice, the CFL system needs to find clusters $\{\hat{G}_j\}_{j=1}^{k}$ that are close to $\{{G_j}^*\}_{j=1}^{k}$, and then have the clients in the $\{\hat{G}_j\}_{j=1}^{k}$ cooperate to find the optimal cluster models $\{\hat{\theta}_j\}_{j=1}^{k}$ that are close to $\arg\min F_j(\theta), j \in [k]$.

Workflow: As shown in Fig. 1, MingledPie follows the common client-side clustering CFL workflow. In the initialization phase, the server initializes the cluster model for each cluster and sets the hyperparameters for training and client mingling. In each round of iterative training, the client first synchronizes all the cluster models (step ①). The client then uses these cluster models to estimate the cluster identity (step ②) and trains the local model on the corresponding cluster model (step ③). The client then sends its local model and cluster identity to the server (step ④). Finally, the server aggregates new cluster models through FedAvg [1] (step ⑤).

B. Adversary Model

We consider the adversary is the honest-but-curious CFL server. It aims to infer the client's preference. We define client

local data preference as the characteristics of its data distribution, manifested as an imbalance in categories or features. For instance, in an image classification task, if a client's local dataset contains significantly more images labeled as cat and dog compared to other labels, it indicates a preference for these labels. Two types of preference profiling attacks could be adopted by the server.

A1: Cluster Identity-based Inference Attack. The server infers client preferences based on the preferences of the cluster to which the client belongs. Consistent with the assumptions of most classic CFL, the classic CFL server setup groups of learning tasks and form different CFL clusters to admit and join clients who having the corresponding data-preference of the learning task for training the cluster model. In this case, the server has a prior knowledge of the required data distribution for each cluster and is aware of the cluster to which each client's local model belongs. As client's data distribution requirements, the server can naturally associate clients with the preferences of their respective clusters, and infer client preferences.

A2: Local Model-based Inference Attack. The server infers preferences from clients' submitted local models prapameters during CFL model aggregation. It's detail has been revealed in several existing works [46], [56], [57]. For instance, an adversary can infer client preferences by retraining local models on auxiliary datasets with varying data distributions and analyzing gradient shifts. This type of attack allows the adversary to infer the distribution of the Top-k labels, or even the full label distribution of the client.

C. Design Goals and Challenges

The design goal of this paper is to protect the client's preference privacy during the CFL training process, and the security goals can be summarized as follows:

R1: Preference Privacy. The primary goal of MingledPie is to defend against preference profiling attacks. Specifically, the curious-but-honest server could not infer clients' true cluster identities, local model parameters, and its data preference.

R2: Cluster Model Accuracy. The proposed defense designs should not break the usability of CFL. The aggregated models should have similar accuracy performances to those models trained without involving the MingledPie's preference protection algorithms.

R3: Autonomous Deployment. The defense should be able to run fully autonomous without any additional third parties or infrastructures.

To the best of our knowledge, existing defense mechanisms, which primarily focus on hiding client identities [13] or protecting local models [16], [18], are designed for traditional FL. However, these approaches are ineffective in CFL, as they either fail to defend against attacks based on cluster identity and local models simultaneously, or conflict with the clustering algorithms used in CFL. Therefore, MingledPie addresses the following challenges:

C1: How to conceal the cluster identity of clients? In particular, the server should be allowed to access the client's cluster identity or local model, as this would enable cluster-based or model-based inference attacks.

C2: How to accurately aggregate cluster models without knowing clients' local model parameters and the cluster identities? To ensure the accuracy of the cluster models, the server must precisely aggregate local models belonging to the same cluster. However, existing CFL methods rely on cluster identities or local models for clustering and aggregation, which conflicts with our need to protect preference privacy.

C3: How to develop the privacy protection design that conquers **C1** and **C2** without the assistance of an additional third-party?

IV. MingledPie

In the following, we outline the high-level idea and details of MingledPie, and analyze the computation and communication overhead of each component.

A. High-level Overview

The high-level overview of MingledPie is shown in Fig. 2. Our approach involves clients sending their model updates to multiple clusters. From the server's perspective, each cluster contains correct models and false positive models belonging to other clusters, making it impossible for the server to accurately infer the client's cluster identity. Meanwhile, we use a secure aggregation method based on homomorphic encryption to protect model parameters. These mingled model updates within the clusters are aggregated indiscriminately into mingled cluster models. By carefully controlling the proportion of mingled models in each cluster, we propose an algorithm to rebuild accurate cluster models from the mingled cluster models, ensuring that the cluster models maintain high accuracy performances. MingledPie includes three critical components:

Indistinguishable cluster identity generation. The client identifies its true cluster by selecting the model with the lowest loss function, and then generates some mingled cluster identities based on a predefined false positive rate. The average number of cluster identities is proportional to the false positive rate relative to the total number of clusters.

Mingled cluster model aggregation. The server constructs mingled clusters based on the set of cluster identities provided by each client. These mingled clusters include both the true clients and false positive clients, and the server aggregates all local models within the mingled clusters to form the mingled cluster models. As a result, the mingled cluster model can be viewed as a combination of the accurate cluster model and a certain proportion of models from other clusters.

Cluster model rebuilding. The client synchronizes the mingled cluster models and calculates the number of clients from other clusters within each mingled cluster. This allows the client to construct a non-homogeneous system of linear equations composed of the accurate cluster models, the actual false positive rate, and the mingled cluster models. By solving this system, the client can rebuild the accurate cluster models.

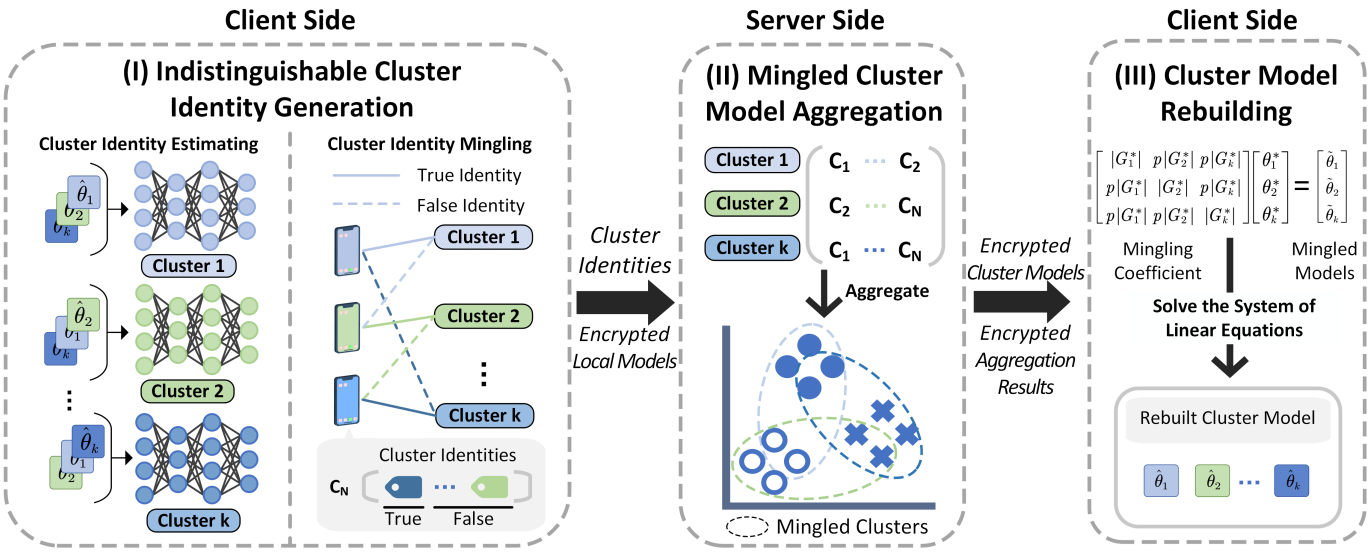


Fig. 2: Illustration of MingledPie in each iteration

B. Detail Design of MingledPie

We describe the algorithm used to implement our proposed framework as follows.

(I) Indistinguishable Cluster Identity Generation. The key to protecting client preferences in CFL is to break the link between client identities and clusters. As shown in Algorithm 1, we mingle a client's true cluster identity by generating multiple cluster identities for its model updates. We also make the false model updates in each cluster close to a false positive rate p, which allows us to rebuild cluster models.

Algorithm 1 Cluster Identity Generation

Input: cluster addresses $\{dsk_j\}_{j=1}^k$, cluster public keys $\{dpk_j\}_{j=1}^k$, privacy threshold T, cluster models $\{\theta_j^t\}_{j=1}^k$
Output: client i's cluster identity set CI_i, true cluster identity $\hat{j}_i$, encrypted cluster identifiers $\{HE(\sigma_{i,j})\}_{j=1}^k$
Client estimates cluster identity $\hat{j}_i = argmin_{j\in[k]} F_i(\theta_j^t)$
while $|CI_i| < T$ **do**
  $CI_i \leftarrow \{\}$, $r \xleftarrow{\$} \mathbb{Z}_q$, $u \leftarrow g^r$
  for each cluster j **do**
    if $H((h_{\hat{j}}^l)^r) \equiv H(u^{x_j^l}), \forall l \in [n]$ **then**
      $CI_i \leftarrow CI_i \cup j$
    else
      continue
    end if
  end for
end while
$\{HE(\sigma_{i,j})\}_{j=1}^k \leftarrow \sigma_{i,\hat{j}} = 1$, others 0
return $CI_i, \hat{j}_i, \{HE(\sigma_{i,j})\}_{j=1}^k$

Before the first round of training begins, the CFL server sets some system parameters. Specifically, the server generates a cluster public key dpk_j and a cluster address dsk_j for each cluster as follows:

$$dsk_j = \{x_j^1, ..., x_j^n\}, dpk_j = \{g, h_j^1, ..., h_j^n\}, j \in [k] \quad (1)$$

where g is a generator of a cyclic group G of prime order q, $x_l \in \mathbb{Z}_q$, and $h_j^l = g^{x_j^l}$ for $l \in [n]$. The length of the cluster public key and the cluster address is associated with a false positive rate $p = 2^{-n}$. In addition, the server sets a privacy threshold T, which means that the client needs to generate at least T cluster identities. The server then publishes the cluster addresses $\{dsk_j\}_{j=1}^k$, the cluster public keys $\{dpk_j\}_{j=1}^k$ and the initialized cluster model $\{\theta_j^0\}_{j=1}^k$ to all clients.

Algorithm 1 outlines the workflow of cluster identity generation for each client. In each round t, the client i estimates its true cluster identity $\hat{j}_i$ via finding the cluster model $\theta_{\hat{j}}^t$ with the lowest loss as follows:

$$\hat{j}_i = argmin_{j\in[k]} F_i(\theta_j^t) \quad (2)$$

If the client's cluster identity has not changed from before, it can use the previously generated cluster identity set. Otherwise, the client then generates a private random number r and computes $u \leftarrow g^r$. The r and u are used to prevent clients with the same $\hat{j}$ from generating the same cluster identity set. If $H : G \rightarrow \{0, 1\}$ is modeled as a random oracle, the client then determines whether it belongs to cluster j according to the following equation:

$$H((h_{\hat{j}}^l)^r) \equiv H(u^{x_j^l}), \forall l \in [n] \quad (3)$$

Once all clusters have been checked, the client starts checking whether the cluster identities for the current round protect its privacy. If the number of cluster identities in $|CI_i|$ does not reach the privacy threshold T, the client will choose a new random number to regenerate the cluster identities until the privacy requirement is met.

