Published: 16 October 2023

RAID 2023: The 26th International Symposium on Research in Attacks, Intrusions and Defenses
October 16 - 18, 2023
Hong Kong, China

RESEARCH-ARTICLE

Exploring Clustered Federated Learning's Vulnerability against Property Inference Attack

HYUNJUN KIM, Inter-University Semiconductor Research Center, Seoul, South Korea

YUNGI CHO, Inter-University Semiconductor Research Center, Seoul, South Korea

YOUNGHAN LEE, Inter-University Semiconductor Research Center, Seoul, South Korea

HO BAE, Ewha Womans University, Seoul, Seoul, South Korea

YUNHEUNG PAEK, Inter-University Semiconductor Research Center, Seoul, South Korea

RAID '23: Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses (October 2023)
https://doi.org/10.1145/3607199.3607218
ISBN: 9798400707650

Exploring Clustered Federated Learning's Vulnerability against Property Inference Attack

Hyunjun Kim*
hjkim@sor.snu.ac.kr
Seoul National University
Department of Electrical and Computer Engineering and Inter-university Semiconductor Research Center
Seoul, Republic of Korea

Yungi Cho*
ygcho@sor.snu.ac.kr
Seoul National University
Department of Electrical and Computer Engineering and Inter-university Semiconductor Research Center
Seoul, Republic of Korea

Younghan Lee
yhlee@sor.snu.ac.kr
Seoul National University
Department of Electrical and Computer Engineering and Inter-university Semiconductor Research Center
Seoul, Republic of Korea

Ho Bae†
hobae@ewha.ac.kr
Ewha Womans University
Department of Cyber Security
Seoul, Republic of Korea

Yunheung Paek†
ypaek@snu.ac.kr
Seoul National University
Department of Electrical and Computer Engineering and Inter-university Semiconductor Research Center
Seoul, Republic of Korea

ABSTRACT

Clustered federated learning (CFL) is an advanced technique in the field of federated learning (FL) that addresses the issue of catastrophic forgetting caused by non-independent and identically distributed (non-IID) datasets. CFL achieves this by clustering clients based on the similarity of their datasets and training a global model for each cluster. Despite the effectiveness of CFL in mitigating performance degradation resulting from non-IID datasets, the potential risk of privacy leakages in CFL has not been thoroughly studied. Previous work evaluated the risk of privacy leakages in FL using the *property inference attack* (PIA), which extracts information about unintended properties (i.e., attributes that differ from the target attribute of the global model's main task). In this paper, we explore the potential risk of unintended property leakage in CFL by subjecting it to both *passive* and *active PIAs*. Our empirical analysis shows that the passive PIA performance on CFL is substantially better than that on FL in terms of the attack AUC score. Moreover, we propose an enhanced active PIA method tailored for CFL to improve the attack performance. Our method introduces a *scale-up parameter* that amplifies the impact of malicious local updates, resulting in better performance than the previous technique. Furthermore, we demonstrate that the vulnerability of CFL can be alleviated by applying differential privacy (DP) mechanisms at the client-level. Unlike previous works, which have shown that applying DP to FL can induce a high utility loss, our empirical results indicate that DP can be used as a defense mechanism in CFL, leading to a better trade-off between privacy and utility.

CCS CONCEPTS

• **Security and privacy**; • **Computing methodologies** → **Distributed artificial intelligence**;

KEYWORDS

clustered federated learning, property inference attack, differential privacy

ACM Reference Format:
Hyunjun Kim, Yungi Cho, Younghan Lee, Ho Bae, and Yunheung Paek. 2023. Exploring Clustered Federated Learning's Vulnerability against Property Inference Attack. In *The 26th International Symposium on Research in Attacks, Intrusions and Defenses (RAID '23), October 16–18, 2023, Hong Kong, China.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3607199.3607218

1 INTRODUCTION

Federated learning (FL) is a collaborative learning method that trains a single global model using local datasets of individual clients. During the training process, only the local updates (i.e., gradients) generated by each client are uploaded and aggregated on the server to prevent private local datasets from being exposed to other clients or the server. Such a scheme enables the server to train the global model without accessing private raw data directly. In real-world scenarios, the clients' private local datasets exhibit heterogeneity, which presents the non-independent and identically distributed (non-IID) data problem. Although early FL studies [12, 16] reported that their method could handle heterogeneity in the data distributions, training the global model over non-IID data inevitably entails

*Both authors contributed equally to this research.
†Corresponding authors: Yunheung Paek; Ho Bae.

RAID '23, October 16–18, 2023, Hong Kong, China

ACM ISBN 979-8-4007-0765-0/23/10…$15.00
https://doi.org/10.1145/3607199.3607218

performance degradation. To overcome this issue, many researchers have explored advanced FL techniques for handling non-IID data, such as revising the loss function, the aggregation method, and the dataset distribution [8, 10, 21–23, 28].

Among these techniques, clustered federated learning (CFL) is a simple yet powerful method that groups clients based on the similarity of their data distributions [8, 21, 22]. As clients with similar data distributions are grouped into a single cluster, their data distributions become relatively independent and identically distributed (IID) within that cluster. Each cluster is allocated to one of the global models, and each global model is trained only with clients in its allocated cluster. Sattler et al. [22] proposed a hierarchical clustering method based on cosine similarity between updates generated from each pair of clients. Ghosh et al. [8] moved the clustering location from the server to the client to reduce the computational cost. Under this approach, each client receives every global model of each cluster and then selects the cluster whose model has the lowest loss over the client's dataset. Furthermore, Ruan et al. [21] proposed a soft clustering method as a relaxed version of the hard clustering method; their method regards each client's local dataset as a mixture of multiple data distributions, and a global model is maintained for each data distribution. Each client participates in training multiple global models according to the degree of relevance between each global model's data distribution and the client's data distribution. The aforementioned studies on CFL have effectively mitigated the issue of performance degradation in FL caused by non-IID datasets. However, the clustering process in CFL results in relatively IID data distribution among clients within a specific cluster, which may raise privacy concerns regarding the presence of similar data between the clients. This potential privacy risk in CFL has not been thoroughly examined yet. Previous studies on FL have evaluated the risk of privacy leakages using advanced inference attacks that infer private information from local updates, aggregated updates, or the trained global model [5, 6, 11, 18, 19, 24].

Among these attacks, our focus is on the property inference attack (PIA), which seeks to extract unintended property information from a model trained for a specific task [5, 18, 19]. Unintended properties refer to attributes that are different from the target attribute of the model's main task. For instance, in the process of training a global model to infer races from given human faces, unintended property information (e.g., wearing sunglasses) regarding the training images can be included in the aggregated updates. PIA extracts such unintended property information even if it is independent of the race, thereby compromising the person's privacy. Melis et al. [18] proposed two types of PIA specifically designed for FL: passive and active. The passive PIA analyzes the pattern of aggregated updates computed from the change in the global model parameters to determine whether the unintended property appears in the clients' datasets. The active PIA method further improves the attack performance by training the global model to learn separable representations for the data with and without the unintended property by manipulating the attacker's local update.

After conducting both passive PIA and active PIA on CFL, we confirm our assertion that CFL scheme may raise the risk of unintended property leakage. In our experiments, we demonstrate that the AUC of passive PIA is on average 30.32% higher on CFL compared to FL (among 5 task combinations). We analyze and discuss this phenomenon in subsection 4.4. Furthermore, we find that active PIA is relatively ineffective against CFL because the attacker can only manipulate a single global model at a time. To address this limitation, we propose a method inspired by the work of Bagdasaryan et al. [2] on backdoor attacks, which involves increasing the magnitude of the attacker's local update. As a result, our active PIA shows better performance, with an average increase of 4.14% in the AUC, compared to the previous active PIA [18].

