Poisoning as a Post-Protection: Mitigating Membership Privacy Leakage From Gradient and Prediction of Federated Models

Gaoyang Liu, *Member, IEEE*, Tianlong Xu, Yang Yang, *Member, IEEE*, Ahmed M. Abdelmoniem, *Senior Member, IEEE*, Chen Wang, *Senior Member, IEEE*, and Jiangchuan Liu, *Fellow, IEEE*

***Abstract*—Federated learning (FL) is a distributed learning paradigm that enables multiple clients to train a unified model without sharing their private data. However, recent works demonstrate that FL models are vulnerable to membership inference attacks (MIAs), which can infer whether a data sample was used to train a given FL model. Existing countermeasures either require far-reaching modifications of FL training process or enforce extra processing in prediction phase, yielding them unlikely to be applied well in practice. In this paper, we design a post-protection mechanism, dubbed *P^2-Protection*, which degrades the inference performance of MIAs by simultaneously poisoning the prediction and gradient of the target FL model to reduce the privacy leakage of training data while keeping the model prediction accuracy. *P^2-Protection* only involves one additional training round to embed the poisoned prediction and gradient into the target FL model, without requiring model retraining or training process modification. We evaluate *P^2-Protection* and compare it with two state-of-the-art defenses against three MIAs on five realistic datasets. Experimental results show that *P^2-Protection* outperforms the existing defenses by offering limited implement overhead and improved utility-privacy trade-off.**

***Index Terms*—Federated learning, membership inference attack, poisoned prediction and gradient.**

I. Introduction

Deep learning (DL) has achieved extraordinary success in many fields, such as image classification and medical diagnosis [1]. Such success mainly relies on the large-scale training data generated from different domains, which may contain privacy sensitive information, including medical records or personal locations. However, the training process of traditional centralized DL requires collecting all relevant data from distributed sources to train the DL models, causing severe concerns about data privacy and user confidentiality [2], [3].

Recently, federated learning (FL) [4], [5] has emerged as a privacy-aware alternative to the decentralized learning paradigm. FL enables multiple clients (such as organizations or edge devices) to jointly train a unified model without their personal data leaving the local devices. This is achieved by the FL training process which consists of two main steps: training local models on each client's local data, and exchanging model updates (e.g., the parameters and gradient of the local model) between each client and the server to generate a unified FL model. Without sharing data, FL thus allows addressing critical issues such as data privacy, data leakage, and data access rights, and so it has been utilized in a series of applications in practice such as self-driving [6], medical research [7] and smart healthcare [8].

Unfortunately, recent studies have revealed that the training data information can be retrieved from the prediction and gradient of the trained FL models by various attacks, such as source inference attacks [9], property inference attacks [10], and attribute inference attacks [11]. In this paper, we focus on the so-called membership inference attacks (MIAs) against FL models, where the adversary aims to infer whether a given sample (i.e., the target sample) was used as part of the training data of a given FL model (i.e., the target model) [12], [13]. MIAs pose severe privacy and security threats to FL models. For instance, considering an FL diagnosis model trained on the data distributed across multiple medical institutions, if an adversary knows a target sample was used to train this model, then the attacker can directly infer the data owner's health status. Beyond privacy, since providing the training data is a product of massive costs and expertise efforts (including data preprocessing and annotation), MIAs also damage the intellectual property of the training data [14], [15].

To mitigate the risk of membership leakage in FL models, multiple countermeasures have been proposed. Since overfitting (i.e., the prediction gap between training and test data) is a major reason of membership information leakage [16], a widely recognized manner of defenses [17], [18] is to reduce the overfitting level by interfering in the training process of the target model. Other lines of defenses including using differ-

This work was supported in part by the National Natural Science Foundation of China under Grants 62272183, 62002104 and 62372343;by the National Key R&D Program of China under Grant 2024YFE0103800; by the Major Science and Technology Project of Hubei Province under Grants 2024BAA008 and 2024BAA011; by the Key R&D Program of Hubei Province under Grants 2024BAB016, 2024BAB031 and 2023BAB074; by the UKRI EPSRC Grant EP/X035085/1; and by the Open Project Funding of the Key Laboratory of Intelligent Sensing System and Security, Ministry of Education under Grant KLISSS202401. *(Corresponding author: Chen Wang.)*

G. Liu is with the Hubei Key Laboratory of Internet of Intelligence, School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, China. He was with the School of Computing Science, Simon Fraser University, Burnaby, BC V5A 1S6, Canada. Email: liugaoyang@hust.edu.cn.

T. Xu and C. Wang are with the Hubei Key Laboratory of Internet of Intelligence, School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, China. Email: {tianlongxu, chenwang}@hust.edu.cn.

Y. Yang is with the Key Laboratory of Intelligent Sensing System and Security, Ministry of Education, China, and the School of Artificial Intelligence, Hubei University, Wuhan 430062, China. Email: yangyang@hubu.edu.cn.

A. M. Abdelmoniem is with School of Electronic Engineering and Computer Science, Queen Mary University of London, UK. Email: ahmed.sayed@qmul.ac.uk.

J. Liu is with the School of Computing Science, Simon Fraser University, Burnaby, BC V5A 1S6, Canada. Email: jcliu@cs.sfu.ca.

ential privacy (DP) [19], [20], introducing feature obfuscation to the target model [21], or restricting the whole prediction functionality of the target model [16], [22]. Nevertheless, while these defense approaches can enhance the membership privacy of the target models, they often involve modifications to data features, model gradients, and prediction processes, which may compromise the model's predictive performance. Moreover, these countermeasures either require far-reaching modifications of FL training process, or enforce extra processing in prediction phase, yielding them unlikely to be applied widely in practice, particularly for FL models that have already been trained but still require protection.

In this paper, we propose an MIA defense for trained FL models, dubbed *P^2-Protection*, which simultaneously Poisons the prediction and gradient as a Post-Protection for trained FL models against MIAs. The success of current MIAs primarily stems from the inherent overfitting tendency of FL models, which results in distinct patterns in model predictions and gradients between training and testing data. MIAs exploit this discrepancy to breach the membership privacy of the target FL models. Therefore, the core idea of our proposed *P^2-Protection* is to obscure these prediction and gradient patterns between training and testing data, thereby increasing the difficulty of performing MIAs. Specifically, we introduce controlled perturbations to the predictions and gradients of the target model on its training data, aiming to maximize their deviation from the original values in order to protect membership privacy in FL models. To implement this, we perform an additional FL training round in which we embed carefully designed perturbations into the model. With our poisoned perturbations, *P^2-Protection* could mitigate the membership information contained in the model's prediction and gradient, thus significantly degrading the performance of existing MIAs while keeping the model prediction accuracy.

As a practical solution designed for MIA defense, *P^2-Protection* involves only one additional training round for FL models, and can be used for any trained FL models without retraining or modifications. We evaluate *P^2-Protection* by defending against three state-of-the-art MIAs, and the results demonstrate that our protection could alleviate the leakage risk of the membership privacy of FL models.

We summarize our major contributions as follows:

- We propose *P^2-Protection*, an after-training defense against MIAs for FL models, which does not require the modification of the existing FL training process or FL paradigm, and is able to defend against attacks with even white-box information.
- We exploit the intrinsic correlation between the prediction and gradient of the model, and unify gradient poisoning and prediction poisoning for FL models into a utility-constrained optimization problem. Compared with existing works, our method can achieve an improved utility-privacy trade-off of FL models.
- We evaluate *P^2-Protection* and compare it with two state-of-the-art defenses against three MIAs on five realistic datasets, and the results demonstrate the effectiveness and efficacy of our defense. The code has been released for reproducibility purposes[1].

The remainder of this paper is structured as follows. Section II presents some preliminary knowledge on FL and MIA. Section III describes the threat model. Section IV details the design of *P^2-Protection*. Section V provides the performance evaluation and Section VI reviews some related works. Finally Section VII concludes the paper.

II. Preliminary

A. Federated Learning

FL enables multiple clients to jointly train a model without sharing their data. In a typical paradigm of FL, there is a server S controlling the training process and M clients owning their own dataset $D_i (i = 1, 2, \ldots, M)$. The central server organizes the model training process, by repeating the following steps until the training is stopped:

Step 1. All clients download the current global model $f_t, (t = 0, 1, \ldots, T$, where T is the total number of FL training epochs) and the training setting from S.

Step 2. Each client trains its local model on D_i. After the local training, the client computes the model updates $\boldsymbol{w}_i$ and uploads $\boldsymbol{w}_i$ to S.

Step 3. S collects $\boldsymbol{w}$ from all clients and aggregates the client updates $\boldsymbol{w} = \sum_{i=1}^{M} \alpha_i \boldsymbol{w}_i$, where α_i is the client weight. Then S updates the global model f_t based on the aggregation and thereby obtains an updated global model f_{t+1}.

The above steps would be executed for T times until satisfying the termination criteria, and the obtained global model f_T is the final trained FL model.

Currently, FL has two main paradigms, namely the horizontal FL and the vertical FL [23], [24]. The former is applicable to the scenarios where datasets of clients share similar features but concern different users, while the latter is for situations in which datasets of clients concern the same users but share different features. Considering the wide application of the horizontal FL in practice, we focus on defending MIAs for the horizontal FL in our work.