Subsequently, the client i trains its local model based on the cluster model $\theta_{\hat{j}}^t$ and encrypts its local model update as $HE(\theta_i^t)$ using a homomorphic encryption algorithm. Additionally, client i also generates encrypted cluster identifiers $\{HE(\sigma_{i,j})\}_{j=1}^k$ based on its true cluster identity $\hat{j}$ as follows:

$$\{HE(\sigma_{i,j})\}_{j=1}^k = \{HE(\sigma_{\hat{j}} = 1, \sigma_{j \neq \hat{j}} = 0)\}_{j=1}^k \quad (4)$$

where $\sigma_{\hat{j}}$ equals to 1 and others are 0. These cluster identifiers will be used to count the number of clients from each cluster within the mingled cluster for use in cluster model rebuilding.

Finally, the client sends the encrypted local model parameters $HE(\theta_i^t)$, the encrypted cluster identifier$\{HE(\sigma_{i,j})\}_{j=1}^k$, and the cluster identity set CI_i to the server. Without any additional information, the cluster identities in CI_i are indistinguishable from the server, and the client's local model is assigned to the clusters in CI_i.

During the training process, the false positive rate p is fixed, which makes it easy for us to formalize the privacy-preserving algorithm. In fact, p can be adjusted dynamically to fulfill different privacy-utility considerations. The server can adjust p at any time by generating different lengths of cluster public keys and cluster addresses.

(II) Mingled Cluster Model Aggregation. Once all clients have sent their model update messages to the server, the server begins constructing mingled clusters based on the client identity sets $\{CI_i\}_{i=1}^N$ and aggregates both the mingled cluster models and their identifiers for each cluster.

Specifically, the server first assigns the clients' model updates to different clusters based on their cluster identities. After mingling, each cluster contains all the local models that truly belong to that cluster, along with some local models from other clusters. We define the indicator function $\mathbb{I}(j \in CI_i)$, which is equal to 1 if the element j is present in the set CI_i, and 0 otherwise. For the j-th mingled cluster G_j, the server aggregates the mingled cluster model $\tilde{\theta}_j^{t+1}$ within the homomorphic encryption domain as follows:

$$HE(|G_j| \cdot \tilde{\theta}_j^{t+1}) = \sum_{i=1}^{N} HE(\theta_i^t) \cdot \mathbb{I}(j \in CI_i), j \in [k] \quad (5)$$

The server should also aggregate cluster identifiers for each mingled cluster. Referring back to the cluster identity generation, each client submits k encrypted cluster identifiers to multiple clusters, with a value of 1 for the identifier corresponding to its true cluster and 0 otherwise. Therefore, by summing the k cluster identifiers within a mingled cluster, the server can determine the number of true clients in that cluster, as well as the number of false positive clients assigned to each other cluster. The encrypted b-th cluster identifier for the a-th mingled cluster is computed as follows:

$$HE(x_{a,b}) = \sum_{i=1}^{N} HE(\sigma_{i,b}) \cdot \mathbb{I}(a \in CI_i), a \in [k] \quad (6)$$

These cluster identifiers form a mingling coefficient matrix $H_{k\times k} = \{x_{a,b}\}, (a, b = 1, 2, ..., k)$, where $x_{a,b}$ denotes the number of clients from cluster G_b^* within mingled cluster G_a. We represent the aggregation result of these cluster identifiers as $HE(H_{k\times k})$.

After finishing the mingled cluster model aggregation, the server sends the encrypted mingled cluster models $\{HE(|G_j| \cdot \tilde{\theta}_j^{t+1})\}_{j=1}^k$, along with the sizes of the mingled clusters $\{|G_j|\}_{j=1}^k$ and the encrypted mingling coefficient matrix $HE(H_{k\times k})$ to all clients.

(III) Cluster Model Rebuilding. Since mingled cluster models cannot be used directly, the client needs to rebuild the accurate cluster model before the next round of training. Algorithm 2 shows the cluster model rebuilding algorithm.

We assume that in the mingled cluster, local models from other clusters are uniformly sampled. Therefore, the mingled cluster model $\tilde{\theta}_j^{t+1}$ can be expressed as follows:

$$|G_j| \cdot \tilde{\theta}_j^{t+1} = |G_j^*| \cdot {\theta^*}_j^{t+1} + \sum_{c=1, c\neq j}^{k} x_{j,c} \cdot {\theta^*}_c^{t+1} \quad (7)$$

where $\{{\theta^*}_j^{t+1}\}_{j=1}^k$ are the accurate cluster models and $\{x_{j,c}\}_{c=1}^k$ are the number of local models in cluster G_c^* that are mingled with cluster G_j.

Based on Eq. 7, we can combine the expressions of all mingled cluster models to form a system of linear equations with the accurate cluster models as the unknowns. This system can be represented as follows:

$$H_{k\times k} {\theta^*}_{k\times 1} = \tilde{\theta}_{k\times 1} \quad (8)$$

where $H_{k\times k}$ is the mingling coefficient matrix computed according to Eq. 6, $\tilde{\theta}_{k\times 1} = \{\tilde{\theta}_1, ..., \tilde{\theta}_k\}$ is the mingled cluster model matrix and ${\theta^*}_{k\times 1} = \{{\theta^*}_1, ..., {\theta^*}_k\}$ is the accurate cluster model matrix. The solution of this system of linear equations can be easily obtained by Gaussian elimination, etc.

Algorithm 2 Cluster Model Rebuilding

1: **Input:** mingled cluster sizes $\{|G_j|\}_{j=1}^k$, encrypted mingling coefficient matrix $HE(H_{k\times k})$, and encrypted aggregation results $\{HE(|G_j|\tilde{\theta}_j^{t+1})\}_{j=1}^N$.
2: **Output:** rebuilt cluster models $\{\hat{\theta}_j^{t+1}\}_{j=1}^k$
3: **for** each client i **do**
4: **for** each cluster j **do**
5: Decrypt $HE(|G_j| \cdot \tilde{\theta}_j^{t+1})$ and $HE(H_{k\times k})$
6: $\tilde{\theta}_j^{t+1} \leftarrow |G_j| \cdot \tilde{\theta}_j^{t+1} / |G_j|$
7: **end for**
8: $\theta_{k\times 1} \leftarrow \{\tilde{\theta}_j^{t+1}\}_{j=1}^k$
9: $\{\hat{\theta}_j^{t+1}\}_{j=1}^k \leftarrow$ solve $H_{k\times k}\tilde{\theta}_{k\times 1} = \theta_{k\times 1}$
10: **end for**
11: **return** $\{\hat{\theta}_j^{t+1}\}_{j=1}^k$

C. Complete MingledPie Algorithm

Algorithm 3 shows the complete MingledPie algorithm. In each iteration, it mainly consists of the following five steps: 1) The server sends the encrypted mingled cluster model and cluster identifiers to the client (line 7). 2) The client then uses the cluster model rebuilding algorithm to compute the

Algorithm 3 MingledPie Algorithm

1: **Input:** number of clusters k, number of clients N, learning rate γ, number of local gradient steps τ
2: **Output:** cluster models $\{\tilde{\theta}_j\}_{j=1}^{k}$
3: Server: Initialization
4: $\{\theta_j^0\}_{j=1}^{k} \leftarrow$ Initialize cluster models.
5: **for** each training iteration t **do**
6: **for** each client i **do**
7: Synchronizing model updates from server.
8: **if** $t \neq 0$ **then**
9: Clients: Cluster Model Rebuilding
10: $\{\hat{\theta}_j^t\}_{j=1}^{k} \leftarrow$ Algorithm 2
11: **end if**
12: Clients: Indistinguishable Cluster Identity Generation
13: $CI_i, \hat{j}, \{HE(\sigma_{i,j})\}_{j=1}^{k} \leftarrow$ Algorithm 1
14: $\theta_i^{t+1} \leftarrow LocalUpdate(\theta_{\hat{j}}^t, \gamma, \tau)$
15: Send CI_i, $HE(\theta_i^{t+1})$ and $\{HE(\sigma_{i,j})\}_{j=1}^{k}$ to server.
16: **end for**
17: Server: Mingled Cluster Model Aggregation
18: **for** each cluster j **do**
19: $HE(|G_j|\tilde{\theta}_j^{t+1}) \leftarrow \sum_{j \in CI_i} HE(\theta_i^t)$
20: $HE(x_{j,b}) = \sum_{i=1}^{N} HE(\sigma_{i,b}) \cdot \mathbb{I}(j \in CI_i)$
21: **end for**
22: $HE(H_{k\times k}) \leftarrow HE(x_{ab}), a, b \in [k]$
23: Send $\{HE(|G_j|\tilde{\theta}_j^{t+1})\}_{j=1}^{k}$, $HE(H_{k\times k})$ and $\{|G_j|\}_{j=1}^{k}$ to all clients.
24: **end for**
25:
26: **procedure** LOCALUPDATE($\theta_{\hat{j}}^t, \gamma, \tau$)
27: **for** each local epoch e in $[\tau]$ **do**
28: $\theta_i^{e+1} \leftarrow \theta_i^e - \gamma \nabla F_i(\theta_i^e)$
29: **end for**
30: **end procedure**

accurate cluster models (line 10), and uses the cluster identity generation algorithm to estimate its true cluster identity and the mingled cluster identity set (line 13). 3) The client trains the local model based on its true cluster identity (line 14). 4) The client sends the encrypted local model update and the cluster identifier to the server (line 15). 5) The server aggregates the encrypted local models and cluster identifiers for each mingled cluster (line 17-22).

Here we analyze the computation and communication overhead of MingledPie. Let $p_{correct}$ denote the actual false positive rate under the privacy threshold, which may be higher than p. It can be defined as follows:

$$p_{correct} = \frac{\sum_{c=T}^{k-1} cPr_c}{(k-1)\sum_{c=T}^{k-1} Pr_c} \tag{9}$$

where Pr_c is the probability that Algorithm 1 generates c cluster identities, it can be computed as follows:

$$Pr_c = C_{k-1}^{c-1} p^{c-1} (1-p)^{k-c} \tag{10}$$

Therefore, the computational complexity of cluster identity generation is $O(k/p_{correct})$, the model aggregation algorithm has a complexity of $O(kN(1+p_{correct}))$, and model rebuilding has a complexity of $O(k^3)$. The communication overhead of MingledPie mainly stems from the transmission of encrypted models, where the client's communication complexity is $O(k|HE(\theta)|)$ and the server's is $O(N|HE(\theta)|)$.

V. Theoretical Analysis

In this section, we conduct a rigid theoretical analysis of MingledPie from the perspectives of security, usability, and algorithm convergence.