To mitigate PIA attacks on CFL, we explore several existing defense techniques that have been reported to be effective against privacy leakage attacks such as dropout [29], L2 regularization [11, 24], and differential privacy (DP) [1, 7, 17, 20, 25, 26]. While dropout and L2 regularization are widely used as regularization techniques, they can also be used for defense against privacy leakage attacks because they remove partial information in local updates. DP provides a privacy guarantee by adding noise to local updates. However, previous studies [18, 19] have reported that dropout and DP are not effective against PIA in FL. In this study, we evaluate the effectiveness of dropout, L2 regularization, and DP as defense techniques against PIA in CFL. Our experimental results show that dropout and L2 regularization are not effective for securing CFL against PIA. We also evaluate the effectiveness of DP by adopting it to CFL at the client-level, similar to previous works [18, 19]. Specifically, we add noise to each client's update using DP-SGD [1] locally before aggregation. When evaluating the PIA attack performance under the privacy budget $\epsilon = 33.14$ (noise level $\sigma = 0.2$), our results indicate that DP noise effectively prevents the attacker from inferring unintended properties on both FL and CFL. Also, when evaluating the utility loss, the CFL model performance decreases by 3.51% on average, while the FL model performance decreases by 19.55% on average after applying DP. This suggests that CFL provides a better trade-off between privacy and utility than FL with DP.

Our contributions are as follows:

- To the best of our knowledge, we are the first to perform PIA on a well-known CFL framework and show CFL is significantly vulnerable to PIA. In addition, we further improve the attack performance by proposing a powerful active attack method tailored for CFL.
- We provide an informative insight why CFL is more vulnerable than FL against PIA using visual representations.
- We demonstrate that DP can be effectively applied to CFL (unlike FL) to mitigate PIA while maintaining the model performance.

1.1 Organization

Section 2 describes FL and CFL, and an overview of previous PIA on FL. In section 3, we establish CFL scenario and PIA threat model, followed by an explanation of how PIA can be adapted to CFL, and proposed active PIA. In section 4, we show that CFL is more vulnerable than FL, and describes that a proposed active attack is more powerful than the previous active attack in CFL. Moreover, we analyze the root cause for CFL's vulnerability. In section 5, we provide experiments on CFL with existing defense techniques including DP approach. Finally, in section 6, we discuss the limitations and future work.

Algorithm 1: Clustered Federated Learning (CFL)

Input : Number of clusters NC, Number of clients N, Loss function F, Private dataset D, Global epoch E_g, Local epoch E_l, Local learning rate β

Output : Global models w_{E_g}

Initialize Global models $w_0[NC]$

for *each global epoch* $r = 1, 2, ..., E_g$ **do**

  Server: **Broadcast** w_{r-1} to N clients

  for *each client* $n = 1, 2, ..., N$ ***in parallel*** **do**

    $i_n \leftarrow \text{argmin}_{j \in [NC]} F(w_{r-1}[j])$

    $s_n \leftarrow \{s_{n,j}\}_{j=1}^{NC}$ with $s_{n,j} = \mathbf{1}\{i_n = j\}$ or $\mathbf{0}\{i_n \neq j\}$

    $\Delta_n \leftarrow \text{LocalUpdate}(F, D[n], w_{r-1}[i_n], E_l, \beta)$

  for *each cluster* $j = 1, 2, ..., NC$ **do**

    $w_r[j] \leftarrow (\sum_{n=1}^{N} s_{n,j}\Delta_n)/(\sum_{n=1}^{N} s_{n,j})$

return Global models w_{E_g}

Algorithm 2: LocalUpdate

Input : Loss function F, Local dataset d, Global model w, Local epoch E_l, Local learning rate β

Output : Local model w_{E_l}

$w_0 \leftarrow w$

for *each local epoch i = 1, 2,..., E_l* **do**

  mini-batch b_i sampled from d

  $w_i \leftarrow w_{i-1} - \beta \frac{\Delta F(b_i, w)}{\Delta w}$

if *active attack* **then**

  return $\gamma(w_{E_l} - w) + w$

else

  return Local model w_{E_l}

2 RELATED WORK

In this section, we briefly introduce FL and CFL. Next, we explain the previous PIA on FL.

2.1 Federated Learning

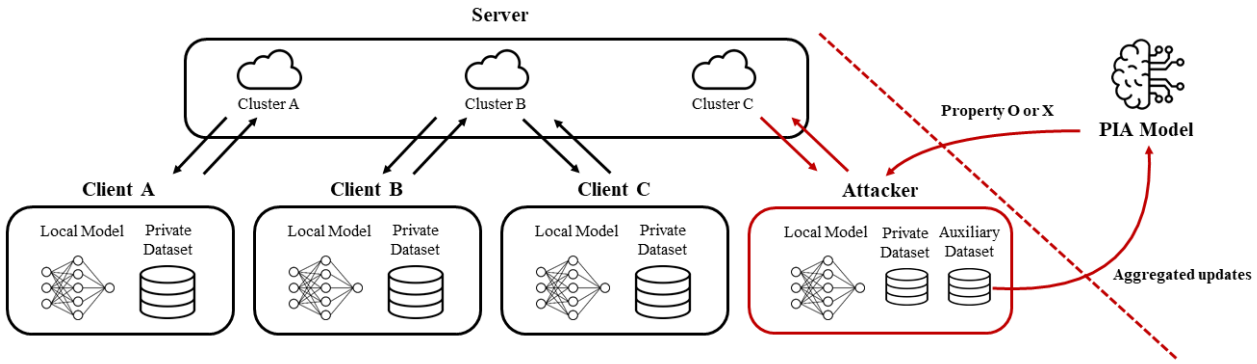
Due to the demand for learning over decentralized private data without any direct access, FL has been proposed and extensively studied recently [12, 16]. FL scheme involves a server and N clients, each having its own private dataset. FL consists of three main stages: 1) Model distribution: The server S distributes the global model w_{global} to the clients; 2) Local update: Each client uses its local data d_n to compute a local update Δ_{local} with the local epoch E_l; and 3) Aggregation: The clients send their Δ_{local} to the server. Then, the server aggregates the clients' Δ_{local} according to a given aggregation rule and updates w_{global} with the aggregated updates. We use FedAVG as the default aggregation rule, and FL iterates these three stages for global epoch E_g. The latest issue in FL is how to train a global model with non-IID data since data from individual devices are usually non-IID in the real world. A significant discrepancy amongst the clients' data distributions can lead to catastrophic forgetting and a failure of the global model's training [10, 23]. The following studies have focused on training multiple global models by clustering clients and assigning them to specific global models [8, 15, 21, 22]. The following subsection details these CFL studies.

2.2 Clustered Federated Learning

CFL is an FL technique designed to handle non-IID data by clustering clients so that the data distribution within each cluster is reasonably IID. Each cluster retains its own global model, and the server updates the global models using only clients that belong to that cluster. Google first introduces this idea [15] and devises a simple expectation-maximization-type (EM-type) algorithm to find appropriate clusters for each client. This algorithm assigns the best hypothesis for each cluster in terms of minimizing the weighted sum of each client's losses. In follow-up research, Sattler et al. [22] revise the central clustering method; the server collects local updates and computes the cosine similarity between them as a criterion for clustering. Ghosh et al. [8] change the location of the client clustering process from the server to the clients to reduce the burden of the server. After downloading all models from the server, each client chooses the cluster that shows the lowest loss value of the global model corresponding to the cluster with each client's current data batch. Clients may change their assigned clusters due to data sampling in batch process at each global epoch. As a result, the number of clients assigned to each cluster can vary across clusters. Ruan et al. [21] focus on the fact that some data distributions are composed of multiple mixtures of data distributions. This method assigns a global model to each data distribution instead of assigning a global model to each group of clients. Each cluster updates its global model with the stochastic mixture of clients' local updates according to their respective data distributions. Therefore, we can consider Ruan's method as an extension of Ghosh et al.'s [8] for non-IID environments. These CFL studies have primarily focused on improving the model's accuracy when trained with non-IID data while overlooking potential side effects. Following previous studies, we aim to examine the relationship between CFL and privacy leakages, focusing on the well-known CFL method [8] and PIA.

2.3 Property Inference Attack

PIA is a privacy leakage attack that infers the presence of unintended properties in the training data. Unintended properties refer to the properties that exist in specific subsets of the training data, but generally not in all training data and are also unrelated to the main task. Melis et al. [18] develop a passive PIA suitable for FL assuming that the attacker is one of the clients in FL and wants to know whether any clients use the data containing unintended property in a specific global epoch. The primary strategy is to generate simulated aggregated updates using an auxiliary dataset and train a PIA model with them to infer the presence of the unintended property. Specifically, since the global model is distributed to all clients, the attacker can compute the aggregated updates by subtracting



	Model Distribution	Local Update	Aggregation	Attack Phase
Server	- Broadcasts all cluster global models		- Aggregates the local updates	
Client	- Downloads the cluster models	- Performs local model update	- Uploads the local update to the server	
Attacker	- Downloads the cluster models and saves them	- Calculates the previous aggregated updates with saved cluster models - Generates training data for the PIA model with an auxiliary dataset	- Uploads the local update (passive) or malicious local update (active) to the server	- Trains the PIA model with simulated aggregated updates - Infers the unintended property

Figure 1: Overview of PIA against CFL.

the previous global model parameters from the current ones. To generate the simulated aggregated updates as the training data for the PIA model, the attacker simulates the local update algorithm 2 at every global epoch with the previous global model parameters and the auxiliary dataset. The PIA model is then trained with a set of simulated aggregated updates generated by the attacker to determine whether the aggregated updates contain unintended properties. This paper builds on the work of Melis et al. [18] for the CFL setting, as will be described further in section 3.