B. Membership Inference Attacks

MIA aims to breach the membership privacy of the training data of a model, and was first introduced [25] in a centralized machine learning scenario. Formally, given a target model f and a data sample $\mathbf{x}$, MIA aims to determine whether $\mathbf{x}$ was used to train f or not. We denote a data sample as a member if it participates in the training process of the target model.

Existing MIAs can utilize the prediction or gradient of the target model to infer the membership information. More formally, the purpose of MIAs can be expressed as:

$$\mathcal{A}(\mathbf{x}, f|\Omega(f)) \rightarrow \textbf{In}/\textbf{Out} \tag{1}$$

where $\mathcal{A}$ represents MIA attacking functionality, $\Omega(f)$ means the obtained information about the target model, and **In** (resp. **Out**) indicates that $\mathbf{x}$ is a member (resp. non-member).

For gradient-based MIAs, the attacker can get the target model's structure, parameters, and intermediate calculations.

[1] https://www.dropbox.com/s/ue2r59dys453tva/P2-Protection-Code.zip

On contrast, prediction-based MIAs assume the attacker has no access to the internal information about the target model but can only get the prediction probability or even the prediction labels for an input sample. In this paper, we aim to poison both the gradient and prediction of the target model, thus defending against MIAs using different information of the target model.

III. Threat Model

Different from previous works concentrating on FL training process, we focus on the membership leakage in the post-training scenario of FL models, i.e., after the FL model is trained and released to FL clients. In such a scenario, previous defenses which aim to protect the privacy during the FL training phase may not defend against inference attacks when the model is released due to their in-training operations. Instead, we consider how each client can achieve a post-training protection for the published FL model without intervening with the post-training phase, e.g., perturbing on the output or restricting to Top-k probability outputs (we will discuss these defenses in Section V-A). As a defender, the client can only perform defending methodology during its local training phase, whose objective is to minimize the privacy leakage after the model is released. Our threat model are detailed as follows.

A. Attacker

Since our goal is to propose an MIA defense method for FL models, we consider a strong MIA attacker with the full access to the final trained FL model. Our assumed attacker, who could be either the FL server or an FL client, can obtain the FL model's predictions and gradients and then infer the membership information against other FL clients.

Attacker's Knowledge. In our paper, we assume that the attacker has full access to trained FL models, allowing them to obtain detailed information about the models, including their structure, parameters, intermediate computations such as activation values and gradients, and their predictions. Recognizing that most MIAs involve the construction of an inference model in a supervised manner, we further assume that the attacker has access to a subset of the training samples. It is practical since the FL server can access a portion of the training data from all clients, and the FL client has the access to its own local training samples. Note that this means the attacker has almost all the information one can obtain, and our method is able to defense against the strongest attacker.

Attacker's Capability. We assume that the attacker is either an FL server or an FL client, who can obtain the final trained FL model. Therefore, the attacker can use the trained FL model's query interface to obtain predictions for input samples. Moreover, the attacker can leverage the known internal details of this model to compute the gradients with respect to a given sample.

Attacker's Goal. Given a target FL model and a target sample, the attacker's goal is to determine whether that sample was utilized in the training of the model or not. Therefore, the essence of this objective is to perform a binary classification, categorizing the sample based on the available information from the target model into one of two classes: member or non-member. With the prediction and gradient of the target model w.r.t. the target sample, the attacker can construct a binary classifier to infer the membership property of the target sample. Formally, we have:

$$\mathcal{A}(\mathbf{x}; \mathcal{P}, \mathcal{G}) \rightarrow \textbf{In}/\textbf{Out} \tag{2}$$

where $\mathcal{A}$ is the inference model and $\mathbf{x}$ is the target sample. $\mathcal{P}$ and $\mathcal{G}$ is the target model's prediction and gradient w.r.t. $\mathbf{x}$. Furthermore, since the attacker could get a part of the training data of the target model, the inference model $\mathcal{A}$ could be trained with any supervised classification algorithms.

B. Defender

The defender aims to defend against MIAs from the prediction probabilities and gradients of the trained FL model to protect the membership privacy of FL clients' training data. In our proposed method, the defender is just the client who is sensitive to data privacy leakage and aims to protect the membership privacy of his local training data.

As a defender, each FL client could have the access to its local training data. As for the defender's capability, the client has sufficient computation capability for local training with local model, just as the routine FL setting. To enhance the privacy preserving capability of the global model, the defender can also launch another round of FL training after the routine FL training is finished. In the additional training round, the client could operate locally and submit elaborated parameters of local model to the server. Then the server collects the updated parameters from each client to update the global model. Throughout the entire process, it is essential to note that the protection mechanism is executed locally on the defender's device, and the server does not access the training data. Consequently, the privacy of local data is preserved in this mechanism, ensuring that our protection does not result in any information leakage.

Given a trained FL model, the defender aims to leverage data poisoning technique to achieve the following two goals.

- **Privacy Goal.** The inference model of MIA attacker is inaccurate at distinguishing the members from the non-members w.r.t. the given FL model.
- **Utility Goal.** The prediction performance of the protected FL model degrades within an acceptable range.

IV. Design of P^2-*Protection*

A. Overview

In order to defend against MIAs in FL, we design P^2-*Protection* from the perspective of poisoning the prediction and gradient of the target FL model that are leveraged by the attacker to construct the inference attack models. P^2-*Protection* adds controlled perturbations to the prediction of the target model on its training data, and embeds the perturbed prediction into the target model to poison the corresponding gradient. Then the federated central server collects the updates of the FL clients, and updates the FL model with one more round of training. Overall, P^2-*Protection* mainly includes the following two parts (c.f. Fig. 1 and Algorithm 1).

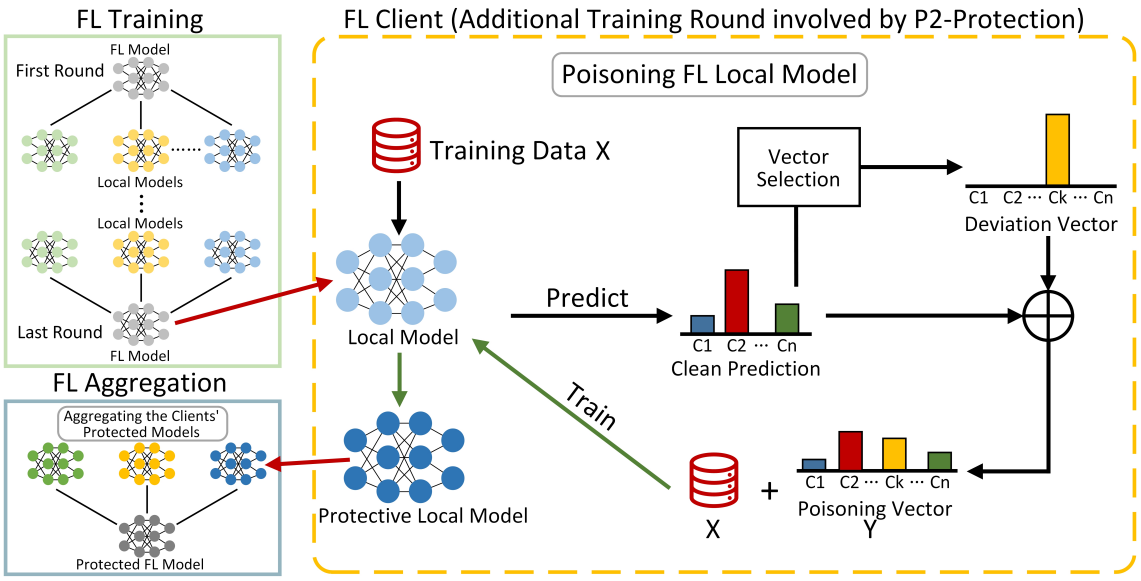


Fig. 1: The framework of P^2-*Protection*. After the FL training (the blue box at the upper left), each FL client downloads the current global FL model as its local model, removes its membership information locally, and uploads the poisoning updates (the right yellow box). Thereafter, the FL server collects and aggregates all updates from the FL client to form a secure FL model (the yellow box at the lower left). Note that P^2-*Protection* only runs one more round of FL training after the completion of the main FL training process.

(1) Poisoning Local Models. Given the trained FL global model f, each FL client first performs P^2-*Protection* to get the poisoned prediction $\mathbf{y}_{\text{poison}}$ of its training data. Then the FL client locally trains the current global model with its training data and the corresponding poisoned prediction, thereby poisoning the gradient of the training data concerning the current global model.

(2) Updating Global Model. After the first step of P^2-*Protection*, each client obtains a protected local model and derives the model update w.r.t. the current global model. Then the central server collects all the updates and renews the global model with enhanced protection of P^2-*Protection* from the aggregation of the poisoned updates.

B. Poisoning Local Models

Poisoning Prediction and Gradient. After completing the standard FL training process, P^2-*Protection* would be executed on every FL client first for protection. In particular, the i_{th} FL client initializes its local model $f^i(i = 1, 2, \cdots, M)$ with the downloaded global parameters $\mathbf{w}$ from the server, and then computes the clean prediction of f^i on its local training data $y_{clean} = f^i(\mathbf{x}) = (p_1, p_2, \ldots, p_C)$.