A. Security Analysis

Theorem 1 (*Identity Indistinguishability*)**.** *The mingled identities generated by MingledPie are indistinguishable for all possible adversary $\mathcal{A}$.*

Proof. The information obtained by the attacker $\mathcal{A}$ from the client includes CI_i, $HE(\theta_i)$ and $HE(\sigma_{i,j})$, where $HE(\theta_i)$ and $HE(\sigma_{i,j})$ are encrypted. If the chosen homomorphic encryption scheme is CPA-secure, the two plaintexts are indistinguishable based on their ciphertexts, and $\mathcal{A}$ cannot extract any useful information from$HE(\theta_i)$ and $HE(\sigma_{i,j})$. The attacker can only attempt to infer the client's identity from the cluster identity set. According to Algorithm 1, upon receiving a client's cluster identity, $\mathcal{A}$ can know these cluster address matches the public key of the client's true cluster. However, due to the one-way nature of the hash function in Eq. 3 and the privacy of the random number r, $\mathcal{A}$ cannot deduce the true cluster identity from the cluster identity set. Furthermore, the server cannot analyze the client's true cluster identity from the variations in the identity set across multiple iterations, as the client only generates a new identity set when there is a change in the true cluster identity. This comparison is therefore meaningless. Therefore, the client's cluster identity remains indistinguishable.

Theorem 2 (*Bounds of the Preference Profiling Attacks*)**.** *If Theorem 1 holds, the probability Pr_{attack} of an adversary $\mathcal{A}$ could infer users' preferences is bounded as $1/k < Pr_{attack} < 1/T$.*

Proof. According to Theorem 1, the optimal strategy for an attacker attempting to infer a client's identity based on transmitted information is to guess randomly. The most effective guessing approach for the attacker is to select from the client's cluster identity set, yielding an accuracy of $1/|CI_i|$. Since $|CI_i \in [T, k]|$, the lower bound of the attack accuracy Pr_{attack} is $1/k$, and the upper bound is $1/T$.

B. Usability Analysis

For the cluster model rebuilding algorithm, we obtain accurate cluster models by solving a system of linear equations. Here, we prove that this system of linear equations is solvable and the solution is unique.

Theorem 3 (*Unique Solution in the Ideal Condition*)**.** *Given conditions that $p \in (0,1)$, $\{|G^*|_j\}_{j=1}^k > 0$ and $|G_j^*| \geq (N - |G_j^*|)p$, $H_{k\times k}^* {\theta^*}_{k\times 1} = \tilde{\theta}_{k\times 1}$ has a unique solution of ${\theta^*}_{k\times 1}$.*

Proof. The condition for the system of linear equations to have a unique solution is that $H_{k\times k}^*$ is invertible. In the ideal condition, the composition of the clusters strictly follows the false positive rate p with $|G| = |G^*| + (N - |G^*|)p.$, where the number of cluster identifiers from all clients is kp. The ideal coefficient matrix $H_{k\times k}^*$ can be represented as follows:

$$H_{k\times k}^* = \begin{bmatrix} |G_1^*| & p|G_2^*| & \cdots & p|G_k^*| \\ p|G_1^*| & |G_2^*| & \cdots & p|G_k^*| \\ \vdots & \vdots & \ddots & \vdots \\ p|G_1^*| & p|G_2^*| & \cdots & |G_k^*| \end{bmatrix} \tag{11}$$

Given $|G_j^*| \geq (N - |G_j^*|)p$ for each cluster G_j^*, we can further get that the diagonal elements of $H_{k\times k}^*$, denote as $x_{i,i}^*$, satisfy $x_{i,i}^* \geq \sum_{j=1, j\neq i}^k x_{ij}$. Therefore, $H_{k\times k}^*$ is a strictly diagonally dominant matrix. Through the row transformation for $det(H_{k\times k}^*)$, we can get:

$$det(H_{k\times k}^*) = (1+p(k-1))(1-p)^{k-1}|G_1^*||G_2^*|\cdots|G_k^*| \tag{12}$$

Since we require each cluster to be non-empty and the false positive rate $p \in (0,1)$, we can get $det(H_{k\times k}^*) \neq 0$ and the ideal coefficient matrix $H_{k\times k}^*$ is invertible.

Theorem 4 (*Pertubation of Coefficient Matrix*)**.** *If Theorem 3 holds, giving that $min\{|G_b^*|\}_{b=1}^k > (k-1)^2 + 105/16$, the perturbation of coefficient matrix $H_\delta = H_{k\times k} - H_{k\times k}^*$ is considered negligible.*

Proof. See Appendix A-A.

Theorem 5 (*Unique solution of the Rebuilding Model*)**.** *If Theorem 3 and 4 holds, $H_{k\times k}{\theta^*}_{k\times 1} = \tilde{\theta}_{k\times 1}$ of the real coefficient matrix $H_{k\times k}$ has a unique solution of ${\theta^*}_{k\times 1}$.*

Proof. $H_{k\times k}$ is a matrix that fluctuates around $H_{k\times k}^*$, which can be viewed as composed of the ideal matrix $H_{k\times k}^*$ and a perturbation matrix H_δ. Theorem 3 proves that the system with $H_{k\times k}^*$ has a unique solution under certain conditions. According to matrix perturbation theory, $H_{k\times k}^*$ remains invertible after adding a small perturbation H_δ. Theorem 4 proves that it is a small perturbation with high probability and is negligible. Therefore, $H_{k\times k}$ is invertible, and the system of linear equations has a unique solution.

C. Convergence Analysis

Following the prior works [7], [58], [59], we assume the loss function $F_j(\cdot)$ is λ-strongly convex and L-smooth, the variance of $f(\theta; z)$ is upper bounded by η^2, and the variance of $\nabla f(\theta; z)$ is upper bounded by v^2. The maximum norm of the theoretically optimal models is bounded: $\max_{j\in[k]} \|\overline{\theta}_j\| \lesssim 1$. The initial model parameter estimates satisfy $\|\hat{\theta}_j^0 - \overline{\theta}_j\| \leq \left(\frac{1}{2} - \alpha_0\right)\sqrt{\frac{\lambda}{L}}\Delta$ for all $j \in [k]$. The amount of data is such that $m \gtrsim \frac{k\eta^2}{\alpha_0^2\lambda^2\Delta^4}$.

Next, we introduce the following definitions: Let ξ denote the proportion of the cluster's size relative to that of the mingled cluster, defined as $\xi = \frac{|G_j^*|}{|G_j|}$. Define β as the proportion of this cluster's size relative to the mingled cluster for a different cluster j' (where $j' \neq j$), given by $\beta_{j'} = \frac{|G_{j'}^* \cap G_j|}{|G_j|}$. Let $\omega_j = \frac{|G_j^*|}{m}$ represent the fraction of clients belonging to the j-th cluster, and assume that $\omega = \min\{\omega_1, \omega_2, \ldots, \omega_k\}$ satisfies $\omega \gtrsim \frac{\log(mN)}{N}$, where m denotes the amount of data used. Define $\left\{\overline{\theta}_j\right\}_{j\in[k]}$ as the theoretically optimal model to be obtained. Let Δ represent the minimum distance between different cluster models, defined as $\Delta = \min_{j\neq j'} \|\theta_j - \theta_{j'}\|$, and the signal-to-noise ratio is given by $\rho = \frac{\Delta^2}{\sigma^2}$. Additionally, we require that

$$\Delta \geq \widetilde{\mathcal{O}}\left(\max\left\{\alpha_0^{-2/5} m^{-1/5}, \alpha_0^{-1/3} N^{-1/6} m^{-1/3}\right\}\right) \tag{13}$$

where the closeness parameter α_0 satisfies $0 < \alpha_0 < \frac{1}{2}$.

Theorem 6. *(The convergence guarantee of the entire algorithm). Given $\hat{\theta}_j^t$ as the t-th iteration in the algorithm and $\delta \in (0,1)$, we have for any fixed $j \in [k]$, with probability at least $1-\delta$, we can obtain $\|\hat{\theta}_j^t - \overline{\theta}_j\| \leq \varepsilon$,where*

$$\begin{aligned} \varepsilon \lesssim & \frac{vkL\log(mN)}{\omega^{5/2}\lambda^2\delta\sqrt{mN}} + \frac{\eta^2 L^2 k\log(mN)}{\omega^2\lambda^4\delta\Delta^4 m} \\ & + \frac{L\Delta}{\omega\lambda\xi}\sum_{j'\neq j}\beta_{j'} + \widetilde{\mathcal{O}}\left(\frac{1}{m\sqrt{N}}\right) \end{aligned} \tag{14}$$

Proof. We first prove the convergence of the algorithm in a single iteration, and then analyze the convergence of multiple iterations on this basis.

Let $\hat{\theta}_j^+$ be the next iterate in the algorithm, and θ_j^* be the accurate cluster model matrix. According to the triangle inequality, the error between the model of the next iteration and the optimal model can be expressed as:

$$\|\hat{\theta}_j^+ - \overline{\theta}_j\| \leq \|\hat{\theta}_j^+ - \theta_j^{*+}\| + \|\theta_j^{*+} - \overline{\theta}_j\| \tag{15}$$

Here, we analyze the bound of $\|\hat{\theta}_j^+ - \theta_j^{*+}\|$ and $\|\theta_j^{*+} - \overline{\theta}_j\|$ separately.

a) Bound $\|\hat{\theta}_j^+ - \theta_j^{*+}\|$: $\widetilde{\theta}_j^+$ aggregate all local models within the cluster and the local models from other clusters, that is,

$$\widetilde{\theta}_j^+ = \xi\theta_j^{*+} + \sum_{j'\neq j}\beta_{j'}\theta_{j'}^{*\prime} \tag{16}$$

where $\theta_{j'}^{*\prime}$ represents the aggregation of local models belonging to $G_{j'}^* \cap G_j$. In the cluster model rebuilding, we establish the following expression for solving:

$$\widetilde{\theta}_j^+ = \xi\hat{\theta}_j^+ + \sum_{j'\neq j}\beta_{j'}\hat{\theta}_{j'}^+ \tag{17}$$

We can get the following bound from Eq. 16 and Eq. 17:

$$\|\hat{\theta}_j^+ - \theta_j^{*+}\| \leq \frac{1}{\xi}\sum_{j'\neq j}\beta_{j'}\|\hat{\theta}_{j'}^+ - \theta_{j'}^{*\prime}\| \tag{18}$$

Since the cluster models between different clusters have minimum separations Δ, the difference between the models

aggregated from different clients within the same cluster must be smaller than Δ, that is $\|\hat{\theta}_{j'}^{+} - \theta_{j'}^{*\prime}\| \leq \Delta$. Therefore, we can change Eq. 18 and get the bound of $\|\hat{\theta}_{j}^{+} - \theta_{j}^{*+}\|$ as follows:

$$\|\hat{\theta}_{j}^{+} - \theta_{j}^{*+}\| \leq \frac{\Delta}{\xi} \sum_{j' \neq j} \beta_{j'} \tag{19}$$

b) Bound $\|\theta_{j}^{*+} - \overline{\theta}_{j}\|$: Following the proof in prior works [7], [60], we can obtain

$$\begin{aligned} \|\theta_{j}^{*+} - \overline{\theta}_{j}\| \leq & (1 - \frac{\omega\lambda}{8L})\|\hat{\theta}_{j} - \overline{\theta}_{j}\| + \frac{c_0 v}{\delta L\sqrt{\omega m N}} \\ & + \frac{c_1 \eta^2}{\delta\alpha^2\lambda^2\Delta^4 m} + \frac{c_2 v\eta k^{3/2}}{\delta^{3/2}\alpha\lambda L\Delta^2\sqrt{N}m} \end{aligned} \tag{20}$$

By substituting Eq. 19 and Eq. 20 into Eq. 21, given $\delta_0 \in (0, 1)$, with probability at least $1 - \delta_0$, we can derive:

$$\|\hat{\theta}_{j}^{+} - \overline{\theta}_{j}\| \leq (1 - \frac{\omega\lambda}{8L})\|\hat{\theta}_{j} - \overline{\theta}_{j}\| + \varepsilon_0 \tag{21}$$

where

$$\begin{aligned} \varepsilon_0 \lesssim & \frac{v}{\delta_0 L\sqrt{\omega m N}} + \frac{\eta^2}{\delta_0\alpha^2\lambda^2\Delta^4 m} \\ & + \frac{v\eta k^{3/2}}{\delta_0^{3/2}\alpha\lambda L\Delta^2\sqrt{N}m} + \frac{\Delta}{\xi} \sum_{j' \neq j} \beta_{j'} \end{aligned} \tag{22}$$

Following the proof in prior works [7], [60], we can conclude that after $t = \frac{8L}{\omega\lambda} \log\left(\frac{2\Delta}{\varepsilon}\right)$ iterations. To ensure that the cumulative failure probability for a single iteration is less than the overall failure probability δ of the entire algorithm, We set $\delta_0 = \frac{\omega\lambda\delta}{ckL\log(mN)}$. $\hat{\theta}_{j}^{t}$ is closest to the theoretically optimal model, that is,

$$\|\hat{\theta}_{j}^{t} - \overline{\theta}_{j}\| \leq \frac{16L}{\omega\lambda}\varepsilon_0 \tag{23}$$

Using Eq. 22 and Eq. 23, we can obtain Eq. 14, which completes the proof.