In addition to this passive PIA, Melis et al. [18] propose an active PIA that intentionally causes the global model to leak more information about the unintended property by injecting a malicious gradient using multi-task loss. To mitigate PIA, several studies have applied the DP techniques to FL. Since the nature of DP is to increase privacy while reducing utility, it is essential to find the appropriate level of DP noise so that the model can maintain its performance while preventing privacy leakage. However, previous studies have reported being unable to establish the proper DP level against PIA in a normal FL setting [18, 19].

3 PROPERTY INFERENCE ATTACK ON CFL

In this section, we first describe our base CFL model and present a threat model for PIA against CFL. Next, we describe the PIA against FL [18], outlining the modifications to adapt it for CFL. Then, we introduce our active attack designed to address the limitation that arises when the previous active PIA is adapted to CFL.

3.1 Base CFL Method

The base CFL algorithm used in this study [8] is depicted in algorithm 1. Similar to FL, this algorithm consists of three main parts: 1) Model distribution: All global models w are broadcasted to clients (Line 3 in algorithm 1); 2) Local Update: Each client determines the global model with the lowest loss value and sets its index as the cluster identity estimates i (Lines 4-5 in algorithm1). After determining the global model with the lowest loss value, each client initializes its local model with the corresponding global model w_i and trains it using the method outlined in algorithm 2 (Line 7 in algorithm 1); and 3) Aggregation: For each cluster, the server receives the local models from the cluster's clients, aggregates them by averaging, and generates the next global model for the cluster (Lines 8-9 in algorithm 1).

3.2 Threat Model

Our threat model follows the previous PIA studies [18, 19] with few modifications for CFL settings. We assume that K clients (where $K \geq 3$) participate in FL or CFL. One of these clients is the attacker, who wants to infer the presence of the unintended property in the training batch. The attacker is considered *honest-but-curious* and the server and clients strictly follow algorithm 1 without colluding with each other.

3.2.1 Attacker's objective. The attacker's objective is to identify the presence of the target unintended property in the clients' training data from the aggregated updates for each global epoch. In our threat model, since the server does not provide any other information to clients except the global models, the attacker can only see the aggregated updates, not the local updates. In this scenario, the

attacker attempts to infer the presence of the unintended property in the training data but does not link this information to a specific client.

3.2.2 Attacker's capabilities. In the prior PIA study on FL [18], the attacker is able to compute the aggregated updates by subtracting the last updated global model from the previous global model, since all clients including the attacker receive the single global model from the server on every global epoch. In CFL, we assume that the attacker has knowledge of the global model parameters as previous studies [8, 13, 14, 21] distribute all clusters' global model parameters to clients for efficiency. Since the attacker can access all global models at every epoch, the attacker can compute the corresponding aggregated updates for each cluster. Additionally, for training the PIA model, the attacker has an auxiliary dataset correctly labeled with the unintended property and the main task. The attacker has the capability to execute malicious actions during local computations. The attacker cannot violate the overall learning protocol. In such scenarios, the attacker can employ the active PIA approach, which distinguishes itself from the passive PIA approach by utilizing a multi-task loss instead of the base single-task loss used by the other benign clients during the local update process.

3.3 Attack Method

3.3.1 Overview of PIA. In this subsection, we first present a description of the PIA against FL proposed in the previous work [18]. We then explain how we adapt PIA for CFL, providing an overview of the PIA against CFL.

To elaborate on PIA against FL, during the model distribution phase, the server distributes the global model to the clients; meanwhile, the attacker downloads and saves it for further analysis. In the local update phase, the clients perform local model updates, and the attacker calculates the previous aggregated updates using saved global models by subtracting the current global model from the previous one. The attacker also generates training data (simulated aggregated updates) for the PIA model using an auxiliary dataset. In the aggregation phase, the clients transmit their local updates to the server. Meanwhile, in the case of an active attack, the attacker deliberately uploads malicious updates to the server. More details on active PIA will be provided in subsubsection 3.3.4. During the attack phase, the attacker trains the PIA model using the simulated aggregated updates and infers the unintended property from the actual aggregated updates.

Figure 1 provides a concise overview of PIA on CFL. For CFL, we define the attack model very similarly, with the following minor changes. The server sends multiple global models to the clients, one for each cluster. The attacker then stores these models and uses them to calculate aggregated updates for each cluster. The attacker also generates the simulated aggregated updates for each cluster. The attacker infers the unintended property using the single PIA model rather than the multiple PIA models according to each cluster. The rest of the process is the same as in FL. In the following subsections, we will explain more details about how to train the PIA model and how to infer the unintended property with the PIA model.

Algorithm 3: Training PIA model on CFL

Input: Number of clusters NC, Loss function F, Global epoch E_g, Local epoch E_l, Local learning rate β, The number of simulated clients it_{client}, The number of aggregation simulation it_{aggr}, Auxiliary dataset labeled with the unintended property D_{prop}, Auxiliary dataset labeled without the unintended property $D_{nonprop}$
Output: PIA model M

$G_{prop}[NC] \leftarrow \emptyset, G_{nonprop}[NC] \leftarrow \emptyset$
`/* training data for PIA model */`
Initialize PIA model M
`/* We use Random Forest as a PIA model */`
for *each global epoch* $r = 1, 2, ..., E_g$ **do**
  `/* line 2 in Algorithm 1 */`
  receive all clusters' global models w_{r-1} from the server
  estimate cluster identity i
  $\Delta \leftarrow \text{LocalUpdate}(F, D_{prop} \cup D_{nonprop}, w_{r-1}[i], E_l, \beta)$
  upload Δ to the server
  for *each cluster* $j = 1, 2, ..., NC$ **do**
    `/* generating PIA training data */`
    $g_{prop} \leftarrow \emptyset, g_{nonprop} \leftarrow \emptyset$
    for *each simulation iter* $t = 1, 2, ..., it_{client}$ **do**
      `/* LocalUpdate randomly samples each mini-batch from the dataset and returns different local updates at each iteration of simulation */`
      $\Delta_{prop} \leftarrow \text{LocalUpdate}(F, D_{prop}, w_{r-1}[i], E_l, \beta)$
      $\Delta_{nonprop} \leftarrow \text{LocalUpdate}(F, D_{nonprop}, w_{r-1}[i], E_l, \beta)$
      $g_{prop} \leftarrow g_{prop} \cup \Delta_{prop}$
      $g_{nonprop} \leftarrow g_{nonprop} \cup \Delta_{nonprop}$
    for *each simulation iter* $k = 1, 2, ..., it_{aggr}$ **do**
      $A_{prop} \leftarrow$ the simulated aggregated updates from $g_{prop} \cup g_{nonprop}$
      $A_{nonprop} \leftarrow$ the simulated aggregated updates from $g_{nonprop}$
      $G_{prop} \leftarrow G_{prop} \cup A_{prop}$
      $G_{nonprop} \leftarrow G_{nonprop} \cup A_{nonprop}$
Train PIA model M with $G_{prop}, G_{nonprop}$
return PIA model M

3.3.2 Training the PIA model. In FL, the attacker categorizes the auxiliary dataset into two groups: one that exhibits the unintended property and another that does not. Subsequently, the attacker operates the local update with each auxiliary dataset, generating simulated aggregated updates that correspond to whether or not the unintended property is present. In CFL, we can use a similar

training process, but with one important difference: the attacker generates the simulated aggregated updates separately for each cluster. A detailed explanation of this is as follows.