In the design of P^2-*Protection*, we integrate the gradient poisoning and prediction poisoning of an FL model in form of a constrained optimization problem, which is constrained by the model performance. It should be noted that the poisoned gradient is derived from the poisoned prediction w.r.t. the target model, which is the basis for our integrated poisoning of both gradient and prediction. In order to simplify our solution, we first define and find the deviation vector y_{dev} that can obtain the maximal gradient deviation from the gradient w.r.t. y_{clean} (c.f. line 6 of Algorithm 1). For simplicity, we assume that y_{dev} is the one-hot vector corresponding to the possible labels. We use L2 norm to measure the gradient deviation corresponding to y_{dev} and y_{clean}. Given y_{clean}, we design a simple maximizing gradient deviation **(MGD)** method to compute the deviation vector y_{dev} that satisfies:

$$y_{dev} = \arg \max_{o_k, o_k \in \mathbf{O}} \|\mathbf{g}_{clean} - \mathbf{g}_k\|_2 \tag{3}$$

where $\mathbf{g}_{clean} = \nabla\mathcal{L}(f(\mathbf{x}), y_{clean})$, $\mathbf{g}_k = \nabla\mathcal{L}(f(\mathbf{x}), o_k)$, $\mathcal{L}$ is the training loss function of FL training process, and $\mathbf{O}$ is a set that contains all possible one-hot formed vectors.

Then along the direction from y_{clean} to y_{dev}, we find a poisoning vector y_{poison}, as a post-training label of $\mathbf{x}$ (c.f. line 7 of Algorithm 1). Since poisoning the local model with y_{dev} directly may result in serious accuracy degradation but not contributes much to privacy protection, we combine y_{dev} and y_{clean} to get the poisoned prediction which can increase the difficulty of MIAs but maintain the performance of the FL model. The objective is formalized as follows:

$$\begin{aligned} &\arg \max_{y_{poison}} \|\mathbf{g}_{poison} - \mathbf{g}_{clean}\|_2 \\ &\text{where} \quad y_{poison} = (1-\alpha)y_{dev} + \alpha y_{clean} \\ &\qquad\quad \mathbf{g}_{poison} = \nabla\mathcal{L}(f(\mathbf{x}), y_{poison}) \\ &\qquad\quad \mathbf{g}_{clean} = \nabla\mathcal{L}(f(\mathbf{x}), y_{clean}) \\ &\quad s.t. \quad \alpha \in [0, 1] \\ &\qquad\quad \arg\max_k y^k_{clean} = \arg\max_k y^k_{poison} \end{aligned} \tag{4}$$

In the above equations, α controls the balance of y_{dev} (privacy) and y_{clean} (utility). To determine y_{poison}, we use binary search to find α. Specifically, the optimization problem aims to maximize the gradient deviation with respect to y_{poison}

and y_{clean}. Second, it ensures the prediction consistency of y_{clean} and y_{poison} for the premise that *P*^{2}*-Protection* does not change the prediction label of target model. In other words, the gradient of y_{poison} and y_{clean} are quite different but they bring about a similar accuracy effect to the model.

Training Local Models. After determining y_{poison} of each training sample, we next turn to embed the poisoned prediction and gradient into the local FL model, aiming to eventually remove the membership information of the training data. As such, *P*^{2}*-Protection* requires every client to perform local training using $(\mathbf{x}, y_{poison})$ to get the poisoned updates $\mathbf{w}^i_{poison}(i \in [1, 2, \cdots, M])$ (c.f. lines 8 and 10 of Algorithm 1), which can poison the target model by perturbing its gradient and prediction w.r.t. the training data.

C. Updating FL Global Model

Now that every FL client obtains a poisoning updates $\mathbf{w}^i_{poison}$ for local training data, embedding these updates into the FL global model could remove the sensitive membership information of original training data. The embedding process is the exact same as FL aggregation: The FL central server collects and averages the poisoning updates $\mathbf{W}_{poison} = [\mathbf{w}^1_{poison}, \mathbf{w}^2_{poison}, \cdots, \mathbf{w}^M_{poison}]$ from all FL clients. We denote the current global model parameters as $\mathbf{w}$. Then the server updates the current global model (c.f. line 18 of Algorithm 1) using:

$$\hat{\mathbf{w}} = \mathbf{w} + \sum_{i=1}^{M} \frac{n_i}{n} \mathbf{w}^i_{poison} \tag{5}$$

where n_i is the dataset size of the i_{th} client, $n = \sum_{i=1}^{M} n_i$, and $\hat{\mathbf{w}}$ is the parameters of global model $\hat{f}$ "poisoned" by our defense. After removing membership information from the global model, the global model can be released to the public without worrying about the membership leakage.

Discussion. *P*^{2}*-Protection* is a post-protection mechanism designed to defend against MIAs, which introduces an extra round of local model training and global model updating. Intuitively, our method could be extended to the centralized learning paradigm, given that the centralized paradigm can be viewed as a special case of FL involving only one client. However, directly applying our method to central models would incur substantial computational costs due to the necessity of poisoning the model for every training sample. Although we can choose to protect only a portion of training samples, the challenge lies in determining the criteria for selecting these specific samples. But in FL scenario, FL clients who are highly sensitive to privacy issues can choose to adopt our approach, which can significantly reduce the cost of our approach.

V. Performance Evaluation

A. Experimental Setup

1) Datasets and Models: We evaluate the performance of our defense using six publicly available datasets, distinguishing between independent and identically distributed (IID) and non-IID settings.

Algorithm 1 *P*^{2}*-Protection*

Require: the trained FL global model f
Output: the protected FL global model $\hat{f}$

1: ▷ ***FL client executes*:**
2: **for** each client $C_i, i \in \{1, 2, \cdots, M\}$ **in parallel do**
3: $D_{poison} = \emptyset$
4: **for** each local training data $(\mathbf{x}, y)$ **do**
5: $y_{clean} \leftarrow f(\mathbf{x})$
6: $y_{dev} \leftarrow \mathrm{MGD}(f, (\mathbf{x}, y_{clean}))$ // Get deviation vector with Eq. (3)
7: $y_{poison} = \mathrm{PoisonPrediction}(y_{dev}, y_{clean})$ // Get poisoning vector with Eq. (4)
8: $D_{poison} = D_{poison} \cup (\mathbf{x}, y_{poison})$
9: **end for**
10: $\mathbf{w}^i_{poison} = \mathrm{ComputeUpdate}(f, D_{poison})$ // Train local models
11: **end for**
12: **return** $\mathbf{w}^i_{poison}$
13: ▷ ***FL server executes*:**
14: $\mathbf{W}_{poison} = \emptyset$
15: **for** each client $C_i, i \in \{1, 2, \cdots, M\}$ **do**
16: $\mathbf{W}_{poison} = \mathbf{W}_{poison} \cup \mathbf{w}^i_{poison}$ // Collect updates
17: **end for**
18: $\hat{\mathbf{w}} = \mathbf{w} + \sum_{i=1}^{M} \frac{n_i}{n} \mathbf{w}^i_{poison}$ // Update model with Eq. (5)
19: **return** $\hat{f}$

Adult[2]. Adult contains 14 features (such as age, education, and gender) and a target which shows whether a person's income is over \$50K a year or not. It includes $48,842$ records and we use it in a binary classification task.

MNIST[3]. It is a handwritten recognition dataset that contains 10 classes of handwritten digits from 0 to 9. MNIST contains $70,000$ digits formatted as 28×28 gray images. The value of each pixel in the image is limited to $0 \sim 255$.

Purchases[4]. This dataset is based on Kaggle's "acquire valued shoppers" challenge dataset that contains shopping histories for thousands of individuals. It totally consists of $197,324$ shopping records from customers and each record contains purchase status of 600 kinds of products. We randomly choose $16,000$ data records. Following [16], [18], we use K-Means algorithm to cluster the dataset into 20 classes. These 20 classes are regarded as new labels.

CIFAR-10[5]. This dataset consists $60,000$ color images of 10 classes such as airplane, dog and bird. For each class, it has $6,000$ images of size 32×32.

FEMNIST[6]. Like MNIST, FEMNIST is a 62 classes handwritten dataset (10 for digits from 0 to 9 and 26 for both capital letter and lower-case letter, respectively) coming from $3,550$ users. It totally has $805,263$ images and each image has

[2]http://archive.ics.uci.edu/ml/machine-learning-databases/adult/
[3]http://yann.lecun.com/exdb/mnist/
[4]https://www.kaggle.com/c/acquire-valued-shoppers-challenge/data
[5]http://www.cs.toronto.edu/ kriz/cifar.html
[6]https://github.com/TalwalkarLab/leaf/tree/master/data/femnist

TABLE I: The model structures for each dataset.

Dataset	Model Structure
Adult	3 FC layers
MNIST	4 FC layers
Purchase20	4 FC layers
CIFAR-10	2 Conv., 2 Pool., and 3 FC layers
FEMNIST	4 FC layers
Celeb-A	VGG16

28×28 gray-scale pixels. We randomly select $70,000$ samples from FEMNIST.