VI. Evaluation

In this section, we first present the experimental setting, and then evaluate our MingledPie against the preference profiling attack. Afterward, we evaluate the performance of our approach.

A. Experimental Setup

Testbed and Baselines: We implement MingledPie in Python 3.9 using the deep learning framework PyTorch 1.10.0. We deploy it on a server with Intel Xeon Gold 6430 CPU, NVIDIA GeForce RTX 4090 GPU, and 120GB RAM. To validate the performance of MingledPie, we evaluate it with three advanced FL methods: (a) IFCA [7] is a CFL method of client-side clustering. (b) FedProx [61] is an FL method that deals with statistical heterogeneity by regularization. (c) FedEM [62] is an FL method that deals with statistical heterogeneity by an expectation maximization algorithm. IFCA trains a personalized cluster model for each cluster, and FedProx and FedEM train a generalized global model for all clients.

Datasets: For our evaluation, we use the image datasets MNIST [63], Fashion MNIST [64], CIFAR-10, CIFAR-100 [65], and the medical dataset Texas100 [40], as well as the shopping records dataset Purchase100 [40]. MNIST and Fashion MNIST both contain 10 classes, with 60k training images and 10k test images. CIFAR-10 and CIFAR-100 contain 10 and 100 classes respectively, and both datasets have 50k training images and 10k test images. Texas100 and Purchase100 each contain 100 classes.

To simulate the setting of CFL, we divide all labels of the dataset into several label sets, where each cluster corresponds to a label set. To achieve non-IID scenarios, we randomly assign training data from the same label set to clients, ensuring that each client's label preferences align with the respective label set. Clients are then allocated to their corresponding clusters. In the homogeneous clustering scenario, we evenly distribute dataset labels so that each label set has the same size, and each cluster has an equal number of clients. In a heterogeneous clustering scenario, dataset labels can be partitioned arbitrarily, and cluster membership allocation is based on the proportion between the label sets.

Models: We use different model architectures for six datasets. For the MNIST and the Fashion MNIST, we use a fully connected neural network (FCNN) [66] with two fully connected layers and a single hidden layer of size 200. For the CIFAR-10, we adopt a convolutional neural network (CNN) [67] with two convolutional layers, a max-pooling layer, and three fully connected layers. For the CIFAR-100, we adopt the ResNet-18 model [68]. For the Texas100 and Purchase100, we use the multilayer perceptron model (MLP) [66].

Default Configurations: For all datasets, the number of clusters is set to 5, the number of clients to 120, the number of pre-training epochs to 1, the number of local iterations to 5, the false positive rate is 0.5, and the threshold is 2. For MNIST, Fashion MNIST, and CIFAR-100, the number of training epochs is set to 100. For CIFAR-10, the number of training epochs is set to 240. For Texas100 and Purchase100, the number of training epochs is set to 300. The learning rate is set to 0.01 for MNIST, Fashion MNIST, and Purchase100, and to 0.1 for the other datasets.

B. Defense Against Preference Profiling Attack

*1) **Defense Effects under Different Parameters**:* In the defense experiment of MingledPie, we assume that the adversary can analyze each client's traffic and obtain the cluster identity results and model updates sent out by each client from the analysis. For the model update results, the clients use HE to encrypt the ciphertext for transmission, and the adversary cannot decrypt the ciphertext to obtain the model information. Since our work focuses on cluster identity protection, the model information protection part can be replaced by any efficient HE scheme or other types of schemes that can protect model privacy. We focus on the attack that trying to obtain the client's cluster identity.

We conducted experiments under different parameters, i.e., the effect of the parameters of false positive rate, number

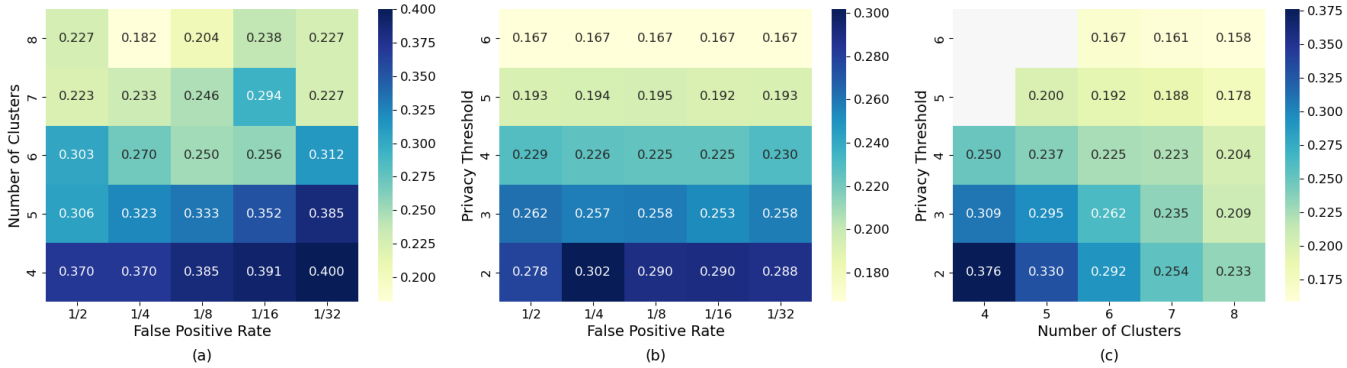


Fig. 3: Adversary preference profiling accuracy with default parameters $p = 0.5, k = 5$ and $T = 2$

of clusters, and cluster privacy threshold on the probability that an adversary will infer the true cluster identity of a client, i.e., preference profiling. As shown in Fig. 3, three heat maps demonstrate the impact of the above parameters on the adversary's preference profiling accuracy. Fig. 3 (a) fixes the privacy threshold $T = 2$, and the result shows that a lower false positive rate slowly increases the adversary's attack accuracy because the lower the false positive rate, the fewer additional false positive cluster identities are generated. For example, when we have 5 clusters and $p = 0.5$, the adversary has an accuracy of 33.3% to obtain the true identity of the client with an average of 3 mingled cluster identities. The result shows that the accuracy is 30.6%, so the defense accuracy is 69.4%, which is close to the ideal result. Fig. 3 (b) fixes the number of clusters $k = 5$, and it comes out that p has little effect on the results and T effects more on the results. The last experiment in Fig. 3 shows the impact of k and T with a fixed $p = 0.5$. There is a constraint that the threshold needs to be greater than the number of clusters, so that explains the grey part of the figure. The number of clusters significantly affects the adversary's attack accuracy because a larger number of clusters means that the adversary has a greater range of speculation and a greater impact following a larger threshold.

*2) **Compare with IFCA**:* MingledPie and IFCA use the same clustering and training methods. Therefore, IFCA can be considered as our approach without defense components. Here, we compare the accuracy of Top-1 preference profiling attacks on the server between IFCA and MingledPie. As shown in Table II, the privacy-preserving results of MingledPie remain stable at around 29% throughout multiple training rounds. In contrast, the accuracy of preference profiling attacks in IFCA increases rapidly with the number of training epochs. For the MNIST, Fashion MNIST, CIFAR-10, CIFAR-100, Texas100, and Purchase100 datasets, the attacker's average accuracy in the first five rounds of training on IFCA is 87.6%, 96.8%, 99.8%, 96.3%, 98.2%, and 95.8%, respectively. The primary reason is that, initially, the clustering results for clients may not be accurate, making the server's cluster-identity-based inference inaccurate. However, after a period of training, IFCA's clustering algorithm produces increasingly accurate clustering results, and cluster identity-based inference attacks become more accurate.

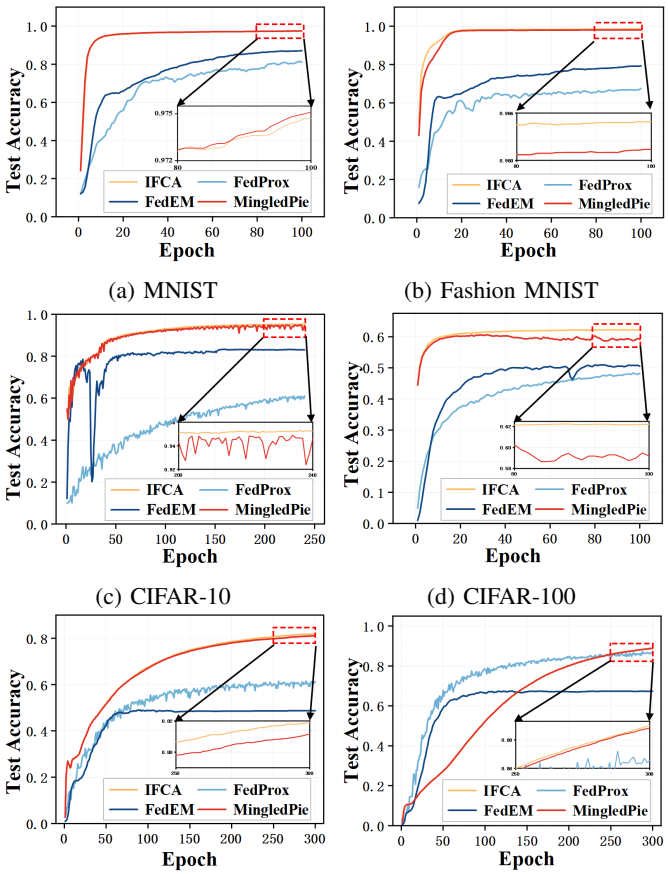
TABLE II: Comparison of preference profiling accuracy with default parameters $p = 0.5, T = 2, k = 5$, and $N = 120$

Dataset	Method	Epoch		
		1	3	5
MNIST	MingledPie	27.8%	27.8%	27.8%
	IFCA	60.3%	97.2%	100.0%
Fashion MNIST	MingledPie	28.8%	28.8%	28.8%
	IFCA	86.7%	100.0%	100.0%
CIFAR-10	MingledPie	29.0%	29.0%	29.0%
	IFCA	98.8%	100.0%	100.0%
CIFAR-100	MingledPie	28.9%	28.9%	28.9%
	IFCA	81.7%	100.0%	100.0%
Texas100	MingledPie	30.1%	30.1%	30.1%
	IFCA	90.8%	100.0%	100.0%
Purchase100	MingledPie	29.0%	29.0%	29.0%
	IFCA	79.2%	100.0%	100.0%

C. Training Performance

*1) **Average Test Accuracy**:* In this section, we evaluate performance using average test accuracy, primarily because MingledPie and IFCA are personalized federated learning approaches that produce personalized cluster models rather than a global model. Therefore, using average test accuracy allowed for a more straightforward comparison with the model accuracy of FedProx and FedEM.