We define G_{prop} and $G_{nonprop}$ as sets of simulated aggregated updates with and without the unintended property data, respectively (Line 1 in algorithm3). To generate these updates, the attacker follows the CFL training process (Lines 3-7 in algorithm3) and iterates algorithm 2 to generate sets of local updates with and without the property data (Lines 8-14 in algorithm 3). The attacker samples some of these local updates from $g_{prop} \cup g_{nonprop}$ and aggregates them to create the simulated aggregated updates with the property data A_{prop} (Line 16 in algorithm 3), while the simulated aggregated updates without any property data, $A_{nonprop}$ are generated solely using $g_{nonprop}$ (Line 17 in algorithm 3). The attacker then accumulates A_{prop} and $A_{nonprop}$ into G_{prop} and $G_{nonprop}$, respectively (Lines 18-19 in algorithm 3). Then, the attacker trains the PIA model using the sets G_{prop} and $G_{nonprop}$ (Line 20 in algorithm 3).

3.3.3 Inference with the PIA model. In FL, the PIA model uses aggregated updates of each global epoch as input to identify the presence of the unintended property. By subtracting the global models from two consecutive epochs, the aggregated updates in the earlier epoch can be calculated. Additionally, Melis et al. [18] reported that the random forest model exhibits the best performance when used as a PIA model. For that, we also adopt the random forest model as the PIA model. In CFL, at each global epoch, the aggregated updates are not a single value because there are multiple aggregated updates corresponding to each cluster's global model. To handle each cluster's global model, we employ a single PIA model trained on all the simulated aggregated updates generated by each cluster instead of employing an individual PIA model for each. Thus, the single PIA model is used to predict the presence of the unintended property in every cluster.

3.3.4 Active attack. In this subsection, we describe how the previous active PIA, proposed by Melis et al. [18], works on FL and highlight its limitation when applied to CFL. Subsequently, we introduce an improved active PIA that addresses this limitation.

The key distinction between the active PIA and the passive PIA lies in the utilization of different loss functions. Instead of the basic single-task loss function, the previous active PIA uses the multi-task loss to manipulate the global model to have an improved internal separation of the unintended property. To compute their malicious local updates with the multi-task loss, the attacker extends its local model with a property classifier attached to the last layer. The multi-task loss is as follows:

$$L_{mt} = \alpha \cdot L(x, y; w_{local}) + (1 - \alpha) \cdot L(x, \hat{y}; w_{local}) \quad (1)$$

where x is the input labeled with the main task and unintended property, and w_{local} is the local model held by the attacker. y and $\hat{y}$ are the main task label and unintended property label, respectively. α is the multi-task loss weight that balances the weight of the two loss terms in the multi-task loss.

The limitation of the previous active PIA is that it may not be significantly better than the passive PIA in CFL setting. This is because the active PIA relies on the attacker's malicious local updates to directly impact the global model. However, in CFL, the reflection of malicious local updates on the global model occurs less frequently compared to FL, due to the consideration of multiple global models.

In that, we propose a method that increases the magnitude of malicious local updates. This idea is inspired by the work of Bagdasaryan et al. [2], which scales up the attacker's local update to replace the global model with one infected by backdoors. We attempt to replace the global model in each cluster with a model that heavily incorporates the unintended property. Our active PIA scales up the attacker's malicious local gradients during the local training phase before uploading the attacker's local updates. We include the aforementioned modification at lines 5-6 in algorithm 2. The attacker calculates the difference between the previous global model, w, and the local updates, w_{E_l} (which is the local model after local training). The difference is then increased by the scale-up parameter γ and added to the previous global model w. Finally, our malicious local updates are computed as $\gamma(w_{E_l} - w) + w$.

Although the frequency of impacting the global model is reduced by the nature of CFL, our active PIA increases the magnitude of the impact, leading to the replacement of the global model with one that incorporates the unintended property. Although the attacker manipulates the attacker's local updates, it is still "*honest-but-curious*" because it follows the whole training protocol precisely and does not tamper with the communications during training.

4 EXPERIMENTS

In this section, we describe the datasets, model architectures, and hyper-parameters used in our research and then compare passive PIA's performance in both FL and CFL, highlighting the weakness of CFL against passive PIA. Moreover, we evaluate the performance of the previous active PIA for FL and CFL and our enhanced active PIA for CFL. We then examine the reasons behind the increased vulnerability of CFL against PIA. For all experiments in this section, we designate one client as a victim who is the only one with the property data following previous studies [18, 19]. We use the AUC (area under the ROC curve) score as a metric for evaluating the model's performance on the main task and the PIA attack performance.

4.1 Experimental Settings

4.1.1 Datasets. We use the same datasets as those used in multi-party PIA [18]: the Labeled Faces in the Wild (LFW) dataset [9] and the Yelp-author dataset. LFW dataset is a face attributes dataset with 13,233 images covering 5,749 individuals, each of which has a size of 62x47 pixels and is labeled with multiple attributes such as gender, race, and whether the person is wearing glasses or sunglasses. the Yelp-author dataset is a subset of the Yelp dataset that covers healthcare service reviews tagged with ratings and attributes such as parking information, business category, and location. This sampled dataset includes 16,207 reviews written by the top 10 most participated reviewers.

For each dataset, we examine multiple combinations of the main task and the target unintended property inferred by the attacker, allowing us to evaluate various scenarios through experiments. These combinations are outlined in Table 2, and we provide an explanation of task combination A as an example. For the combination A, the main task is to classify the individuals' faces in the LFW dataset as either smiling or not smiling, while the unintended property

Table 1: Hyper-parameter settings for FL and CFL

Dataset	Global Epoch (E_g)	Local Epoch (E_l)	Batch Size	Learning Rate (β)	Multi-task Loss Weight (α)	Size of Root Dataset	Scale-up (γ)
LFW	5,000	1	32	0.01	0.9	13,233	5 or 10
Yelp-author	500	1	32	0.05	0.9	16,207	5 or 10

Table 2: Combinations of the main task and the target unintended property that is inferred by the attacker

Task Combination	A	B	C	D	E
Dataset	LFW			Yelp-author	
Main Task	Smile	Race	Gender	Review Score	Review Score
Main Task Labels	Not Smiling / Smiling	Asian / White / Black	Female / Male	From 0 to 5	From 0 to 5
Target Unintended Property	Wearing Sunglasses or Not	Wearing Sunglasses or Not	Race is Black or Not	Author of Review is Author 1 or Not	Author of Review is Author 2 or Not

refers to whether the individuals are wearing sunglasses or not. The attacker's PIA model performs a binary classification task that identifies whether there is an individual wearing sunglasses in the cluster.

For each combination, we generate a distinct experimental dataset using the same methodology as the previous study [18]. The original dataset is divided into a training set and a test set. The test set is used to evaluate the model's performance at the last global epoch. The training set is further divided into two datasets: one containing only data with the target unintended property ($Train_p$), and the other containing only data without the target unintended property ($Train_{np}$). Half of $Train_p$ is assigned to the attacker, while the remaining half is assigned to the victim. $Train_{np}$ is distributed to all clients, including the attacker and the victim. To create non-IID local datasets for each client, we employ the Dirichlet sampling method, as used in Wang et al. [28]. The attacker uses the assigned dataset as an auxiliary dataset for simulating the aggregated updates. As a result, the attacker possesses an auxiliary dataset that shares a similar data distribution with the victim's dataset.

4.1.2 Model architecture and hyper-parameters. In our experiments, we use the neural networks used in previous work [18]. Specifically, for LFW, we use a convolutional neural network (CNN) model with three spatial convolutional layers (32, 64, and 128 filters), each with a filter size of (3, 3) and subsequent max-pooling layers with a pooling size of 2. For Yelp-author, we use a GRU model with an embedding layer, a GRU layer, and a fully connected layer. The embedding layer embeds words to the embedding vectors with 100 dimensions, and the GRU layer maps an embedding vector sequence to a 128-dimensional hidden vector sequence. The last hidden vector in the hidden vector sequence is then fed to a fully connected layer for the final decision. For our active attack, we utilize grid search to determine an optimal scale-up parameter γ, which is set to either 5 or 10. The hyper-parameters used in our experiments are summarized in Table 1.