CelebA[7]. CelebA is a large-scale face attribute dataset containing over $200,000$ celebrity images with annotations. In our experiments, we utilize it as a classification dataset by randomly selecting facial pictures from 100 users and evenly assigning them to various FL clients. We focus on four attributes (mustache, wearing a hat, wavy hair, and young) to categorize all pictures into 16 classes. Consequently, it can be regarded as a 16-classification task with distinct data distributions in each FL client.

For the IID datasets, including Adult, MNIST, Purchase, and CIFAR-10, we evenly distribute the dataset to each client so that there is no overlapping data between every two clients. For the non-IID datasets, including FEMNIST and CelebA, we employ a non-IID distribution by maintaining the original data distribution of each client, reflecting a more realistic scenario where data on different clients may have distinct characteristics. Furthermore, for all datasets, we randomly split each client's data equally into a training set and a test set.

As for the model structures, we investigate the deep neural network (DNN) and the convolutional neural network (CNN) on aforementioned datasets. Table I shows the model structure of each dataset. Particularly, the FC layer represents the fully connected layer in DNN, and the Conv. (resp. Pool.) layer means convolutional (resep. maxpooling) layer in CNN. For face image dataset CelebA, we use VGG16 [26] network.

2) FL and P^2-Protection Settings: We use FedAvg [27] algorithm to train FL models. We set the number of clients to 10 and the total global rounds to 10 unless otherwise specified. In every global training round, FL clients locally train their models for 10 epochs. For the local training, we use SGD [28] with the learning rate of 0.005 (FEMNIST with 0.05). Since *P^2-Protection* is a post-training defense mechanism, we configure it to perform only one additional global round of training, encompassing 10 epochs of local model training. Regarding the selection of the defense intensity value, the parameter α in Eq. (4) is not statically set but is dynamically optimized for each training sample from the FL clients. Following the procedure of our algorithm, we automatically determine the defense intensity values for the samples in all datasets used in our experiments.

3) MIAs: To quantify the robustness of *P^2-Protection*, we evaluate the performance against five typical MIAs.

[7]http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html

Shadow Attack (S-Attack) [16]. It trains multiple shadow models with the same structure as the target model to imitate the prediction behavior. Then the attacker trains 50 attack models on the gradient of shadow models to perform MIAs.

ML-Leaks [18]. Different from S-Attack, ML-Leaks builds multiple sub-shadow models of different algorithms, which further are combined as one shadow model. Then ML-Leaks trains one attack model on the gradient of this combined shadow model.

TEAR [12]. This attack leverages changes in adversarial robustness that occur during the training process of the FL model, and then constructs the attack model in a supervised manner.

Gradient Attack (G-Attack) [29]. It builds an attack model by using the target model's gradient, and predicts the cluster with a lower uncertainty as the member of the training set.

GD-Attack [30]. It calculates the difference between the norm of the global FL model's gradient updates and the norm of the gradients after excluding the target sample from the global model's updates. If this difference is greater than zero, the sample is predicted to be a member of the training set.

Note that the first three attacks utilize the prediction while the last two attack uses the gradient of the target model.

We perform all the above attacks on randomly selected samples from the FL clients' training and test datasets, where the number of members is set equal to the number of non-members, in order to achieve a baseline accuracy of 0.5, which is equivalent to the random guess.

4) Defenses: We compare our protection method with five existing defenses.

DP-SGD [19], [20], [31]. DP-SGD is a DP based protection, which introduces Gaussian or Laplacian noise to the parameter gradient during the FL training process. In this paper, we add Gaussian noise to the gradient updates, and adjust the ϵ value across the range of 0.001 to 1.0, capturing both the inference accuracy of G-Attack and classification accuracy of FL models. From the results obtained with different ϵ values, we select a preferred value, $\epsilon = 0.01$, striking a balance between fidelity and defense effectiveness. Besides, since DP-SGD requires involvement in the training process of FL models, we arrange for DP-SGD to participate in the entire training of the target FL models.

Top-*k* masking [25]. With top-k masking, the attacker can only get the top k prediction probabilities of the target model, which could reduce the information utilized by MIAs. Following Shokri et al. [25], we set k to 1 and 3.

Soteria [32]. Soteria is an FL training framework designed to prioritize the protection of user privacy embedded within the gradients. It generates a perturbation added to the activation values of FL models, ensuring that the perturbed data representations closely resemble the true data representations to preserve FL performance. Subsequently, the model updates are computed using these perturbed representations, making it challenging for an adversary to infer sensitive information from the updates. We set its pruning rate at 80%, which means all gradients are generated on the perturbed representation with a pruning rate of 80%.

TABLE II: Comparison of P^2-*Protection* with the baseline defenses against five MIAs. Note that Top-1 and Top-3 masking are ineffective against G-Attack, GD-Attack, and TEAR, as these attacks exploit model gradients and adversarial robustness, which can be derived from the Top-1 prediction probability.

Dataset	Defense Method	Prediction Accuracy	Attack Accuracy				
			S-Attack	ML-Leaks	G-Attack	GD-Attack	TEAR
Adult	No defense	0.843	0.52	0.52	0.625	0.663	0.690
	DP-SGD	0.762	0.50	0.50	0.609	0.647	0.665
	Top-1	-	0.50	0.51	-	-	-
	Top-3	-	-	-	-	-	-
	Soteria	0.801	0.51	0.51	0.578	0.638	0.644
	FedPass	0.835	**0.49**	0.51	0.590	0.626	0.640
	HAMP	0.821	0.50	0.52	0.580	0.623	0.631
	P*^{2}*-Protection	0.830	0.50	**0.50**	**0.574**	**0.605**	**0.620**
MNIST	No defense	0.947	0.52	0.51	0.620	0.669	0.703
	DP-SGD	0.777	0.48	0.48	0.588	0.630	0.652
	Top-1	-	0.46	0.39	-	-	-
	Top-3	-	0.49	0.49	-	-	-
	Soteria	0.824	0.47	**0.47**	0.572	0.610	0.623
	FedPass	0.923	0.48	0.49	0.581	0.621	0.637
	HAMP	0.890	0.48	**0.47**	0.554	0.573	0.605
	P*^{2}*-Protection	0.937	**0.46**	0.48	**0.540**	**0.564**	**0.674**
Purchase20	No defense	0.898	0.58	0.53	0.648	0.690	0.702
	DP-SGD	0.745	**0.50**	0.50	0.589	0.621	0.637
	Top-1	-	**0.50**	**0.48**	-	-	-
	Top-3	-	0.53	0.51	-	-	-
	Soteria	0.805	0.51	0.52	0.565	0.593	0.606
	FedPass	0.871	0.52	0.54	0.578	0.602	0.611
	HAMP	0.853	0.51	0.50	0.539	**0.567**	**0.582**
	P*^{2}*-Protection	0.879	**0.50**	0.50	**0.531**	0.571	0.653
CIFAR-10	No defense	0.921	0.52	0.51	0.641	0.690	0.713
	DP-SGD	0.835	0.51	0.50	0.625	0.652	0.660
	Top-1	-	**0.49**	**0.49**	-	-	-
	Top-3	-	0.52	0.51	-	-	-
	Soteria	0.865	0.50	0.51	0.603	0.634	0.639
	FedPass	0.913	0.50	0.51	0.594	0.641	0.657
	HAMP	0.885	0.50	0.50	**0.576**	0.613	**0.620**
	P*^{2}*-Protection	0.904	0.50	0.58	0.605	**0.613**	0.681
FEMNIST	No defense	0.720	0.55	0.53	0.643	0.680	0.704
	DP-SGD	0.577	0.48	0.46	0.605	0.631	0.651
	Top-1	-	0.47	**0.33**	-	-	-
	Top-3	-	0.49	0.48	-	-	-
	Soteria	0.658	0.49	0.48	0.602	0.627	**0.635**
	FedPass	0.709	0.49	0.49	0.607	0.630	0.648
	HAMP	0.698	0.48	0.45	0.583	0.598	0.674
	P*^{2}*-Protection	0.712	**0.47**	0.46	**0.582**	**0.589**	0.692
CelebA	No defense	0.856	0.65	0.67	0.836	0.874	0.892
	DP-SGD	0.785	0.58	0.58	0.775	0.824	0.854
	Top-1	-	0.58	0.57	-	-	-
	Top-3	-	0.62	0.60	-	-	-
	Soteria	0.812	0.56	0.57	0.759	0.742	0.760
	FedPass	0.850	0.58	0.60	0.734	0.746	0.768
	HAMP	0.827	0.58	0.56	0.667	0.707	**0.714**
	P*^{2}*-Protection	0.845	**0.54**	**0.53**	**0.658**	**0.683**	0.873

* The **bold** numbers represent for the best defense performance, while the underlined numbers indicate the highest prediction accuracy among the defended models.

FedPass [21]. FedPass introduces an adaptive obfuscation module integrated within the FL models. It dynamically adjusts the obfuscation during the training process to protect sensitive features of the training data. Following the settings of FedPass, for all datasets, we add an FC layer and a normalization layer at the input of the corresponding model architecture. The output dimension of the FC layer is kept consistent with the dimension of the input data, ensuring that the transformed data can still be processed by the models outlined in Table I.