First, we conduct experiments under homogeneous clustering conditions, initializing each cluster with the same number of clients. As shown in Fig. 4, we observe that the model accuracy of MingledPie and IFCA are significantly higher than those of FedProx and FedEM. The model accuracy of MingledPie and IFCA are very close, and their convergence processes are also very similar. According to TABLE III, with identical default parameters, on the MNIST, Fashion MNIST, CIFAR-10, CIFAR-100, Texas100, and Purchase100 datasets, the average model accuracy of MingledPie and IFCA differs



(a) MNIST (b) Fashion MNIST

(c) CIFAR-10 (d) CIFAR-100

(e) Texas100 (f) Purchase100

Fig. 4: Average cluster model convergence curve in homogeneous clustering setting

by only 0.02%, 0.34%, 0.51%, 3.00%, 0.74%, and 0.16%, respectively, indicating minimal disparity between the two approaches. From this, we can infer that the rebuilt model is similar to the cluster model without mingled aggregation. However, due to computational errors during model rebuilding, there is a slight difference in model accuracy.

Next, we conduct experimental analysis under the scenario of heterogeneous clustering. In the initialization phase, we randomly divide the data labels. In the MNIST, Fashion MNIST, and CIFAR-10 datasets, each containing 10 labels, we divide the labels into five label sets ranging from 1 to 3. In the CIFAR-100 dataset, which has 100 labels, we divide the labels into five label sets ranging from 10 to 30 in size. In the Texas100 and Purchase100 datasets, each with 100 labels, we divide the labels into ten label sets ranging from 5 to 20 in size. Then, we allocate clients based on the proportion between the label sets. The results, as shown in TABLE III, indicate that the model accuracy of MingledPie and IFCA is significantly higher than that of FedProx and FedEM. Additionally, MingledPie and IFCA have slight differences in model accuracy. It is worth noting that even if some heterogeneous settings do not satisfy the conditions in Theorem 5, our method can still rebuild an accurate cluster model. Therefore, whether

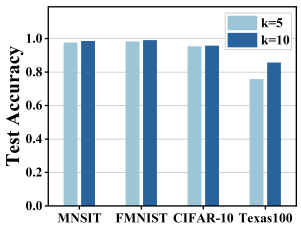
TABLE III: Average cluster model accuracy with default parameters $p = 0.5, T = 2, k = 5$, and $N = 120$

Dataset	Method	Accuracy	
		Homogeneous	Heterogeneous
MNIST	MingledPie	97.51%	95.81%
	IFCA	97.49%	95.86%
	FedProx	81.11%	81.11%
	FedEM	87.14%	87.14%
Fashion MNIST	MingledPie	98.14%	96.14%
	IFCA	98.49%	96.15%
	FedProx	67.53%	67.53%
	FedEM	79.33%	79.33%
CIFAR-10	MingledPie	94.76%	87.30%
	IFCA	95.27%	90.62%
	FedProx	61.11%	61.11%
	FedEM	83.10%	83.10%
CIFAR-100	MingledPie	59.20%	55.64%
	IFCA	62.20%	60.92%
	FedProx	48.22%	48.22%
	FedEM	50.55%	50.55%
Texas100	MingledPie	81.14%	75.15%
	IFCA	81.88%	76.39%
	FedProx	61.22%	61.22%
	FedEM	48.72%	48.72%
Purchase100	MingledPie	88.83%	77.72%
	IFCA	88.99%	80.96%
	FedProx	84.98%	84.98%
	FedEM	67.36%	67.36%

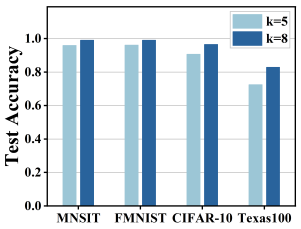
under homogeneous or heterogeneous clustering conditions, MingledPie can achieve privacy protection while maintaining high model accuracy.

*2) **Parameter Influence**:* In this part, we analyze the impact of parameter adjustments on the experimental results. We adjust the number of clusters k, the number of clients N, the false positive rate p, and the privacy threshold T, then test their experimental results under both homogeneous and heterogeneous clustering conditions for comparison.

Impact of the number of clusters: In the homogeneous clustering setting, we evaluate the impact of increasing the number of clusters from 5 to 10 on model accuracy. Specifically, for the MNIST and Fashion MNIST datasets, when the number of clusters is 10, we adjusted the learning rate due to insufficient samples in each cluster, and decreased it on the epoch-specific to prevent overfitting. Fig. 5 (a) shows the accuracy for each dataset under different cluster counts. We observe that the average model accuracy with 10 clusters is higher than with 5 clusters. For example, on the MNIST dataset, the accuracy is 97.51% with 5 clusters and 99.40% with 10 clusters. In the heterogeneous clustering setting, we increase the number of clusters from 5 to 8. Fig. 5 (b) shows the accuracy for each dataset with different numbers of clusters under heterogeneous clustering. We observe that the average model accuracy is higher with 8 clusters compared to 5 clusters. This improvement can be attributed to the fact that by increasing the number of clusters, clients can be grouped in a more fine-grained manner, allowing for better

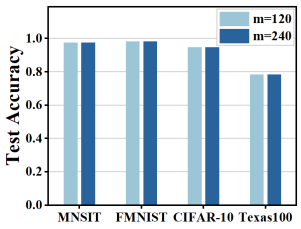


(a) Homogeneous Clustering

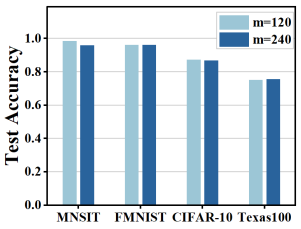


(b) Heterogeneous Clustering

Fig. 5: Impact of the number of clusters

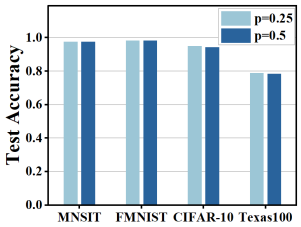


(a) Homogeneous Clustering

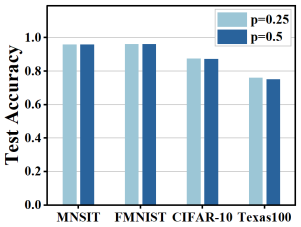


(b) Heterogeneous Clustering

Fig. 6: Impact of the total number of clients

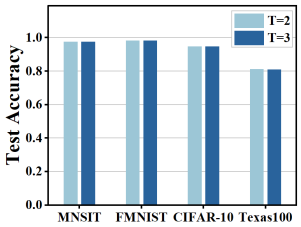


(a) Homogeneous Clustering

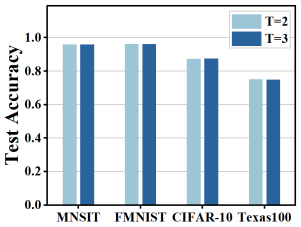


(b) Heterogeneous Clustering

Fig. 7: Impact of the false positive rate



(a) Homogeneous Clustering



(b) Heterogeneous Clustering

Fig. 8: Impact of the privacy threshold

model personalization to match the data distribution within each cluster. As a result, the model training process becomes more effective, leading to an overall improvement in model performance.

Impact of the total number of clients: To investigate the impact of the number of clients on model accuracy, we increase the number of clients from 120 to 240. Fig. 6 illustrates the accuracy of each dataset under both homogeneous and heterogeneous clustering conditions with different numbers of clients. According to Fig. 6, we observe that increasing the number of clients slightly affects model accuracy. For example, in the homogeneous clustering setting, the MNIST model accuracy differs by 0.07%, and the CIFAR-10 model accuracy differs by 0.13%. In the heterogeneous clustering setting, the MNIST model accuracy differs by 0.04%, and the CIFAR-10 model accuracy differs by 0.53%. Therefore, the number of clients does not significantly affect the experimental results. During the model rebuild process, the establishment of different coefficient matrices due to varying numbers of clients does not affect the accuracy of the model rebuild. Therefore, changes in the number of clients have little effect on model accuracy, resulting in only slight fluctuations in precision.

Impact of false positive rate: In this study, we investigate the impact of the false positive rate on model accuracy while keeping other parameters constant. We reduce the false positive rate from 0.5 to 0.25 and conduct experimental analyses and comparisons under both homogeneous and heterogeneous clustering conditions. According to Fig. 7, we observe that when the false positive rate changes, the model accuracy does not significantly vary. In the homogeneous clustering setting, the model accuracy for the MNIST, Fashion MNIST, CIFAR-10, and Texas100 datasets differs by only 0.03%, 0.01%, 0.13%, and 0.43%, respectively. In the heterogeneous clustering, with false positive rates of 0.25 and 0.5, the model accuracy for MNIST, Fashion MNIST, CIFAR-10, and Texas100 differs by 0.06%, 0.01%, 0.12%, and 0.81%, respectively. Since changing the false positive rate alters the coefficient matrix but does not affect the model rebuilding results, we can conclude that changing the false positive rate does not significantly affect the experimental results.

Impact of privacy threshold In this study, we analyze the effect of varying the privacy threshold on model accuracy while keeping other parameters constant. We increase the privacy threshold from 2 to 3 and conduct a comparative experimental analysis under both homogeneous and heterogeneous clustering conditions. As observed in Fig. 8, varying the privacy threshold has minimal impact on model accuracy. In the homogeneous clustering setup, the accuracy differences for the MNIST, Fashion-MNIST, CIFAR-10, and Texas100 datasets are only 0.06%, 0.01%, 0.13%, and 0.08%, respectively. In the heterogeneous clustering scenario, the accuracy differences are 0.06%, 0.01%, 0.12%, and 0.35%, respectively. Therefore, changing the privacy threshold does not have a significant impact on experimental outcomes.

D. Runtime Overhead of MingledPie

We measure the computation time of MingledPie on the Texas100 dataset, with the results presented in Table IV. The computation time of cluster identity generation is mainly influenced by p, T, k, and N. As shown in Table IV, the computation time for generating cluster identities is relatively small. As the false positive rate p decreases, the length of the cluster addresses that the client needs to match increases, leading to longer computation time. Increasing T results in the client needing to run the cluster identity generation algorithm more times on average to generate a sufficient number of cluster identities, which also adds to the computation time.