4.2 Attack Performance of Passive PIA

As shown in Table 3, we compare the passive PIA attack performance between FL and CFL. We vary the combination of the main task/unintended property, as presented in Table 2, and the number of clients to demonstrate its impact on the passive PIA attack performance. Note that the results in the table are averaged across five different random seeds due to the stochastic behavior of the clustering algorithm used in CFL. We conduct a total of 85 experiments, varying the number of clients from 3 to 20. Among the experiments conducted with the number of clients from 3 to 20, Table 3 shows the results for 5, 10, 15, and 20 clients. The rest of the test cases are consistent with Table 3, and the full table can be found in Appendix B. Our experiments show that passive PIA on CFL outperforms passive PIA on FL in 80 out of 85 experiments, with an average improvement of 30.32% in passive PIA attack performance. As previously stated in Melis et al. [18], some task combinations make it difficult to detect the presence of the unintended property. Additionally, the attack performance decreases as the number of clients increases. Our results indicate that passive PIA on CFL may not outperform passive PIA on FL in scenarios where the attack performance is limited due to the task combination's difficulty and the number of clients. Consequently, our experiments support that unintended property leakage is more pronounced in CFL. We further analyze the reasons for CFL's increased vulnerability to PIA and the limitations of PIA on CFL in comparison to PIA on FL in subsection 4.4.

4.3 Attack Performance of Active PIA

We evaluate the attack performance of the previous active PIA [18] for FL and CFL and our proposed active PIA for CFL, as presented in Table 4. We use combinations A, B, and C, and vary the number of clients to 5, 10, 15, and 20. For our proposed active PIA method, we report the most effective attack performance achieved by setting the scale-up parameter γ to either 5 or 10. Also, we note that our approach outperforms the previous active PIA method, regardless of the value of γ. In most cases, the attack performance of the previous active PIA method is higher when applied to CFL than when

Table 3: Passive Property Inference Attack with FL and CFL

	Attack Performance (AUC)									
Task Combination	A		B		C		D		E	
FL Method / # of Clients	FL	CFL	FL	CFL	FL	CFL	FL	CFL	FL	CFL
5	0.7326	**0.9424**	0.8154	**0.9684**	0.9297	**0.9769**	0.5302	**0.7570**	0.5764	**0.6937**
10	0.5957	**0.9009**	0.6145	**0.9371**	0.8429	**0.9084**	0.4421	**0.5975**	0.3051	**0.4763**
15	0.5488	**0.8284**	0.574	**0.8265**	0.7934	**0.8354**	0.4816	**0.6094**	**0.5526**	0.4064
20	0.5367	**0.7414**	0.5534	**0.7952**	0.6731	**0.8002**	0.4401	**0.5232**	**0.4933**	0.3808

Table 4: Active Property Inference Attack with FL and CFL

	Attack Performance (AUC)								
1-10 Task Combination	A			B			C		
FL Method / 1-10 & PIA / # of Clients	FL ([18])	CFL ([18])	CFL (Ours)	FL ([18])	CFL ([18])	CFL (Ours)	FL ([18])	CFL ([18])	CFL (Ours)
5	0.8560	0.9592	**0.9613**	0.8406	0.8992	**0.9324**	0.9681	0.9427	**0.9809**
10	0.6905	0.8939	**0.9042**	0.7035	0.8238	**0.8644**	0.7575	0.8871	**0.9532**
15	0.6219	0.8218	**0.8503**	0.6039	0.7563	**0.7632**	0.7870	0.8395	**0.9212**
20	0.5999	0.8023	**0.8141**	0.5602	0.7351	**0.7802**	0.7533	0.8411	**0.8951**

applied to FL, with an average improvement of 18.39%. However, when the number of clients is 5 and the task combination is C, the opposite is observed due to the limitation of the previous active PIA method on CFL. However, our proposed active PIA method shows higher attack performance when applied to CFL in this case. Moreover, our proposed active PIA method shows an average improvement of 4.14% compared to the previous active PIA method when applied to CFL. Therefore, these experimental results successfully demonstrate that our proposed active PIA method effectively addresses the limitation of the previous active PIA method on CFL.

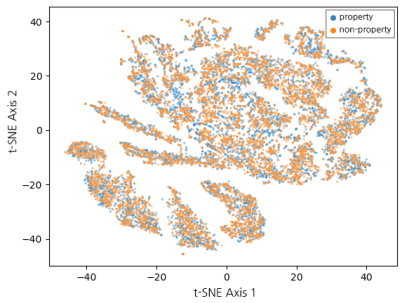
4.4 Analysis of PIA in the CFL Setting

We find that CFL is relatively weaker than FL against PIA due to the following reasons. Firstly, Melis et al. [18] reported that PIA achieves better attack performance when fewer clients participate in FL. In the case of CFL, clients are divided into multiple clusters, which can improve PIA performance by reducing the number of clients participating in a single global model. However, this also makes it more difficult for the PIA model to identify the pattern of unintended property due to multiple aggregated updates generated from multiple global models. Therefore, the decreased number of participating clients can be one of the reasons for the phenomenon but does not provide a comprehensive explanation of this phenomenon. To provide a more complete explanation, we analyze the difference between FL and CFL in terms of data distribution and catastrophic forgetting. When FL is trained over non-IID data, there can be a significant difference between the data distribution of the current and previous global epochs [10, 23]. Since the global model is fitted to the data batch of the previous global epoch, the current local updates, which are generated from the current data distribution, can result in catastrophic forgetting and drastic changes in the global model parameters. Although the gradient pattern generated by the unintended property in the input image is independent of the main task, the drastic changes in the global model parameters can result in an inconsistent pattern of unintended property in the aggregated updates. This inconsistency makes it difficult for PIA model to identify the specific pattern of unintended property. In contrast, in the case of CFL, the global model for each cluster is trained over relatively IID data. Therefore, the occurrence of catastrophic forgetting is reduced, and the global models are less likely to undergo excessive changes in parameters compared to FL. As a result, the pattern of the unintended property appears clearly in the aggregated updates, making it easier for PIA model to identify this pattern.

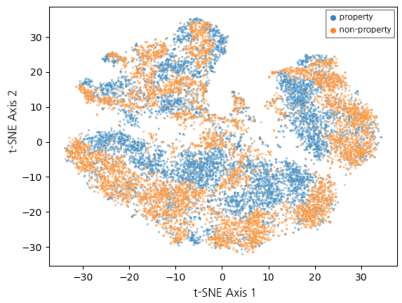
To support our explanations, we employ t-SNE [27] to visually depict the aggregated updates of each global epoch, demonstrating that the pattern of the unintended property appears clearer in CFL compared to FL. We then calculate the cosine similarity between the aggregated updates of two consecutive global epochs to demonstrate the changes in the aggregated updates during the training of the global model.

In Figure 2, we present the results of t-SNE when the number of clients is 5 and the task combination is C. We plot the aggregated updates generated during the first 1000 global epochs and all global epochs in Figure 2. As demonstrated in Figure 2, the label separation of aggregated updates generated in CFL is better than that of those generated in FL. This implies that the aggregated updates generated in CFL contain more distinct information about the unintended property. This information makes it easier for the PIA model to infer the presence of data with the unintended property.

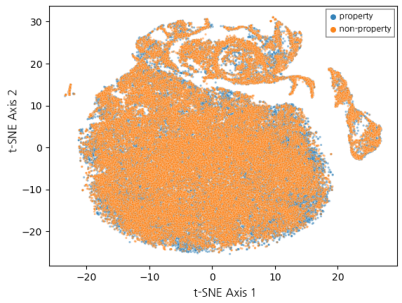
Figure 3 displays the aggregated updates when the number of clients is 13 and the task combination is D, where PIA on CFL



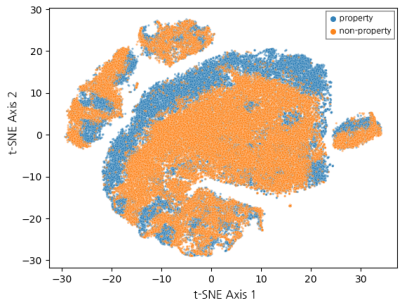
(a) FL, global epoch: 1–1000.



(b) Cluster 0 in CFL, global epoch: 1–1000.



(c) FL, global epoch: 1–5000.



(d) Cluster 0 in CFL, global epoch: 1–5000.

Figure 2: t-SNE visualized aggregated updates in each setting.

does not outperform PIA on FL. In this case, aggregated updates generated in CFL also show less label separation. The limitation of PIA on CFL is that the factors that degrade PIA's performance (e.g., the difficulty of task combinations, and the number of clients) may affect CFL more, resulting in lower PIA performance compared to FL.