HAMP [22]. HAMP protects the membership privacy of training data by pre-assigning high-entropy soft labels to the training data and training the model with these modified labels, thereby causing the model to exhibit less certainty in its predictions on the training data. By doing so, it can enforce the protected models to behave similarly on the training and testing samples. In our experiment, we set the default value of the entropy threshold parameter γ of HAMP to 0.5 for all datasets.

B. Performance

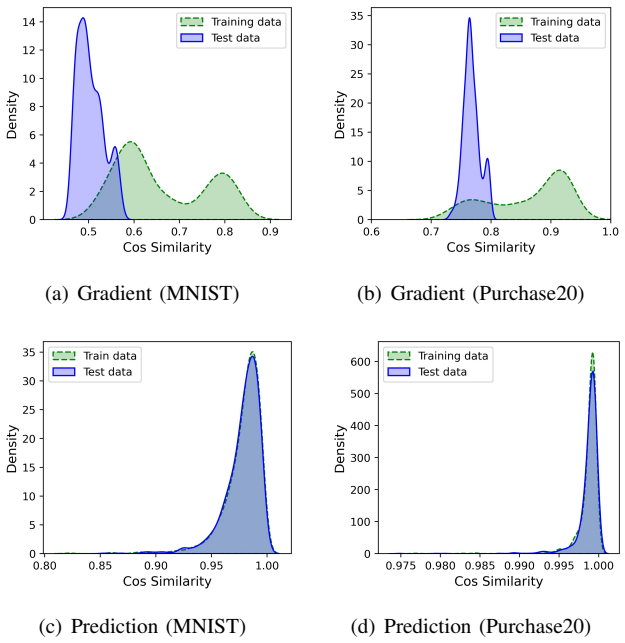
We first test three attack methods against our defense on the five datasets. We implement P^2-*Protection* to all layers in the target models. The results shown in Table II demonstrate that P^2-*Protection* could achieve the following three goals.

Effectiveness. An effective defense mechanism should alleviate the leakage risk of MIAs from FL models. We compare our P^2-*Protection* with three defense methods against

three types of MIAs. The results are shown in Table II. From the results we can see that P^2-*Protection* and the six comparisons have similar performance when defending S-Attack and ML-Leaks. However P^2-*Protection* outperforms all six comparisons when confronting the last three more powerful attacks. For instance, on Purchase20, P^2-*Protection* can decrease the attack accuracy of G-Attack from 0.648 to 0.531 (i.e., by 18.1%). In contrast, DP-SGD only decreases the attack accuracy by 9.1%, about half as much as that of P^2-*Protection*. It seems counterfactual that DP-SGD works even worse despite it operates in each aggregation while P^2-*Protection* only operates at the last aggregation. The reason is that the additive Gaussian noise introduced to each local model gradient may counteract each other. GD-Attack and TEAR perform slightly better than G-Attack. However, P^2-*Protection* still manages to defend against them, achieving a reduction in attack accuracy of over 0.8 across multiple datasets. For Soteria, it is apparent that when confronted with G-Attack, its performance is inferior to that of P^2-*Protection* across all datasets, as Soteria distorts the activation values of FL models and modifies the corresponding gradient updates. However, throughout the FL training process, the gradient related to the perturbed activations of member data still tends to converge toward a specific value and direction. This convergence poses a risk of potential membership information leakage for the gradient protected by Soteria. HAMP is the closest to P^2-*Protection*, with attack accuracy gap of 0.10 or less under multiple attacks.

Nevertheless, from the experiment results we can find an abnormal phenomenon: when confronted with TEAR, P^2-*Protection* exhibits a poor performance. This is because TEAR works during the model training phase, based on the relationship between samples and the model's decision boundary. However, P^2-*Protection* does not intervene in the training process of the model. Therefore, P^2-*Protection* has no significant defensive effect against TEAR.

Fidelity. The fidelity property requires that a defense mechanism preserves the utility of the protected model, minimizing its impact on the prediction accuracy of FL models. The third column of Table II shows the prediction accuracy of FL models before and after undergoing protection with different defenses. The results reveal that P^2-*Protection* excels in maintaining the highest fidelity for the target FL models. Specifically, our method induces a minimal accuracy reduction of 0.01 for MNIST, 0.019 for Purchase20, 0.008 for FEMNIST, and 0.011 for CelebA. In contrast, DP-SGD leads to a significantly larger accuracy decline for these datasets, which is $10\times$ greater than that observed with P^2-*Protection*. This phenomenon is also observed in the case of Soteria. The main reason primarily stems from the utility constraint, as outlined in Eq. (3), which we incorporate into the prediction perturbation. FedPass and P^2-*Protection* have minimal impact on the model performance. However, FedPass exhibits significantly worse robustness against inference attacks compared to P^2-*Protection*. This indicates that **P2-Protection** achieves a better trade-off between effectiveness and fidelity. Many existing defenses neglect the perturbation of FL model predictions concerning model performance. For instance, DP-SGD randomly generates a perturbation noise and then adds it to the gradient updates of FL models. In contrast, our P^2-*Protection* strategically selects a direction that optimally preserves model performance while altering the internal parameters of the model. In reality, a randomly chosen direction may not be the most effective for privacy preservation. The results provide insight that by controlling the direction of the added perturbation, we can mitigate the impact on model performance while achieving the same level of protection.



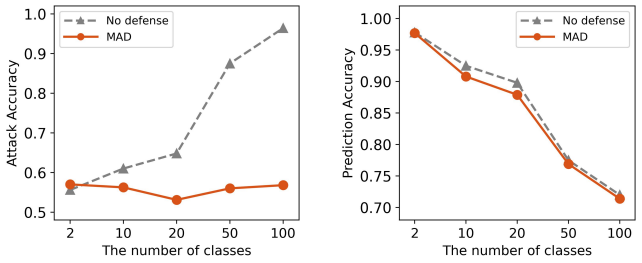
(a) Gradient (MNIST) (b) Gradient (Purchase20)

(c) Prediction (MNIST) (d) Prediction (Purchase20)

Fig. 2: The direction deviation of model gradient and prediction involved by P^2-*Protection*.

Furthermore, in order to show what P^2-*Protection* does to the model, we compare the direction change of gradient and prediction of the target model with and without P^2-*Protection*. Specifically, we use cosine similarity to measure the direction change. A cosine similarity close to 1 means that the change which P^2-*Protection* brings to the protected model is small. Fig. 2 presents the distribution of cosine similarity about gradient and prediction for two datasets. From the experiment results, we can find that for the direction deviation of model gradient, training data distributes more widely while test data has a huger direction deviation. Besides, the prediction deviations of both training data and test data are not remarkable. It seems abnormal that P^2-*Protection* causes more change to the gradient for test data than training data. We attribute this to the intrinsic property of model training: the model converges towards the direction which decreases the loss function for training data. This offsets against the mechanism of P^2-*Protection*, since P^2-*Protection* aims to deviate the gradient direction w.r.t the training data.

Feasibility. A part of existing MIA defenses require full knowledge of models (including training algorithm, model type, and hyper-parameters) and protected data (including sta-



(a) Attack Accuracy (b) Prediction Accuracy

Fig. 3: The impact of the number of classes.

tistical information and a part of data samples), which may not be available in practice. As a practical defense, P^2-*Protection* is similar to DP-SGD which uses perturbation to alleviate the leakage risk of membership information. One major difference is that P^2-*Protection* is a post-training defense while DP-SGD operates during the FL training phase. In this sense, P^2-*Protection* could also protect the normal FL model by adding an extra round of training. As shown in Table II, the accuracy decline of DP-SGD is more than $10\times$ than that of P^2-*Protection*. The balance of privacy and utility is a challenge for the DP-SGD mechanism. To select a preferable privacy budget, the whole FL training may be conducted many times. On the contrary, P^2-*Protection* achieves an improved trade-off between the perturbation and the model performance with the help of the prediction accuracy constraint.

C. Impact of Number of Classes

It can be seen in Table II that a complex dataset (i.e., the dataset with high input or output dimensions) suffers from more risk of information leakage. To study the impact of the number of dataset classes on P^2-*Protection*, we use K-Means algorithm to split the original Purchase dataset into 2, 10, 20, 50 and 100 classes. Besides using P^2-*Protection*, we also set a baseline that does not introduce any defense. Fig. 3 shows the attack accuracy and prediction accuracy as a function of the number of classes for Purchase dataset. As can be seen, with the increasing number of classes, the prediction accuracy decreases gradually for both methods. In addition, the attack accuracy of P^2-*Protection* stays at around 0.57 while the attack accuracy of no defense has a rapid increase, and is much higher than that of P^2-*Protection*. This means that P^2-*Protection* is more sensitive to the prediction accuracy rather than the attack accuracy. In general, a classification task with more classes implies more internal information about the dataset. This is largely because the number of neurons of the last layer (which contains more membership information compared with other layers) is proportional to the number of classes. However, our P^2-*Protection* can restrain the MIA destructiveness resulting from the increasing classes.

D. Impact of Protected Layers

In order to evaluate the impact of the protected layers, we apply P^2-*Protection* to different layers and measure the defense performance against G-Attack. Note that no matter which layer our defense is deployed to, G-Attack always uses the gradient of the whole model parameters to perform MIA. As shown in Table III, there is no significant difference on the attack accuracy for different protected layers. One interesting observation is that implementation in the latter layers brings about a more accurate prediction performance while keeping a similar defense effect. For MNIST, the prediction degradation of protecting the last layer is 0.001, much smaller than that of the first layer or the second layer. We conjecture that latter layer reveals more information about the training data. The result indicates that deploying P^2-*Protection* on latter layers can get a better trade-off of the prediction accuracy and defense effect for the FL models.