TABLE IV: Computation time for different components with default parameters $p = 0.5, T = 2, k = 10$, and $N = 120$

Process	Identity Generation	Model Aggregation	Model Rebuilding
$p = 0.5$	0.068s	18.014s	0.284s
$p = 0.25$	0.101s	11.763s	0.267s
$p = 0.125$	0.181s	8.211s	0.254s
$T = 2$	0.068s	18.014s	0.284s
$T = 3$	0.071s	23.047s	0.287s
$T = 4$	0.077s	29.523s	0.290s
$k = 5$	0.041s	10.137s	0.184s
$k = 8$	0.058s	14.521s	0.243s
$k = 10$	0.068s	18.014s	0.284s
$N = 60$	0.069s	9.154s	0.260s
$N = 120$	0.068s	18.014s	0.284s
$N = 240$	0.068s	39.138s	0.316s

TABLE V: Ablation study of MingledPie with default parameters $p = 0.5, T = 2, k = 5$, and $N = 120$

Dataset	Method	Train Accuracy	Test Accuracy
MNIST	①	97.38%	97.47%
	① + ②	52.07%	52.13%
	① + ② + ③	39.63%	37.94%
	① + ② + ④	97.38%	97.46%
Fashion MNIST	①	98.52%	98.49%
	① + ②	66.12%	65.82%
	① + ② + ③	70.46%	70.16%
	① + ② + ④	98.46%	98.18%
CIFAR-10	①	97.64%	95.27%
	① + ②	65.90%	66.11%
	① + ② + ③	51.39%	51.45%
	① + ② + ④	96.84%	94.13%
Texas100	①	83.67%	78.95%
	① + ②	27.78%	28.00%
	① + ② + ③	18.42%	18.39%
	① + ② + ④	82.60%	78.13%

① Client-side clustering method without defense. ② Perform cluster mingling. ③ Perform model rebuilding based on ideal coefficient matrix. ④ Perform model rebuilding based on the number of mingled clients.

Additionally, increasing k increases the number of cluster addresses that need to be matched, further raising the computational overhead. The server's model aggregation is performed within the CKKS homomorphic encryption domain. The overhead is primarily determined by the number of clusters k and the size of the mingled clusters, with the latter being influenced by p, T, and N. Although the model aggregation time grows with the above parameters, the average aggregation overhead per local model 0.03s appears to be acceptable for the server. Finally, the computation time required for the client to rebuild the cluster model primarily depends on the number of clusters k. As the number of clusters increases, the dimension of the linear system to be solved also increases, with the computational complexity typically being $O(k^3)$.

We further analyze the trade-off between privacy protection and computational overhead. As shown in Table IV, reducing the privacy threshold T leads to a corresponding decrease in computational overhead for both the client and the server. Therefore, selecting the minimum T that satisfies the required privacy guarantees is recommended. Additionally, as the false positive rate p increases, the computational burden on the server grows, while the client's overhead diminishes. Thus, the value of p should be chosen to balance the computational resources of the server and client.

E. Ablation Study

We conduct detailed ablation experiments to determine the effectiveness of cluster model rebuilding. The specific experimental results are shown in TABLE V. We can see that the client-side clustering method without defense achieves high training and testing model accuracy. When only mingled aggregation is used without model rebuilding, both training and testing accuracy significantly decreased. When model rebuilding was attempted using mingled cluster models based on the ideal coefficient matrix, the accuracy did not improve significantly and even decreased. This outcome is due to a discrepancy between the actual number of clients participating in mingled aggregation and the number estimated through the false positive rate, resulting in errors during model rebuilding and reduced accuracy due to cumulative errors.

Therefore, in our approach, using the actual number of mingled clients for model rebuilding results in model accuracy comparable to that of the client-side clustering method without defense. For instance, in the Texas-100 dataset, the client-side clustering method without defense achieves an accuracy of only 0.82% higher than our approach. This demonstrates that using the actual number of mingled clients for model rebuilding effectively improves accuracy, validating the effectiveness of the model rebuilding strategy.

VII. Conclusion

In this work, we propose a privacy-preserving clustered federated learning framework called MingledPie to mitigate the issue of preference profiling attacks. MingledPie mitigates this privacy limitation by treating model updates from other clusters as obfuscations. It mingles client model update messages with different preferences into mingled cluster models, protecting client privacy. MingledPie preserves model performance by rebuilding cluster models based on the false positive rate of the obfuscations. It employs a cluster identity generation method to mingled aggregate and model rebuilding to ensure privacy protection while maintaining usability. The experimental results show that we can effectively enhance the privacy protection of client preferences with negligible accuracy loss compared to baselines.

ACKNOWLEDGMENT

We would like to heartfeltly thank the anonymous reviewers for their constructive comments. This work is supported in part by the National Natural Science Foundation of China (No. 62272154, 62472163, 62202150, Excellent Youth Fund), the Hunan Provincial Natural Science Foundation of China (No. 2024JJ5096), the science and technology innovation Program of Hunan Province (No. 2023RC3125), the Science & Technology talents lifting project of Hunan Province (No. 2023TJ–N23), the Training Program for Excellent Young Innovators of Changsha (No. kq2209008), the Hunan Provincial Key Research and Development Program (2024AQ2041), and the Hunan Provincial Innovation Foundation for Postgraduate (No. CX20230389). Yang Xu is the corresponding author of this paper.

REFERENCES

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *International Conference on Artificial Intelligence and Statistics*, 2017, pp. 1273–1282.

[2] Q. Li, Y. Diao, Q. Chen, and B. He, "Federated learning on non-iid data silos: An experimental study," in *IEEE International Conference on Data Engineering*, 2022, pp. 965–978.

[3] C. Yang, Q. Wang, M. Xu, Z. Chen, K. Bian, Y. Liu, and X. Liu, "Characterizing impacts of heterogeneity in federated learning upon large-scale smartphone data," in *World Wide Web*, 2021, p. 935–946.

[4] S. Zawad, A. Ali, P.-Y. Chen, A. Anwar, Y. Zhou, N. Baracaldo, Y. Tian, and F. Yan, "Curse or redemption? how data heterogeneity affects the robustness of federated learning," in *AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, 2021, pp. 10 807–10 814.

[5] R. Pi, W. Zhang, Y. Xie, J. Gao, X. Wang, S. Kim, and Q. Chen, "DynaFed: Tackling client data heterogeneity with global dynamics," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 12 177–12 186.

[6] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "SCAFFOLD: Stochastic controlled averaging for federated learning," in *International Conference on Machine Learning*, vol. 119, 2020, pp. 5132–5143.

[7] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran, "An efficient framework for clustered federated learning," *IEEE Transactions on Information Theory*, vol. 68, no. 12, pp. 8076–8091, 2022.

[8] Y. Guo, X. Tang, and T. Lin, "FedRC: Tackling diverse distribution shifts challenge in federated learning by robust clustering," in *International Conference on Machine Learning*, 2024.

[9] J. A. Carrillo, N. G. Trillos, S. Li, and Y. Zhu, "FedCBO: Reaching group consensus in clustered federated learning through consensus-based optimization," *Journal of Machine Learning Research*, vol. 25, no. 214, pp. 1–51, 2024.

[10] B. Liu, Y. Ma, Z. Zhou, Y. Shi, S. Li, and Y. Tong, "CASA: Clustered federated learning with asynchronous clients," in *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2024, p. 1851–1862.

[11] H. Kim, H. Kim, and G. D. Veciana, "Clustered federated learning via gradient-based partitioning," in *International Conference on Machine Learning*, 2024.

[12] Y. Wang, Z. Su, Y. Pan, T. H. Luan, R. Li, and S. Yu, "Social-aware clustered federated learning with customized privacy preservation," *IEEE/ACM Transactions on Networking*, vol. 32, no. 5, pp. 3654–3668, 2024.

[13] Y. Chen, Y. Su, M. Zhang, H. Chai, Y. Wei, and S. Yu, "FedTor: An anonymous framework of federated learning in internet of things," *IEEE Internet of Things Journal*, vol. 9, no. 19, pp. 18 620–18 631, 2022.

[14] A. M. Girgis, D. Data, S. Diggavi, P. Kairouz, and A. T. Suresh, "Shuffled model of federated learning: Privacy, accuracy and communication trade-offs," *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 1, pp. 464–478, 2021.

[15] Y. Cai, W. Ding, Y. Xiao, Z. Yan, X. Liu, and Z. Wan, "SecFed: A secure and efficient federated learning based on multi-key homomorphic encryption," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–17, 2023.

[16] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu, "BatchCrypt: Efficient homomorphic encryption for cross-silo federated learning," in *USENIX Annual Technical Conference*, 2020, pp. 493–506.

[17] F. Mo, H. Haddadi, K. Katevas, E. Marin, D. Perino, and N. Kourtellis, "PPFL: Privacy-preserving federated learning with trusted execution environments," in *International Conference on Mobile Systems, Applications, and Services*, 2021, p. 94–108.

[18] P. Rieger, T. Krauß, M. Miettinen, A. Dmitrienko, and A.-R. Sadeghi, "Crowdguard: Federated backdoor detection in federated learning," in *Network and Distributed System Security Symposium*, 2024.

[19] R. Du, C. Liu, and Y. Gao, "Anonymous federated learning framework in the internet of things," *Concurrency and Computation: Practice and Experience*, vol. 36, no. 2, p. e7901, 2024.

[20] N. S. Evans, R. Dingledine, and C. Grothoff, "A practical congestion attack on tor using long paths," in *USENIX Security Symposium*, 2009, p. 33–50.

[21] K. Kohls and C. Pöpper, "DigesTor: Comparing passive traffic analysis attacks on tor," in *European Symposium on Research in Computer Security*, 2018, pp. 512–530.

[22] P. Sirinam, N. Mathews, M. S. Rahman, and M. Wright, "Triplet fingerprinting: More practical and portable website fingerprinting with n-shot learning," in *ACM SIGSAC Conference on Computer and Communications Security*, 2019, p. 1131–1148.

[23] Y. Zhu, X. Fu, B. Graham, R. Bettati, and W. Zhao, "Correlation-based traffic analysis attacks on anonymity networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 7, pp. 954–967, 2010.

[24] Z. Liu, J. Guo, W. Yang, J. Fan, K.-Y. Lam, and J. Zhao, "Privacy-preserving aggregation in federated learning: A survey," *IEEE Transactions on Big Data*, pp. 1–20, 2022.

[25] L. Zhao, J. Jiang, B. Feng, Q. Wang, C. Shen, and Q. Li, "SEAR: Secure and efficient aggregation for byzantine-robust federated learning," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 5, pp. 3329–3342, 2022.

[26] G. Beck, J. Len, I. Miers, and M. Green, "Fuzzy message detection," in *ACM SIGSAC Conference on Computer and Communications Security*, 2021, p. 1507–1528.

[27] K. Wei, J. Li, C. Ma, M. Ding, W. Chen, J. Wu, M. Tao, and H. V. Poor, "Personalized federated learning with differential privacy and convergence guarantee," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 4488–4503, 2023.

[28] F. Sattler, K.-R. Muller, and W. Samek, "Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 8, pp. 3710–3722, 2021.

[29] M. Duan, D. Liu, X. Ji, R. Liu, L. Liang, X. Chen, and Y. Tan, "Fedgroup: Efficient federated learning via decomposed similarity-based clustering," in *IEEE International Conference on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking*, 2021, pp. 228–237.

[30] Y. Gou, R. Wang, Z. Li, M. A. Imran, and L. Zhang, "Clustered hierarchical distributed federated learning," in *IEEE International Conference on Communications*, 2022, pp. 177–182.

[31] S. Vahidian, M. Morafah, W. Wang, V. Kungurtsev, C. Chen, M. Shah, and B. Lin, "Efficient distribution similarity identification in clustered federated learning via principal angles between client data subspaces," in *AAAI Conference on Artificial Intelligence*, vol. 37, no. 8, 2023, pp. 10 043–10 052.