We calculate the cosine similarities between aggregated updates generated at consecutive global epochs to measure the changes in aggregated updates at every global epoch with the same setting in t-SNE experiments. As depicted in Figure 4, cosine similarities in FL are much smaller than ones in CFL on average. The mean and standard deviation values of cosine similarities in FL are 0.3074 and 0.2752, respectively. In contrast, the mean and standard deviation values of cosine similarities of cluster 0 in CFL are 0.9234 and 0.1725, respectively. This result supports that aggregated updates change drastically at every global epoch in FL, making PIA against FL much harder.

5 DEFENSE

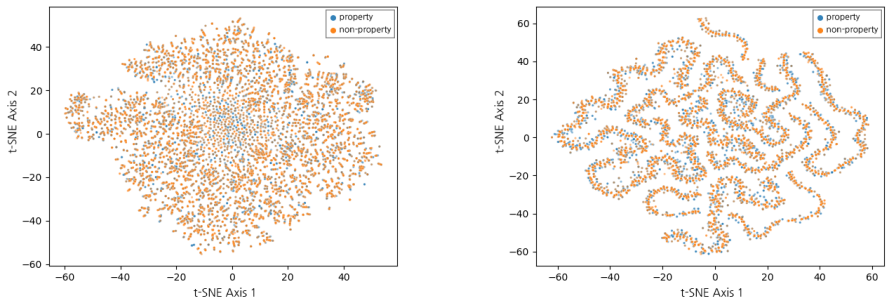
5.1 Existing Defense Techniques

We evaluate the effectiveness of existing defense techniques on CFL for task combinations A, B, and C by conducting experiments using two regularization methods: (i) adding dropout layers and (ii) using L2 regularization. These techniques have been hypothesized to be helpful in mitigating the impact of PIA, as they can reduce unnecessary information, which may include unintended property [11, 24, 29].

The number of clients in these experiments varies with 5, 10, 15, and 20, following the same experimental setup as the active PIA experiment in subsection 4.3. Our results show that both methods resulted in only a minor reduction or even an improvement in PIA performance, as shown in Table 5 and Table 6. We set the dropout rate to 0.1 and 0.2, and the L2 penalty to 0.01 and 0.02. Among the different hyper-parameters, the best PIA attack performances are highlighted in bold.

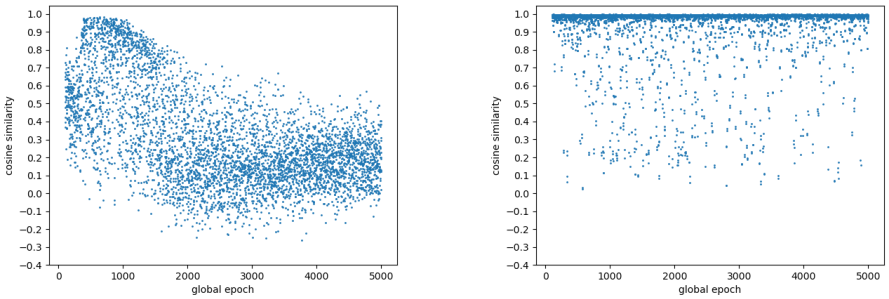
As shown in Table 5, for task combinations B and C, with a dropout rate of 0.1, which shows the worst PIA performance (and the best defense performance), the passive PIA performance is reduced by 14.85% and 5.6% on average, respectively, across all client numbers. However, for task combination A, the passive PIA performance increases on average by 1.26%. When L2 regularization is applied with an L2 penalty of 0.02 which shows the worst PIA performance, for task combinations B and C, the passive PIA performance is reduced by 8.08% and 6% on average, respectively. However, for task combination A, the passive PIA performance increases on average by 4.1%.

As shown in Table 6, using a dropout rate of 0.01 (which shows the worst PIA performance), the results for task combinations A, B, and C show a small reduction in active PIA performance by 3.03%, 9.4% and 4.1% on average, respectively. When L2 regularization is applied against our active PIA attack, we observe minor changes



(a) FL. (b) Cluster 0 in CFL.

Figure 3: t-SNE visualized aggregated updates when PIA shows low performance on CFL.



(a) FL, global epoch: 1–5000. (b) Cluster 0 in CFL, global epoch: 1–5000.

Figure 4: Cosine similarities between aggregated updates generated at consecutive global epochs in each setting.

in performance, with average decreases of less than 2% across all client numbers.

These results are highly unstable and vary depending on the task combinations. Even for the cases that appear to be effective, the reductions in attack performance are minimal. Thus, adding dropout layers or applying L2 regularization is not a reliable defense technique for CFL.

5.2 Applying DP on CFL

DP is a privacy-preserving technique that adds a certain level of noise to data. When DP is satisfied, it provides an upper bound on the difference between the DP technique's outputs when the DP technique's inputs differ by one element. In our paper, we use (ϵ, σ)-differential privacy [4].

In this section, we present an evaluation of DP against PIA in the context of CFL. To the best of our knowledge, this is the first work to investigate the trade-off of DP on CFL. In our paper, we consider the client-level DP (local differential privacy) [1, 3, 18]. To satisfy the DP definition at the client-level, we use DP-SGD [1] as a differential private deep learning optimization method. Specifically, in Line 1 of algorithm 2, we initialize the momentum accountant for tracking privacy loss. In Line 4, we substitute the local model update formula $w_i \leftarrow w_{i-1} - \beta \frac{\Delta F(b_i, w)}{\Delta w}$ with $w_i \leftarrow w_{i-1} - \beta \frac{1}{\|b_i\|_2} (\sum_{d \in b_i} \frac{\Delta F(d,w)}{\Delta w} min(1, \frac{C}{\| \frac{\Delta F(d,w)}{\Delta w} \|_2}) + N(0, \sigma^2 C^2 I))$ where C denotes a clipping parameter and σ denotes a noise scale parameter. Subsequently, the momentum accountant keeps track of the current privacy loss ϵ. After local training, the clients upload their noisy local updates to the server.

We evaluate the impact of DP on defense against PIA and model performance, as shown in Table 7 and Table 8, respectively. We consider different levels of privacy loss, ϵ: 9.23(0.5) and 33.13(0.2), with the corresponding DP noise scales σ indicated in parentheses. First, we examine the attack performance of PIA and find that the best attack performance among passive PIA and active PIA is 0.5111. This result indicates that it is difficult to infer unintended properties through PIA at these levels of privacy loss (ϵ = 9.23 or 33.13).

As previously noted, DP not only results in improving privacy but also leads to a decrease in model performance. Table 8 shows the model performance at each privacy loss level, with the value in parentheses indicating the rate of change after applying DP. With an ϵ of 9.23, both FL and CFL show significant declines in utility on average, with FL model performance dropping by 36.35% and CFL model performance decreasing by 13.96%. Although it is possible to secure FL and CFL against PIA with an ϵ of 9.23, it is still difficult to apply it due to the significant decrease in utility. In contrast, with an ϵ of 33.13, FL performance decreases by 19.55%, whereas CFL performance only experiences a minimal decrease of 3.51%. In more

Table 5: Passive Property Inference Attack with CFL after applying dropout and L2 regularization (lower means the better defense)

		Attack Performance (AUC)								
Method	Task Combination	A			B			C		
Dropout	Dropout Rate / # of Clients	0	0.1	0.2	0	0.1	0.2	0	0.1	0.2
	5	0.9424	0.9245	**0.9469**	**0.9684**	0.8705	0.8291	**0.9769**	0.8289	0.8374
	10	0.9009	0.9011	**0.9184**	**0.9371**	0.7784	0.8556	**0.9084**	0.8591	0.8587
	15	0.8284	0.8140	**0.8293**	**0.8265**	0.6907	0.7881	**0.8354**	0.8019	0.7849
	20	0.7414	**0.8054**	0.8015	0.7952	0.6685	**0.8110**	0.8002	**0.8176**	0.8120
L2	L2 Penalty / # of Clients	0	0.01	0.02	0	0.01	0.02	0	0.01	0.02
	5	0.9424	**0.9647**	**0.9647**	0.9684	**0.9806**	0.9783	**0.9769**	0.8933	0.8896
	10	0.9009	**0.9216**	0.9214	**0.9371**	0.8587	0.8546	**0.9084**	0.8768	0.8728
	15	0.8284	**0.8292**	0.8274	**0.8265**	0.7647	0.7709	**0.8354**	0.7910	0.7865
	20	0.7414	**0.8313**	0.8295	**0.7952**	0.6440	0.6538	**0.8002**	0.7679	0.7707