TABLE III: The impact of the number of protected layers on the defense performance of P^2-*Protection*.

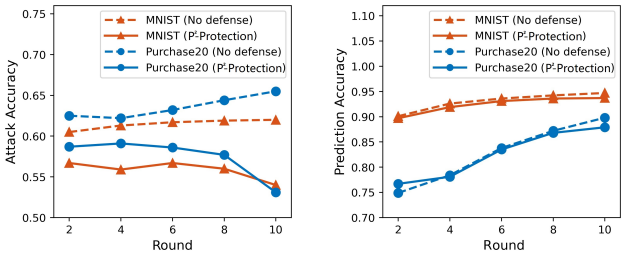
Protected Layers	Attack Acc.		Predic. Degrad.	
	MNIST	Purchase20	MNIST	Purchase20
-	0.620	0.648	-	-
Last Layer	0.560	0.586	0.001	0.016
Forth to last layer	0.558	0.570	0.003	0.018
Third to last layer	0.547	0.539	0.006	0.018
Second to last layer	0.542	0.538	0.005	0.018
First to last layer	0.540	0.531	0.010	0.019

We also investigate the impact of the number of protected layers on the defense performance of P^2-*Protection*. We first deploy P^2-*Protection* into the last layer of the target model and then gradually increase the number of protected layers. We report the attack accuracy for the models trained on both MNIST and Purchase20 in Table III. From the experiment results we can find that the more layers are protected, the stronger defense capability the model has. However, the prediction performance of the protected model is almost not affected by the number of protected layers.

E. Impact of Overfitting Level

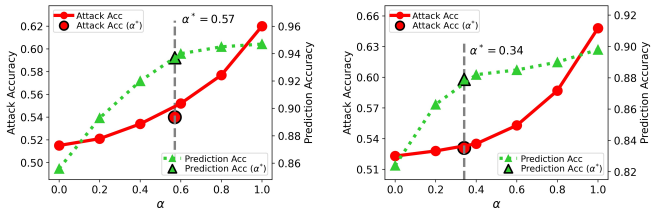
Previous studies have revealed the importance of model overfitting level to a successful MIA [16]. Hence we assess P^2-*Protection* under different overfitting levels. To achieve this, we set the total rounds of aggregation to 2, 4, 6, 8, and 10 respectively, and then perform G-Attack to evaluate the defense performance of our defense. Naturally, the more training rounds, the more severe the overfitting level. As shown in Fig. 4(a), with the increasing overfitting level, the gap of attack accuracy between the FL model with and without P^2-*Protection* becomes larger. Even though the attack accuracy for these datasets reaches the peak at the 10-th round, we can see a large degradation of the attack accuracy, which demonstrates that P^2-*Protection* also works more efficiently in an overfitting situation. In Fig. 4(b), we can observe that the dashed line (without defense) and the solid line (P^2-*Protection*) are so close. Therefore, P^2-*Protection* almost does not affect the prediction functionality of the model in the training phase.

The main reason for the phenomenon in Fig. 4(a) is that an overfitted model overly fits to its training data, leading to



(a) Attack Accuracy (b) Prediction Accuracy

Fig. 4: The attack accuracy and prediction accuracy for FL models with different overfitting levels.



(a) MNIST (b) Purchase20

Fig. 5: The attack accuracy and prediction accuracy for FL models vs. defense intensity. The gray dash line locates α^* obtained by binary search. Bigger markers with black edge represent the attack/prediction accuracy when $\alpha = \alpha^*$.

substantial differences in gradients and predictions between the training and testing data. Thus the gradients and predictions would contain a wealth of membership signals, which can facilitate the execution of MIAs. However, P2-Protection disrupts this by introducing perturbations to the gradients and predictions for each training sample, causing them to deviate from their original values. This deviation reduces the membership information carried by the gradients and predictions. The more overfitted the model is, the more certain its gradients and predictions are on the training samples; hence, a small perturbation can greatly destroy the membership signals they contain. Therefore, as the degree of model overfitting increases, the more membership information is contained in the gradients and predictions, our method becomes more effective at removing this membership information, thus improving the performance of defenses against MIAs.

F. Impact of Defense Intensity

In the design of *P^2-Protection*, the defense intensity α in Eq. (4) is determined by identifying an optimal value that maximizes the gradient deviation while preserving the prediction label of the protected FL model on the FL client's training data. The chosen intensity α significantly influences the utility of the FL model and the privacy of the FL client. In an ideal scenario, as α approaches 0, the poisoning degree of the target model increases, thereby exerting a more substantial impact on the prediction and gradient of the FL models. This, in turn, results in diminished prediction accuracy but a reduced risk of membership information leakage.

In order to better understand the defense intensity α, it is necessary to discuss the impact of α on the defense performance and the prediction performance of the protected model. Specifically, we manually vary α from 0 to 1 with an interval of 0.2 and evaluate the corresponding performance of *P^2-Protection*. The results against G-Attack on two datasets are shown in Fig. 5. From the results we can see that larger α would bring in better prediction utility but weaker defense ability. However, we can find that the prediction accuracy and attack accuracy vary with α in different ways. This observation suggests that by comparing the gradient of two lines, we can choose a certain value of α that satisfies a slowly rising prediction accuracy before a rapid increase in attack accuracy to get a better trade-off between prediction and attack accuracy. By using binary search, we can also achieve a preferable cost-effective privacy and utility tradeoff. Maybe binary search is not the best method to decide α^* with optimal prediction-attack accuracy ratio, but it is an adaptive algorithm and its computation cost is small.

G. Impact of Number of Attackers

In FL scenarios, some malicious clients may collude to jointly infer whether a given target sample belongs to the training data of the remaining benign clients. The more clients that collude, the more training and testing data the attacker has access to, which aids in conducting more powerful MIAs. So in this section, we vary the number of colluding FL clients, ranging from 1 to 9, to assess the impact of FL client attackers on the performance of our defense mechanism. Here, we evaluate the effectiveness of our defense method against G-Attack on the MNIST and Purchase20 datasets, and the results are shown in Table IV.

From our results we can see that the performance of *P^2-Protection* gradually declines on both the MNIST and Purchase20 datasets as the number of colluding FL clients increases. Specifically, for MNIST dataset, our defense performance significantly decreases as the number of colluding FL clients increases from 1 to 6. However, when the number of colluding clients continues to grow, the trend of performance decline becomes less pronounced. Purchase20 dataset exhibits a similar phenomenon, but it is noteworthy that the decline in defense performance only begins to stabilize when the number of malicious clients exceeds 7. The main reason for this phenomenon is that as the number of colluding attackers increases, the amount of data they can use to train the attack model also increases. This increase means that attackers can obtain more information about the relationship between protected gradients and membership properties, which significantly enhances the performance of the inference attack model, leading to a decline in our defense performance. When the data volume reaches a certain level, the data from newly added malicious clients mostly overlaps with existing data, thus the impact on our defense performance gradually decreases.

TABLE IV: The impact of the number of attackers on the defense performance of P^2-*Protection* under the G-Attack.

Number of Malicious Clients	Number of Benign Clients	Attack Acc.	
		MNIST	Purchase20
1	9	0.540	0.531
2	8	0.557	0.553
3	7	0.561	0.576
4	6	0.571	0.591
5	5	0.576	0.614
6	4	0.592	0.619
7	3	0.598	0.625
8	2	0.601	0.627
9	1	0.604	0.635

VI. Related Work

A. Membership Inference Attacks

Centralized Learning. Shokri et al. [25] conduct the first MIA called Shadow Attack targeting centralized ML models. They build a shadow model to imitate the behavior of the target model and use its prediction of member and non-member data as the input features of the attack model. However, it needs a public dataset with the same distribution as the private dataset on which the model is trained. As a follow-up, Salem et al. [18] relax the assumption in Shadow Attack, and study the independence of dataset and shadow model in MIAs. Yeom et al. [33] explore the relationship between overfitting and privacy leakage and find out that overfitting is not a necessary factor for MIAs.

Except for prediction, other forms of model parameters like gradients, loss value, loss trajectory, and prediction label are also shown to be relevant to membership information. For instance, Nasr et al. [29] compare different forms of leakage and find that the gradients of latter layers are more susceptible to privacy attacks, since high dimensional gradients leak more membership information than lower dimensional model outputs [34]. Concerning the black-box models, Liu et al. [35] use a local linear model to approximate the gradients of the target model and then perform MIAs. More recent works study on the relationship between model prediction boundary and membership property, and propose boundary-based attacks [36]–[38]. These boundary-based attacks are more realistic because they only use prediction labels to verify membership information. Liu et al. [39] exploit the loss trajectory of the target sample from the whole training process concerning the target model, and mine the membership information hidden behind the trajectory to differentiate members from non-members. Ko et al. [40] then leverage the embedding similarities between the image and text features, and analyze the membership privacy of training data in large-scale multi-modal models.