[32] G. Long, M. Xie, T. Shen, T. Zhou, X. Wang, and J. Jiang, "Multi-center federated learning: Clients clustering for better personalization," *World Wide Web*, vol. 26, no. 1, p. 481–500, 2022.

[33] C. Briggs, Z. Fan, and P. Andras, "Federated learning with hierarchical clustering of local updates to improve training on non-iid data," in *International Joint Conference on Neural Networks*, 2020, pp. 1–9.

[34] Y. Diao, Q. Li, and B. He, "Exploiting label skews in federated learning with model concatenation," *AAAI Conference on Artificial Intelligence*, vol. 38, no. 10, pp. 11 784–11 792, 2024.

[35] C. Fan, R. Chen, J. Mo, and L. Liao, "Personalized federated learning for cross-building energy knowledge sharing: Cost-effective strategies and

model architectures," *Applied Energy*, vol. 362, pp. 123 016–123 031, 2024.
[36] Z. Wang, H. Xu, J. Liu, Y. Xu, H. Huang, and Y. Zhao, "Accelerating federated learning with cluster construction and hierarchical aggregation," *IEEE Transactions on Mobile Computing*, vol. 22, no. 7, pp. 3805–3822, 2023.
[37] C. Li, G. Li, and P. K. Varshney, "Federated learning with soft clustering," *IEEE Internet of Things Journal*, vol. 9, no. 10, pp. 7773–7782, 2022.
[38] Y. Ruan and C. Joe-Wong, "FedSoft: Soft clustered federated learning with proximal local updating," *AAAI Conference on Artificial Intelligence*, vol. 36, no. 7, pp. 8124–8131, 2022.
[39] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran, "An efficient framework for clustered federated learning," in *International Conference on Neural Information Processing Systems*, 2020.
[40] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *IEEE Symposium on Security and Privacy*, 2017, pp. 3–18.
[41] M. S. R. Shuvo and D. Alhadidi, "Membership inference attacks: Analysis and mitigation," in *IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, 2020, pp. 1410–1419.
[42] K. Ganju, Q. Wang, W. Yang, C. A. Gunter, and N. Borisov, "Property inference attacks on fully connected neural networks using permutation invariant representations," in *ACM SIGSAC Conference on Computer and Communications Security*, 2018, p. 619–633.
[43] J. Zhou, Y. Chen, C. Shen, and Y. Zhang, "Property inference attacks against gans," in *Network and Distributed System Security Symposium*, 2022.
[44] Y. Zhang, R. Jia, H. Pei, W. Wang, B. Li, and D. Song, "The secret revealer: Generative model-inversion attacks against deep neural networks," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 253–261.
[45] T. Zhu, D. Ye, S. Zhou, B. Liu, and W. Zhou, "Label-only model inversion attacks: Attack with the least information," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 991–1005, 2023.
[46] C. Zhou, Y. Gao, A. Fu, K. Chen, Z. Dai, Z. Zhang, M. Xue, and Y. Zhang, "PPA: Preference profiling attack against federated learning," in *Network and Distributed System Security Symposium*, 2023.
[47] Q. Zhou, Z. Han, and J. Wu, "Local difference-based federated learning against preference profiling attacks," in *Web Information Systems Engineering*, 2023, pp. 275–288.
[48] M. Gong, Y. Zhang, Y. Gao, A. K. Qin, Y. Wu, S. Wang, and Y. Zhang, "A multi-modal vertical federated learning framework based on homomorphic encryption," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 1826–1839, 2024.
[49] L. Zhang, J. Xu, P. Vijayakumar, P. K. Sharma, and U. Ghosh, "Homomorphic encryption-based privacy-preserving federated learning in IoT-enabled healthcare system," *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 5, pp. 2864–2880, 2023.
[50] A. Mondal, Y. More, R. H. Rooparaghunath, and D. Gupta, "Poster: FLATEE: Federated learning across trusted execution environments," in *IEEE European Symposium on Security and Privacy*, 2021, pp. 707–709.
[51] R. Hernandez, O. G. Bautista, M. H. Manshaei, A. Sahin, and K. Akkaya, "Outsourcing privacy-preserving federated learning on malicious networks through mpc," in *IEEE Conference on Local Computer Networks*, 2023, pp. 1–4.
[52] C. Zhang, S. Ekanut, L. Zhen, and Z. Li, "Augmented multi-party computation against gradient leakage in federated learning," *IEEE Transactions on Big Data*, pp. 1–10, 2022.
[53] L. Lyu, J. Yu, K. Nandakumar, Y. Li, X. Ma, J. Jin, H. Yu, and K. S. Ng, "Towards fair and privacy-preserving federated deep models," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 11, pp. 2524–2541, 2020.
[54] W. Wang, X. Li, X. Qiu, X. Zhang, V. Brusic, and J. Zhao, "A privacy preserving framework for federated learning in smart healthcare systems," *Information Processing & Management*, vol. 60, no. 1, p. 103167, 2023.
[55] J. Cui, H. Zhu, H. Deng, Z. Chen, and D. Liu, "FeARH: Federated machine learning with anonymous random hybridization on electronic medical records," *Journal of Biomedical Informatics*, vol. 117, p. 103735, 2021.
[56] Z. Dai, C. Zhou, and A. Fu, "Decaf: Data distribution decompose attack against federated learning," *arXiv preprint arXiv:2405.15316*, 2024.
[57] R. Ramakrishna and G. Dán, "Inferring class-label distribution in federated learning," in *Proceedings of the 15th ACM Workshop on Artificial Intelligence and Security*, 2022, pp. 45–56.
[58] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," in *International Conference on Learning Representations*, 2020.
[59] Z. Wang, H. Xu, J. Liu, Y. Xu, H. Huang, and Y. Zhao, "Accelerating federated learning with cluster construction and hierarchical aggregation," *IEEE Transactions on Mobile Computing*, vol. 22, no. 7, pp. 3805–3822, 2022.
[60] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran, "An efficient framework for clustered federated learning," *IEEE Transactions on Information Theory*, vol. 68, no. 12, pp. 8076–8091, 2022.
[61] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Machine learning and systems*, vol. 2, pp. 429–450, 2020.
[62] A. Dieuleveut, G. Fort, E. Moulines, and G. Robin, "Federated-EM with heterogeneity mitigation and variance reduction," *Advances in Neural Information Processing Systems*, vol. 34, pp. 29 553–29 566, 2021.
[63] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.
[64] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.
[65] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.
[66] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.
[67] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[68] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

APPENDIX A
APPENDIX

A. Proof of Theorem 4

We first calculate the probability distribution of the number of false positive clients, and then based on the properties of the matrix $H^*_{k\times k}$, we derive the constraints on the perturbation. Finally, by combining the probability distribution of the perturbation and the constraints, we conclude that H_δ is negligible and thus $H_{k\times k}$ is invertible.

According to Algorithm 1, if a client is mingled into a cluster, their true cluster identity's public key must match the cluster's address in n bits. Assuming within cluster b, the number of clients that satisfy the condition of matching the first position of the cluster address of cluster a, denoted as $x^1_{a,b}$, is a random variable, the probability distribution of $x^1_{a,b}$ is given as:

$$PR^1(x_{a,b}) = C^{x_{a,b}}_{|G^*_b|}(1/2)^{|G^*_b|} \tag{24}$$

On this basis, we iteratively compute the probability distribution of n-bit matches using the following equation:

$$PR^n(x_{a,b}) = \sum_{y=x_{a,b}}^{|G^*_b|} PR^{n-1}(y)\cdot C^{x_{a,b}}_{y}(1/2)^{y} \tag{25}$$

Then, for a strictly diagonally dominant matrix $H^*_{k\times k}$, we have $|x^*_{ii}| > \sum_{j\neq i}|x^*_{ij}|$. For the perturbation matrix H_δ, its diagonal elements are zero, and we assume that the remaining elements do not exceed C, i.e., $|x_{\sigma_{ij}}| < C$ for $i \neq j$. We need to find the value of C to ensure that $H_{k\times k}$ is invertible.

According to the Gerschgorin circle theorem, the eigenvalues λ_H of the matrix $H_{k\times k}$ lie within the following constraint:

$$|\lambda_H - x^*_{ii}| \leq \sum_{j\neq i} |x_{\sigma_{ij}}| \tag{26}$$

Since $|x_{\sigma_{ij}}| < C$, we can express this as:

$$|\lambda_H - x^*_{ii}| \leq (k-1)C \tag{27}$$

To ensure that the eigenvalues of H are non-zero, we require: $|x^*_{ii}| > (k-1)C$, which implies:

$$C < \frac{1}{k-1} \min |x^*_{ii}| \tag{28}$$

For $n = 1$, we have the probability distribution $PR^1_{x_{a,b}}$, which is a binomial distribution. When $|G^*_b|$ is sufficiently large, according to the central limit theorem, if a sufficiently large number of samples are drawn from a population with any distribution having a finite variance, these samples, which follow the same distribution, will have a mean that asymptotically approaches a normal distribution. Therefore, this binomial distribution can be approximated by a normal distribution $X \sim \mathcal{N}((1/2)|G^*_b|, (\sqrt{|G^*_b|}/2)$.

In a normal distribution, 99.73% of the data falls within three standard deviations of the mean, i.e., within the interval $[(1/2)|G^*_b| - 3(\sqrt{|G^*_b|}/2), (1/2)|G^*_b| + 3(\sqrt{|G^*_b|}/2)]$. The maximum value of the fluctuation C is three standard deviations, $3(\sqrt{|G^*_b|}/2$, which is:

$$3(\sqrt{|G^*_b|}/2 < |G^*_b|/(k-1) \tag{29}$$

If the smallest cluster $|G^*_b|$ satisfies Eq. 29, then it will be satisfied for all clusters:

$$min\{|G^*_b|\}_{b=1}^{k} > (k-1)^2 + 105/16 \tag{30}$$

For the case where $n > 1$, the interval in which C falls will be within the range determined by the case $n = 1$. Consequently, it will also satisfy the required conditions. Thus, when $|G^*_b|$ satisfies this condition, C has a 99.73% probability of falling within the three standard deviation interval and H_δ is considered negligible, ensuring that $H_{k\times k}$ is invertible.

B. Datasets used in Evaluation

1) MNIST: contains images of handwritten digits ranging from 0 to 9 [63], totaling 10 categories. It includes 60k training images and 10k test images, all of which are 28x28 grayscale images.

2) Fashion MNIST: is a dataset of fashion item images across 10 categories [64]. Similar to MNIST, it includes 60k training images and 10k test images, all of which are 28x28 grayscale images.

3) CIFAR-10: is a dataset for recognizing general objects [65], containing 10 categories of colored images, such as airplanes, cats, and dogs. It consists of 50k training images and 10k test images, each sized at 32x32 pixels.

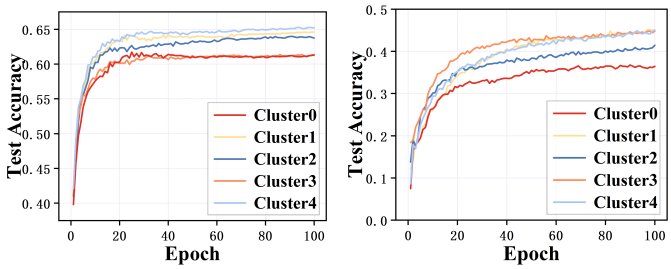
4) CIFAR-100: is a dataset used for image classification tasks [65], containing 100 categories. It includes 50k training images and 10k test images.