Table 6: Active Property Inference Attack with CFL after applying dropout and L2 regularization (lower means the better defense)

		Attack Performance (AUC)								
Method	Task Combination	A			B			C		
Dropout	Dropout Rate / # of Clients	0	0.1	0.2	0	0.1	0.2	0	0.1	0.2
	5	**0.9613**	0.9056	0.8575	**0.9324**	0.7662	0.7562	**0.9809**	0.9216	0.8770
	10	0.9042	**0.9090**	0.8861	**0.8644**	0.7588	0.7687	**0.9532**	0.8777	0.9098
	15	0.8503	0.8319	**0.8673**	0.7632	0.7374	**0.7695**	**0.9212**	0.9029	0.8839
	20	0.8141	0.8542	**0.9049**	**0.7802**	0.7611	0.7343	0.8951	0.8924	**0.9542**
L2	L2 Penalty / # of Clients	0	0.01	0.02	0	0.01	0.02	0	0.01	0.02
	5	**0.9613**	0.9133	0.8930	0.9324	**0.9724**	0.8659	**0.9809**	0.8933	0.8946
	10	0.9042	**0.9271**	0.8433	**0.8644**	0.8506	0.7578	**0.9532**	0.9196	0.9318
	15	0.8503	0.8336	**0.8833**	0.7632	0.8104	**0.8493**	0.9212	0.9464	**0.9702**
	20	0.8141	0.8055	**0.9166**	0.7802	**0.8392**	0.8255	0.8951	**0.9351**	0.9158

detail, with task combinations A, B, and C, the CFL performance decreases by 2.13%, 5.08%, and 3.32%, respectively. We observe that the impact of performance degradation is relatively minimal in CFL compared to FL, thanks to the consistent aggregated updates. As explained in subsection 4.4, the fluctuations in aggregated updates are far more pronounced in FL than in CFL, especially in terms of cosine similarity. Hence, DP, which shows a less negative effect on performance than FL, presents a suitable solution for preventing PIA on CFL.

6 DISCUSSION

In this section, we address the limitations of our work and potential future directions. One limitation of the PIA is that an auxiliary dataset is required to conduct the attack. Additionally, the effectiveness of PIA is heavily influenced by the number of clients and the task combination. We leave the analysis of this phenomenon as future work. Moreover, the attacker can only deduce the presence of the unintended property in the training batch (cluster-level) but not for a specific client. However, the attacker can leverage such information with other extra information [18] (e.g., clients participating and dropping out) to pinpoint the exact victim client. For instance, if a new unintended property is identified in a particular cluster's training batches, the attacker may conclude that such property is associated with one of the newly joined clients in the cluster. In more extreme settings [15, 22] where the server does not distribute all global models to the clients, it may not be possible to perform PIA unless the victim is in the same cluster as the attacker. However, in such cases, the attacker can utilize a global model acquisition attack that extracts all global models by analyzing the attacker's gradient and adjusting its assigned cluster. The details of this attack are documented in Appendix A. Future research could focus on developing a CFL method that enhances the performance of CFL and provides protection while inducing only minimal computational overhead. Furthermore, evaluating the potential risks of CFL against various privacy leakage attacks can also be a direction for future research. In summary, we claim that recent CFL studies have an internal vulnerability against PIA. We analyze the root cause of the vulnerability and suggest DP technique as a suitable defense solution. We emphasize that future research should not

Table 7: Attack Performance of FL and CFL after applying DP (lower means the better defense)

		Attack Performance											
ϵ	Attack Type	Passive PIA						Active PIA					
	Task Combination	A		B		C		A		B		C	
	FL Method / # of Clients	FL	CFL	FL	CFL	FL	CFL	FL	CFL	FL	CFL	FL	CFL
9.23	5	0.4973	0.4983	0.4918	0.4948	0.4944	0.5081	0.5038	0.5038	0.5031	0.4976	0.5040	0.4986
	10	0.4910	0.4974	0.4961	0.4992	0.5046	0.5026	0.5042	0.5049	0.4975	0.4924	0.5024	0.4992
	15	0.4963	0.4959	0.4945	0.5094	0.4990	0.5027	0.4989	0.5050	0.4953	0.4943	0.5039	0.4961
	20	0.4974	0.5037	0.5014	0.4966	0.5034	0.4951	0.4950	0.5027	0.5053	0.4919	0.5111	0.5063
33.13	5	0.4910	0.4960	0.4986	0.4933	0.5022	0.4969	0.5093	0.5001	0.5031	0.4976	0.496	0.4923
	10	0.5012	0.4976	0.4982	0.5055	0.4939	0.5036	0.5045	0.5066	0.4975	0.4924	0.5041	0.4986
	15	0.4961	0.5025	0.5069	0.4990	0.4979	0.5102	0.4983	0.4942	0.4953	0.4943	0.5081	0.4976
	20	0.5032	0.5058	0.4962	0.4957	0.5016	0.4935	0.4968	0.4952	0.5053	0.4919	0.4925	0.5029

Table 8: Model Performance and Its Change Rate on FL and CFL after applying DP (higher means the better utility)

		Model Performance					
ϵ	Task Combination	A		B		C	
	FL Method / # of Clients	FL	CFL	FL	CFL	FL	CFL
9.23	5	0.6119 (-35.68%)	0.9169 (-6.03%)	0.537 (-26.96%)	0.777 (-8%)	0.5489 (-25.22%)	0.792 (-1.42%)
	10	0.5003 (-47.66%)	0.8401 (-13.2%)	0.5039 (-38.8%)	0.7325 (-7.07%)	0.512 (-21.25%)	0.638 (-31.87%)
	15	0.4999 (-47.56%)	0.7992 (-17.42%)	0.5151 (-43.48%)	0.7661 (-12.5%)	0.4988 (-22.63%)	0.7714 (-19.43%)
	20	0.5049 (-46.9%)	0.8493 (-12.04%)	0.5059 (-38.68%)	0.7102 (-20.58%)	0.4942 (-41.35%)	0.7877 (-17.9%)
33.13	5	0.7786 (-18.16%)	0.9641 (-1.19%)	0.6577 (-10.55%)	0.822 (-2.68%)	0.6325 (-13.84%)	0.824 (+2.56%)
	10	0.8994 (-5.91%)	0.9447 (-2.4%)	0.5015 (-39.1%)	0.7496 (-4.91%)	0.6304 (-3.03%)	0.9181 (-1.95%)
	15	0.8786 (-7.85%)	0.9468 (-2.17%)	0.5501 (-39.64%)	0.8528 (-2.6%)	0.4994 (-22.54%)	0.9019 (-5.8%)
	20	0.8477 (-10.85%)	0.9389 (-2.75%)	0.5352 (-35.13%)	0.8037 (-10.12%)	0.6065 (-28.02%)	0.8816 (-8.11%)

only focus on improving the FL performance but also on ensuring clients' data privacy.

ACKNOWLEDGMENTS

This work was supported in part by the BK21 FOUR program of the Education and Research Program for Future ICT Pioneers, Seoul National University in 2023, in part by Inter-University Semiconductor Research Center (ISRC), in part by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2021-0-02068, Artificial Intelligence Innovation Hub) (No.RS-2022-00155966, Artificial Intelligence Convergence Innovation Human Resources Development (Ewha Womans University)) (No. 2022-0-00516, Derivation of a Differential Privacy Concept Applicable to National Statistics Data While Guaranteeing the Utility of Statistical Analysis), in part by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(IITP-2023-2020-0-01602) supervised by the IITP(Institute for Information & Communications Technology Planning & Evaluation).

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 308–318.

[2] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. 2020. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2938–2948.

[3] John C Duchi, Michael I Jordan, and Martin J Wainwright. 2013. Local privacy, data processing inequalities, and statistical minimax rates. *arXiv preprint arXiv:1302.3203* (2013).

[4] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. 2006. Our data, ourselves: Privacy via distributed noise generation. In *Advances in Cryptology-EUROCRYPT 2006: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28-June 1, 2006. Proceedings 25*. Springer, 486–503.

[5] Karan Ganju, Qi Wang, Wei Yang, Carl A Gunter, and Nikita Borisov. 2018. Property inference attacks on fully connected neural networks using permutation invariant representations. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*. 619–633.