Federated Learning. Melis et al. [41] implement MIAs in FL setting and find that some unintended information as well as membership can be revealed from the model updates. An active attacker can thus achieve a more powerful attack by uploading elaborated updates. Nasr et al. [29] investigate the vulnerability of stochastic gradient descent algorithm and deign a general MIA. They explain that the gradient of loss on a data record reflects the membership property, which inspires us to use gradient ascend to protect the privacy. Zari et al. [42] propose a computation and time saving passive MIA and design an attack model architecture that captures the FL training dynamics. Considering the difficulty to acquire a training set with the same distribution as the target model, Zhang et al. [43] use generative adversarial networks to generate the training data of the target client, making the attack possible under the iid condition. Recently, Li et al. [30] discover that for a trained FL model, the gradients of different training samples tend to be almost orthogonal. They then leverage this property by comparing the gradient of a sample with the gradients of the global model to conduct MIAs.

What is more, during the FL training process, both the prediction score and the adversarial robustness series of a target sample with respect to the target models can also breach the membership privacy of the FL client, so Gu et al. [44] and Liu et al. [12] respectively construct an inference model to extract membership features from the series of confidence and robustness on both training and testing data to execute MIAs. Pichler and colleagues [13] introduce an FL server-side MIA. They create unique model parameters for a target sample and utilize clients' updates to discern the membership status of this sample.

B. Defenses Against MIAs

Centralized Learning. One basic idea for the defense against MIAs is to enhance the generalization of the target model. Shokri et al. [25] propose to add an L2 regularization term during the calculation of loss function. Dropout [45] is first proposed to reducing overfitting in ML model but is soon proven to be a defense for privacy leakage [46].

One more effective way of enhancing privacy is DP [47] which provides theoretical privacy guarantee with additional random noise. Abadi et al. [19] provide a simple way called DP-SGD in centralized scenario, to implement DP during model training phase. But it is found to introduce convergence problem and accuracy loss. To lower such side effects, Rahimian et al. [48] point out that DP-SGD not only decelerates the training speed, but also leads to a unstable privacy budget with respect to the model complexity. Instead of processing gradients during the training phase, they choose to add noise to the prediction. Though it does not affect the training process of the target model, it limits the application of white-box models and cannot resist label-only attacks. In addition to directly adding noise to model predictions, Chen et al. [22] introduce HAMP, a defense mechanism that employs high-entropy soft labels during training and modifies prediction outputs at testing time. By enforcing less confident predictions across both training and testing samples, HAMP mitigates the risk of membership privacy leakage without compromising model accuracy.

Federated Learning. Since the privacy information about the training batch can be inferred from the uploading parameters, many defenses focus on adding perturbation on the uploading parameters [49]. For example, Shokri et al. [50] apply DP in FL setting, where each participant adds carefully selected

Laplacian noise to the original updates before sending it to the server. Unfortunately, except for notable accuracy decay, their privacy budget is given per-parameter, resulting in overwhelming calculation for complex neural network models that contain a mass of parameters. Yang et al. [51] design a perturbation method that causes little impact on model utility. However, it only defeats inference attacks launched by clients and the server could recover the training data from the aggregated gradients. To protect the client-level privacy during the FL training, Geyer et al. [31] try to conceal the contribution of clients with a minor cost of model utility, but also requiring an honest server, which means the server cannot tamper with the aggregation results.

Except for perturbation-based defenses, some novel privacy-preserving frameworks are proposed. Zhao et al. [52] present PrivateDL, which allows clients to transfer knowledge from sensitive training data with public dataset. Similarly, Fan et al. [53] propose secret polarization network which imposes perturbations on uploading parameters while maintaining the original performance. However, inducing a new collaborative training process is not an easy way and may involve other vulnerabilities. In [54], the client trains an encryption network privately to modify the raw training data which results in distorting updates. Boutet et al. [55] rectify normal FL training progress by adding a trustworthy third party, but it is not that practical and also computationally intensive. Gu et al. [21] introduce FedPass, a method that incorporates an adaptive obfuscation layer prior to the input of data features into the FL model, effectively concealing the original features and thus safeguarding the sensitivity of the training data.

Among those defenses, perturbation-based methods may hurt utility since random perturbation affects the convergence of optimization; other privacy-preserving frameworks need to modify the original FL training process or even induce new risks. Compared with existing defenses, our method is a post-training defense mechanism, which means it can be implemented (as a patch) after the completion of FL training and does not modify the original FL process. Additionally, although our method is perturbation-based, it differs from other perturbation-based methods, such as DP-SGD, in that it does not directly add noise to the values of gradients or predictions; instead, it perturbs the gradients in terms of the direction perspective and also considers restrictions on the accuracy decay of the FL model. Lastly, our method perturbs both the FL model's predictions and gradients simultaneously, which is an advantage over some existing methods that typically target either predictions or gradients.

VII. Conclusion

In this paper, we have presented P^2-*Protection*, an MIA defense which can be executed after the training process of an FL model, by poisoning the prediction and gradient of the given model's training data. Compared with existing defenses, P^2-*Protection* requires neither the modification of FL training process nor the access to the target FL model's internal parameters and training data. By introducing an extra round of training with the poisoned gradient, P^2-*Protection* can defend against MIAs with respect to the prediction and gradient simultaneously. We compare P^2-*Protection* with two typical defenses against three MIAs on five realistic datasets, and the experiment results demonstrate that our defense can achieve an improved trade-off between model utility and data privacy. We envision our work as a solid step in FL towards defending privacy leakage of client data and provides an alternative approach on defense mechanisms against MIAs.

References

[1] S. Dong, P. Wang, and K. Abbas, "A survey on deep learning and its applications," *Computer Science Review*, vol. 40, p. 100379, 2021.

[2] Y. Zheng, S. Lai, Y. Liu, X. Yuan, X. Yi, and C. Wang, "Aggregation service for federated learning: An efficient, secure, and more resilient realization," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 2, pp. 988–1001, 2022.

[3] J. Zhao, H. Zhu, F. Wang, R. Lu, Z. Liu, and H. Li, "PVD-FL: A privacy-preserving and verifiable decentralized federated learning framework," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 2059–2073, 2022.

[4] J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *CoRR, arXiv:1610.02527*, 2016.

[5] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *CoRR, arXiv:1610.05492*, 2016.

[6] B. Liu, L. Wang, M. Liu, and C.-Z. Xu, "Federated imitation learning: A novel framework for cloud robotic systems with heterogeneous sensor data," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3509–3516, 2020.

[7] G. A. Kaissis, M. R. Makowski, D. Rückert, and R. F. Braren, "Secure, privacy-preserving and federated machine learning in medical imaging," *Nature Machine Intelligence*, vol. 2, no. 6, pp. 305–311, 2020.

[8] D. C. Nguyen, Q.-V. Pham, P. N. Pathirana, M. Ding, A. Seneviratne, Z. Lin, O. Dobre, and W.-J. Hwang, "Federated learning for smart healthcare: A survey," *ACM Computing Surveys*, vol. 55, no. 3, pp. 1–37, 2022.

[9] H. Hu, X. Zhang, Z. Salcic, L. Sun, K.-K. R. Choo, and G. Dobbie, "Source inference attacks: Beyond membership inference attacks in federated learning," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–18, 2023.

[10] Z. Wang, Y. Huang, M. Song, L. Wu, F. Xue, and K. Ren, "Poisoning-assisted property inference attack against federated learning," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 4, pp. 3328–3340, 2022.

[11] B. Z. H. Zhao, A. Agrawal, C. Coburn, H. J. Asghar, R. Bhaskar, M. A. Kaafar, D. Webb, and P. Dickinson, "On the (in) feasibility of attribute inference attacks on machine learning models," in *Proceedings of IEEE S&P*, 2021, pp. 232–251.

[12] G. Liu, Z. Tian, J. Chen, C. Wang, and J. Liu, "TEAR: Exploring temporal evolution of adversarial robustness for membership inference attacks against federated learning," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 4996–5010, 2023.

[13] G. Pichler, M. Romanelli, L. R. Vega, and P. Piantanida, "Perfectly accurate membership inference by a dishonest central server in federated learning," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–18, 2023.

[14] G. Liu, T. Xu, X. Ma, and C. Wang, "Your model trains on my data? protecting intellectual property of training data via membership fingerprint authentication," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 1024–1037, 2022.

[15] P. Maini, M. Yaghini, and N. Papernot, "Dataset inference: Ownership resolution in machine learning," in *Proceedings of ICLR*, 2021.

[16] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proceedings of IEEE S&P*, 2017, pp. 3–18.

[17] M. Nasr, R. Shokri, and A. Houmansadr, "Machine learning with membership privacy using adversarial regularization," in *Proceedings of ACM CCS*, 2018, pp. 634–646.

[18] A. Salem, Y. Zhang, M. Humbert, M. Fritz, and M. Backes, "Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models," in *Proceedings of NDSS*, 2019.