5) Texas100: This dataset records patient discharge data from various medical institutions, released by the Texas Department of Health Services. It includes information such as causes of injury, diagnoses, treatment procedures, and patient details. Texas100 is a processed version of this data [40], containing 67,330 records and 6,169 binary features, divided into 100 categories, each representing different types of patients.

6) Purchase100: Purchaes is a dataset provided by Kaggle, consisting of shopping records of thousands of people over nearly a year, including product names, stores, dates, and more. Purchase100 is a simplified version of this dataset [40], comprising 100 categories with a total of 197,324 records. Each record is represented by 600 binary bits, where each bit indicates whether a specific product was purchased.

C. Extra Experimental Result

1) Different models: We examine the performance of the approach on different models within the same dataset. Taking the CIFAR-100 dataset as an example, we test it on both CNN and ResNet models. As shown in TABLE VI, we observe that the model accuracy reaches 42.57% with the CNN model and 63.24% with the ResNet model. Additionally, as shown in Fig. 9, we also test the accuracy of all cluster models on the CNN and ResNet models. According to TABLE VI, we can see that in the ResNet model, the highest cluster model accuracy is 65.20%, and the lowest is 61.30%, resulting in a difference of 3.90%. In the CNN model, the highest cluster model accuracy is 45.10%, while the lowest is 36.45%, with a difference of 8.65%. The experimental results indicate significant differences in clustering performance between different models on the same dataset. The ResNet model shows higher overall accuracy compared to the CNN model and exhibits smaller accuracy differences across different cluster models, suggesting greater robustness in handling complex data features.



(a) ResNet (b) CNN

Fig. 9: Convergence curve of the cluster model for each cluster

2) IID Rate: In Theorem 6, we analyze that the accuracy loss in MingledPie primarily arises from equating the aggregation results of some clients within a cluster to the aggregation results of all clients in that cluster during model rebuilding. This error is significantly influenced by the IID rate of the clusters containing false-positive clients. Therefore, we conduct experiments by setting different IID rates for the data

TABLE VI: Cluster model accuracy for each cluster

Cluster	ResNet	CNN
Cluster0	61.30%	36.45%
Cluster1	64.60%	45.05%
Cluster2	63.75%	41.45%
Cluster3	61.35%	44.80%
Cluster4	65.20%	45.10%
Average Test Accuracy	63.24%	42.57%

distribution of the clients. According to TABLE VII, we find that when the IID rate is 0%, the difference in model accuracy is at its lowest, and as the IID rate increases, the difference in model accuracy also increases. Thus, when the aggregated model from these false-positive clients more closely aligns with the true cluster model, the resulting accuracy loss is minimized. To further reduce accuracy loss, we can adopt more precise clustering algorithms or client selection strategies to increase the IID rate within clusters.

TABLE VII: Model accuracy with different IID rates with default parameter $p = 0.5$, $T = 2$, $k = 5$ and $N = 120$

Parameter	MingledPie	IFCA
IID rate=0%	98.14%	98.49%
IID rate=20%	83.98%	88.52%
IID rate=40%	76.77%	82.46%
IID rate=60%	72.29%	77.71%

3) Dynamic joining of clients: During the training process, we allow for the dynamic joining of other clients and conduct training. Taking the MNIST dataset as an example, we introduce 10 new clients every 10 training rounds. These clients are assigned to appropriate clusters based on their data distribution. We set the data distribution of the new clients using different IID rates and study their impact on model accuracy. As shown in Table VIII, we observe a significant decrease in training accuracy and a downward trend in testing accuracy with the increase in IID rate. Specifically, when the IID rates are 0%, 20%, 40%, and 60%, the training accuracies are 97.35%, 88.92%, 80.54% and 72.14%, respectively. The decrease in model accuracy is attributed to the increased diversity of the training data; higher IID rates make the learning process more challenging.

TABLE VIII: Dynamic joining of clients every 10 rounds with default parameters newcomers = 10

Parameter	Train Accuracy	Test Accuracy
IID rate=0%	97.35%	97.47%
IID rate=20%	88.92%	97.40%
IID rate=40%	80.54%	97.37%
IID rate=60%	72.14%	97.22%

4) Computation cost of HE: CKKS homomorphic encryption is used in the evaluation as a secure model update and aggregation method. We compute the overhead of model encryption and decryption under different model structures. TABLE IX shows that the overhead of model encryption is correlated with the amount of parameter data in the model, with model encryption and decryption time significantly increasing in Resnet models with a high number of parameters and decreasing in FCCN models with a reduced number of parameters. It is worth noting that we did not set up parallel computation when testing the encryption and decryption process and we did not use the homomorphic encryption scheme of SOTA like BatchCrypt [16] which improves computational performance with less loss of precision. Because the focus of this scheme is not homomorphic encryption, this part can be replaced to achieve better performance.

TABLE IX: Computation Time of HE

Train Model	FCNN	CNN	ResNet-18
Weight Encryption Time	3.32s	7.22s	22.08s
Weight Decryption Time	0.36s	0.73s	2.78s

APPENDIX B
ARTIFACT APPENDIX

A. Description & Requirements

1) How to access: The artifact is available under open-source license at https://github.com/CHENGZ03/MingledPie and https://doi.org/10.5281/zenodo.14135448.

2) Hardware dependencies: Commodity GPUs (e.g., we used NVIDIA GeForce RTX 4090 in our evaluation).

3) Software dependencies: PyTorch, torchvision, argparse, ecpy, numpy, torch, seaborn, copy, sklearn, random, pandas and matplotlib.

4) Benchmarks: (1) Dataset: MNIST [63], Fashion MNIST [64], CIFAR-10 [65], CIFAR-100 [65], Texas100 [40], and Purchase100 [40]. (2) Models: For MNIST and Fashion MNIST, we use a fully connected neural network (FCNN) [66] with two fully connected layers and a single hidden layer of size 200. For CIFAR-10, we adopt a convolutional neural network (CNN) [67] with two convolutional layers, a max-pooling layer, and three fully connected layers. For CIFAR-100, we adopt the ResNet-18 model [68]. For Texas100 and Purchase100, we use the multilayer perceptron model (MLP) [66].

We provide all datasets, and models used in our evaluation. The links to download these data are available in the artifact

B. Artifact Installation & Configuration

Our experiments are run on Python. Please go to the README file in the artifact to install the software dependencies listed in A-3 above, and download the dataset and models listed in A-4 above.

C. Experiment Workflow

Please see the Evaluation section.

D. Major Claims

- (C1): The approach can withstand preference attacks and protect client privacy. This is proven by the experiment (E1) whose results are illustrated/reported in Fig. 3.
- (C2): The approach can achieve the reconstruction of the cluster model to ensure the performance of the cluster model. This is proven by experiment (E2), with the results explained in Fig. 4, Fig. 5, Fig. 6, Fig. 7, Fig. 8, TABLE III, TABLE V.

E. Evaluation

Considering the time limit of the artifact evaluation, we recommend focusing on evaluating the defense against preference profiling attacks and the accuracy of cluster models on 4 datasets. We designed two scaled-down experiments located in the E1 and E2 folders. The code in E1 includes the generation of cluster identities, but omits the model training part to verify the effectiveness of the defense method. The code in E2 focuses on evaluating the performance of cluster model rebuilding under ideal conditions. This evaluation can still conclusively demonstrate that the proposed method effectively defends against preference profiling attacks while maintaining model accuracy.

For anyone interested in a comprehensive evaluation, we have provided the complete experimental code in the "Complete_experimental_procedure" folder.

1) Experiment (E1): [Defense against preference profiling attacks] [5 human-minutes + 5 compute-hour]: This experiment is conducted to explore the adversary preference profiling accuracy under different parameters such as mingled ratio p, number of clusters k, privacy threshold T. The experiment first simulates the cluster identity generation algorithm (Algorithm 1) to generate both real and fake cluster identities for a client. Next, it calculates the probability that an adversary successfully guesses the real cluster identity, which is used as the success probability of the preference analysis.

Execution. The program is stored in the folder "E1." You can directly run the 'run_privacy_eval.sh' file to test the accuracy of preference profiling analysis under different false positive rates, cluster numbers, and privacy thresholds.

Results. As shown in Fig. 10, the program outputs three heatmaps illustrating the preference profiling analysis under different false positive rates, cluster numbers, and privacy thresholds (corresponding to Fig. 3 in the paper).

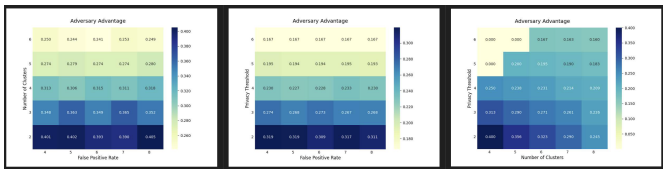


Fig. 10: Example output showing adversary preference profiling accuracy

2) Experiment (E2): [Accuracy of cluster models] [30 human-minutes + 5 compute-hour]: This experiment evaluates the performance of the cluster model rebuilding algorithm (Algorithm 2) under ideal conditions, and evaluates the influence of different parameters. We first set up clients with different data distributions to participate in the training. Then, in each training round, we run the cluster identity generation algorithm for each client and train the local models. Next, we run the cluster model rebuilding algorithm to compute the accurate cluster models from the mingled ones and output the accuracy of the rebuilt cluster models.

Execution. The program is stored in the "E2" folder. Each subfolder corresponds to experiments on a specific dataset. The Python programs in each subfolder test various aspects of the cluster model rebuilding algorithm. For instance, in the E2/mnist folder:

- mnist.py tests the model accuracy under homogeneous clustering,
- non-mnist.py tests model accuracy under heterogeneous clustering,
- mnist-k=10.py tests model accuracy with 10 clusters,
- mnist-m=240.py tests model accuracy with 240 clients,
- mnist-p=0.25.py tests model accuracy with a false positive rate of 0.25,
- mnist-T=3.py tests model accuracy with a privacy threshold of 3, and
- mnist-ifca.py tests the model accuracy using the IFCA method for comparison.

Results. Each Python program outputs model accuracy in the command line and generates two plots: one for the average model accuracy and another for the accuracy of all cluster models (Fig. 11). Using mnist as an example, the results from mnist.py verify Fig. 4 and TABLE III. The results from non-mnist.py verify TABLE III. The results from mnist-k=10.py validate Fig. 5, the results from mnist-m=240.py validate Fig. 6, the results from mnist-p=0.25.py validate Fig. 7, the results from mnist-T=3.py validate Fig. 8, and the results from mnist-ifca.py validate TABLE V.

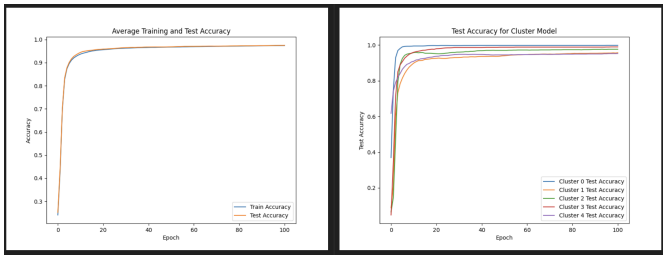


Fig. 11: Example output showing cluster model convergence curve

F. Notes

The Artifact Evaluation Committee (AEC) has evaluated a prior version of this artifact.