[6] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. 2020. Inverting gradients-how easy is it to break privacy in federated learning? *Advances in Neural Information Processing Systems* 33 (2020), 16937–16947.

[7] Robin C Geyer, Tassilo Klein, and Moin Nabi. 2017. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557* (2017).

[8] Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. 2020. An efficient framework for clustered federated learning. *Advances in Neural Information Processing Systems* 33 (2020), 19586–19597.

[9] Gary B Huang, Marwan Mattar, Tamara Berg, and Eric Learned-Miller. 2008. Labeled faces in the wild: A database forstudying face recognition in unconstrained environments. In *Workshop on faces in'Real-Life'Images: detection, alignment, and recognition*.

[10] Wenke Huang, Mang Ye, and Bo Du. 2022. Learn from others and be yourself in heterogeneous federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10143–10153.

[11] Jinyuan Jia, Ahmed Salem, Michael Backes, Yang Zhang, and Neil Zhenqiang Gong. 2019. Memguard: Defending against black-box membership inference attacks via adversarial examples. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*. 259–274.

[12] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492* (2016).
[13] Chengxi Li, Gang Li, and Pramod K Varshney. 2021. Federated learning with soft clustering. *IEEE Internet of Things Journal* 9, 10 (2021), 7773–7782.
[14] Renhao Lu, Weizhe Zhang, Yan Wang, Qiong Li, Xiaoxiong Zhong, Hongwei Yang, and Desheng Wang. 2023. Auction-Based Cluster Federated Learning in Mobile Edge Computing Systems. *IEEE Transactions on Parallel and Distributed Systems* (2023).
[15] Yishay Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh. 2020. Three approaches for personalization with applications to federated learning. *arXiv preprint arXiv:2002.10619* (2020).
[16] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. PMLR, 1273–1282.
[17] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2018. Learning Differentially Private Recurrent Language Models. In *International Conference on Learning Representations*.
[18] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. 2019. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE symposium on security and privacy (SP)*. IEEE, 691–706.
[19] Mohammad Naseri, Jamie Hayes, and Emiliano De Cristofaro. 2022. Local and Central Differential Privacy for Robustness and Privacy in Federated Learning. In *Network and Distributed Systems Security (NDSS) Symposium 2022*.
[20] Vasyl Pihur, Aleksandra Korolova, Frederick Liu, Subhash Sankuratripati, Moti Yung, Dachuan Huang, and Ruogu Zeng. 2018. Differentially-private" draw and discard" machine learning. *arXiv preprint arXiv:1807.04369* (2018).
[21] Yichen Ruan and Carlee Joe-Wong. 2022. Fedsoft: Soft clustered federated learning with proximal local updating. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 8124–8131.
[22] Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. 2020. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE transactions on neural networks and learning systems* 32, 8 (2020), 3710–3722.
[23] Neta Shoham, Tomer Avidor, Aviv Keren, Nadav Israel, Daniel Benditkis, Liron Mor-Yosef, and Itai Zeitak. 2019. Overcoming forgetting in federated learning on non-iid data. (2019).
[24] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*. IEEE, 3–18.
[25] Jingwei Sun, Ang Li, Binghui Wang, Huanrui Yang, Hai Li, and Yiran Chen. 2021. Soteria: Provable defense against privacy leakage in federated learning from representation perspective. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 9311–9319.
[26] Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H Brendan McMahan. 2019. Can you really backdoor federated learning? *arXiv preprint arXiv:1911.07963* (2019).
[27] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).
[28] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. 2020. Federated learning with matched averaging. *ICLR 2020* (2020).
[29] Wenqi Wei, Ling Liu, Margaret Loper, Ka-Ho Chow, Mehmet Emre Gursoy, Stacey Truex, and Yanzhao Wu. 2020. A framework for evaluating client privacy leakages in federated learning. In *Computer Security–ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14–18, 2020, Proceedings, Part I 25*. Springer, 545–566.

A GLOBAL MODEL ACQUISITION ATTACK

In this section, we describe an attack method that acquires global models of CFL from the server that does not distribute all global models to the clients. The limitation of this attack is that it requires the participation of multiple attackers to obtain all global models. If the server changes the client's allocated cluster at every global epoch, the attacker can obtain all global models along multiple global epochs without the participation of multiple attackers, but cannot obtain the latest global model except the one obtained at the current global epoch. In this paper, we focus on Sattler et al.'s CFL method [22], and this attack method can be extended to other CFL methods with additional adaptation to their specific clustering method.

The attack process consists of two phases. In the first phase, the attacker aims to identify its initial cluster. This can be achieved by computing the local update using the attacker's auxiliary dataset or generating a random local update and sending the local gradient to the server. The server assigns a cluster to the attacker based on its criteria. In the second phase, the attacker seeks a local update that can make the attacker to be assigned to a different cluster. In Sattler et al. [22]'s CFL method, the server sends intermediate models to newly joined clients to examine the returned local updates for hierarchical clustering and classify which partition the clients belong to. For this CFL method, the attacker can compute the antiparallel vectors of the initial local gradient and local gradients computed from the intermediate models. The cosine similarities between each of these vectors and its original gradient are all -1. If the newly joined attackers send these vectors to the server, it may induce them to be assigned to other clusters because the server will conclude that their local gradients are significantly different from the initial local gradient. The attacker repeats the second phase until obtaining all global models.

B TABLE OF PASSIVE PROPERTY INFERENCE ATTACK

In this section, we include the complete table presenting the results of Passive PIA.

Table 9: Passive Property Inference Attack with FL and CFL (FULL)

	Attack Performance (AUC)									
Task Combination	A		B		C		D		E	
FL Method / # of Clients	FL	CFL	FL	CFL	FL	CFL	FL	CFL	FL	CFL
3	0.873	**0.9423**	0.9644	**0.9885**	0.9856	**0.9903**	0.5845	**0.7756**	0.6736	**0.8869**
4	0.8158	**0.9689**	0.8666	**0.9898**	0.9622	**0.984**	0.6709	**0.7491**	0.3793	**0.8119**
5	0.7326	**0.9424**	0.8154	**0.9684**	0.9297	**0.9769**	0.5302	**0.7570**	0.5764	**0.6937**
6	0.6815	**0.9371**	0.7587	**0.9587**	0.8544	**0.9732**	0.5422	**0.7320**	0.4495	**0.6570**
7	0.6695	**0.9167**	0.7693	**0.9569**	0.8045	**0.9578**	0.5365	**0.6976**	0.4614	**0.6502**
8	0.6369	**0.9211**	0.6861	**0.9606**	0.7797	**0.9421**	0.4045	**0.7574**	0.5049	**0.5437**
9	0.6589	**0.917**	0.6517	**0.903**	0.7451	**0.9095**	0.4883	**0.6217**	0.3605	**0.4670**
10	0.5957	**0.9009**	0.6145	**0.9371**	0.8429	**0.9084**	0.4421	**0.5975**	0.3051	**0.4763**
11	0.5942	**0.8703**	0.6233	**0.903**	0.8093	**0.9127**	0.4302	**0.6563**	0.2203	**0.4715**
12	0.5931	**0.8666**	0.6016	**0.8735**	0.7121	**0.8742**	0.4667	**0.6096**	0.2196	**0.4369**
13	0.5527	**0.8523**	0.5761	**0.8594**	0.7158	**0.8904**	**0.5575**	0.5529	0.3921	**0.4125**
14	0.5545	**0.8402**	0.5925	**0.8744**	0.7282	**0.8659**	0.5047	**0.5943**	0.3426	**0.4392**
15	0.5488	**0.8284**	0.574	**0.8265**	0.7934	**0.8354**	0.4816	**0.6094**	**0.5526**	0.4064
16	0.5503	**0.7959**	0.5746	**0.8228**	0.7179	**0.8136**	0.4632	**0.5798**	0.3483	**0.4125**
17	0.5303	**0.7754**	0.5666	**0.7819**	0.6376	**0.8112**	0.4343	**0.6150**	0.3435	**0.3601**
18	0.5078	**0.8187**	0.5492	**0.8063**	0.7103	**0.752**	0.5038	**0.5614**	**0.5343**	0.3653
19	0.5462	**0.7686**	0.5557	**0.8413**	0.7118	**0.8016**	0.4844	**0.4848**	**0.4151**	0.4069
20	0.5367	**0.7414**	0.5534	**0.7952**	0.6731	**0.8002**	0.4401	**0.5232**	**0.4933**	0.3808