[19] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of ACM CCS*, 2016, pp. 308–318.
[20] M. Naseri, J. Hayes, and E. De Cristofaro, "Local and central differential privacy for robustness and privacy in federated learning," in *Proceedings of NDSS*, 2022.
[21] H. Gu, J. Luo, Y. Kang, L. Fan, and Q. Yang, "Fedpass: privacy-preserving vertical federated deep learning with adaptive obfuscation," in *Proceedings of IJCAI*, 2023, pp. 3759–3767.
[22] Z. Chen and K. Pattabiraman, "Overconfidence is a dangerous thing: Mitigating membership inference attacks by enforcing less confident prediction," in *Proceedings of NDSS*, 2024.
[23] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 2, pp. 1–19, 2019.
[24] P. Kairouz and H. B. McMahan et al., *Advances and Open Problems in Federated Learning*, ser. Foundations and Trends in Machine Learning. Now Publishers, 2021.
[25] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proceedings of IEEE S&P*, 2017, pp. 3–18.
[26] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
[27] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," in *Proceedings of ICLR*, 2019.
[28] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.
[29] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in *Proceedings of IEEE S&P*, 2019, pp. 739–753.
[30] J. Li, N. Li, and B. Ribeiro, "Effective passive membership inference attacks in federated learning against overparameterized models," in *Proceedings of ICLR*, 2023.
[31] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," *CoRR, arXiv: 1712.07557*, 2017.
[32] J. Sun, A. Li, B. Wang, H. Yang, H. Li, and Y. Chen, "Soteria: Provable defense against privacy leakage in federated learning from representation perspective," in *Proceedings of the IEEE/CVF CVPR*, 2021, pp. 9311–9319.
[33] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy risk in machine learning: Analyzing the connection to overfitting," in *Proceedings of IEEE CSF*, 2018, pp. 268–282.
[34] K. Leino and M. Fredrikson, "Stolen memories: leveraging model memorization for calibrated white-box membership inference," in *Proceedings of USENIX Security Symposium*, 2020, pp. 1605–1622.
[35] G. Liu, T. Xu, R. Zhang, Z. Wang, C. Wang, and L. Liu, "GradientLeaks: Enabling black-box membership inference attacks against machine learning models," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 427–440, 2024.
[36] Z. Li and Y. Zhang, "Membership leakage in label-only exposures," in *Proceedings of ACM CCS*, 2021, pp. 880–895.
[37] G. Zhang, B. Liu, T. Zhu, M. Ding, and W. Zhou, "Label-only membership inference attacks and defenses in semantic segmentation models," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 2, pp. 1435–1449, 2023.
[38] G. Del Grosso, H. Jalalzai, G. Pichler, C. Palamidessi, and P. Piantanida, "Leveraging adversarial examples to quantify membership information leakage," in *Proceedings of IEEE/CVF CVPR*, 2022, pp. 10 399–10 409.
[39] Y. Liu, Z. Zhao, M. Backes, and Y. Zhang, "Membership inference attacks by exploiting loss trajectory," in *Proceedings of ACM CCS*, 2022, pp. 2085–2098.
[40] M. Ko, M. Jin, C. Wang, and R. Jia, "Practical membership inference attacks against large-scale multi-modal models: A pilot study," in *Proceedings of IEEE/CVF ICCV*, 2023, pp. 4871–4881.
[41] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *Proceedings of IEEE S&P*, 2019, pp. 691–706.
[42] O. Zari, C. Xu, and G. Neglia, "Efficient passive membership inference attack in federated learning," *CoRR, arXiv: 2111.00430*, 2021.
[43] J. Zhang, J. Zhang, J. Chen, and S. Yu, "Gan enhanced membership inference: A passive local attack in federated learning," in *Proceedings of IEEE ICC*, 2020, pp. 1–6.
[44] Y. Gu, Y. Bai, and S. Xu, "Cs-mia: Membership inference attack based on prediction confidence series in federated learning," *Journal of Information Security and Applications*, vol. 67, p. 103201, 2022.
[45] E. Galinkin, "The influence of dropout on membership inference in differentially private models," *CoRR, arXiv: 2103.09008*, 2021.
[46] L. Song and P. Mittal, "Systematic evaluation of privacy risks of machine learning models," in *Proceedings of USENIX Security Symposium*, 2021, pp. 2615–2632.
[47] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Proceedings of TCC*, 2006, pp. 265–284.
[48] S. Rahimian, T. Orekondy, and M. Fritz, "Sampling attacks: Amplification of membership inference attacks by repeated queries," *CoRR, arXiv: 2009.00395*, 2020.
[49] N. Rodríguez-Barroso, D. Jiménez-López, M. V. Luzón, F. Herrera, and E. Martínez-Cámara, "Survey on federated learning threats: concepts, taxonomy on attacks and defences, experimental study and challenges," *Information Fusion*, vol. 90, pp. 148–173, 2023.
[50] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of ACM CCS*, 2015, pp. 1310–1321.
[51] X. Yang, Y. Feng, W. Fang, J. Shao, X. Tang, S.-T. Xia, and R. Lu, "An accuracy-lossless perturbation method for defending privacy attacks in federated learning," in *Proceedings of ACM WWW*, 2022, pp. 732–742.
[52] Q. Zhao, C. Zhao, S. Cui, S. Jing, and Z. Chen, "Privatedl: privacy-preserving collaborative deep learning against leakage from gradient sharing," *International Journal of Intelligent Systems*, vol. 35, no. 8, pp. 1262–1279, 2020.
[53] L. Fan, K. W. Ng, C. Ju, T. Zhang, C. Liu, C. S. Chan, and Q. Yang, "Rethinking privacy preserving deep learning: How to evaluate and thwart privacy attacks," in *Federated Learning: Privacy and Incentive*, 2020, pp. 32–50.
[54] H. Lee, J. Kim, S. Ahn, R. Hussain, S. Cho, and J. Son, "Digestive neural networks: A novel defense strategy against inference attacks in federated learning," *Computers & Security*, vol. 109, p. 102378, 2021.
[55] A. Boutet, T. Lebrun, J. Aalmoes, and A. Baud, "Mixnn: Protection of federated learning against inference attacks by mixing neural network layers," *CoRR, arXive: 2109.12550*, 2021.

Gaoyang Liu (S'20-M'21) received the B.S. and Ph.D. degrees from Huazhong University of Science and Technology, China, in 2015 and 2021, respectively. He is now a post-doctoral researcher at the School of Computing Science, Simon Fraser University, British Columbia, Canada. His research interests include trustworthy machine learning, mobile sensing and data privacy protection. He is a member of IEEE.

Tianlong Xu received the B.S. degrees from Wuhan University of Technology, Wuhan in 2020. He is currently working toward the Ph.D degree in School of Electronic Information and Communications, Huazhong University of Science and Technology, China. His recent research interests focus on security and privacy in deep learning and federated learning.

Yang Yang (M'17) received the BE and MS degrees from the Wuhan University of Technology, China, in 2009 and 2012, respectively, and the PhD degree from the Huazhong University of Science and Technology, China, in 2017. He is currently an associate professor with the School of Artificial Intelligence, Hubei University, China. His research interests include computer networks and privacy issuses in intelligent systems.

Ahmed M. Abdelmoniem (M'17-SM'25) received his Ph.D. in Computer Science and Engineering from Hong Kong University of Science and Technology, Hong Kong in 2017. He is an Associate Professor at the Queen Mary University of London, UK and leads the Scalable Adaptive Yet Efficient Distributed (SAYED) Systems Research Group. Formerly, he was a Research Scientist at KAUST, Saudi Arabia, and a Senior Researcher with Huawei's Future Networks Lab in Hong Kong. He is an investigator on several UK and international grants totaling nearly USD 1.5mil in funding. His research interests lie in the intersection of distributed systems, machine learning, and computer networks. He is a member of ACM and USENIX.

Chen Wang (S'10-M'13-SM'19) received the B.S. and Ph.D. degrees from the Department of Automation, Wuhan University, China, in 2008 and 2013, respectively. From 2013 to 2017, he was a postdoctoral research fellow in the Networked and Communication Systems Research Lab, Huazhong University of Science and Technology, China. Thereafter, he joined the faculty of Huazhong University of Science and Technology where he is currently an associate professor. His research interests are in the broad areas of wireless networking, Internet of Things, and mobile computing, with a recent focus on privacy issues in wireless and mobile systems. He is a senior member of IEEE and ACM.

Jiangchuan Liu (S'01-M'03-SM'08-F'17) received B.Eng. (*Cum Laude*) from Tsinghua University, Beijing, China, in 1999, and Ph.D. from The Hong Kong University of Science and Technology in 2003. He is currently a Full Professor (with University Professorship) in the School of Computing Science at Simon Fraser University, British Columbia, Canada. He is a Fellow of the Canadian Academy of Engineering, an IEEE Fellow and an NSERC E.W.R. Steacie Memorial Fellow. Prof. Liu has been a Steering Committee Member of IEEE Transactions on Mobile Computing, and Associate Editor of IEEE/ACM Transactions on Networking, IEEE Transactions on Network Science and Engineering, IEEE Transactions on Big Data, IEEE Transactions on Multimedia, and IEEE Communications Tutorial and Surveys. He is a co-recipient of the Test of Time Paper Award of IEEE INFOCOM (2015), ACM TOMCCAP Nicolas D. Georganas Best Paper Award (2013), the ACM Multimedia Best Paper Award (2012), the IEEE Globecom Best Paper Award (2011), and the IEEE Communications Society Best Paper Award on Multimedia Communications (2009).