TEAR: Exploring Temporal Evolution of Adversarial Robustness for Membership Inference Attacks against Federated Learning

Gaoyang Liu, *Member, IEEE*, Zehao Tian, Jian Chen, *Student Member, IEEE*, Chen Wang, *Senior Member, IEEE*, and Jiangchuan Liu, *Fellow, IEEE*

***Abstract*—Federated learning (FL) is a privacy-preserving machine learning paradigm that enables multiple clients to train a unified model without disclosing their private data. However, susceptibility to membership inference attacks (MIAs) arises due to the natural inclination of FL models to overfit on the training data during the training process, thereby enabling MIAs to exploit the subtle differences in the FL model's parameters, activations, or predictions between the training and testing data to infer membership information. It is worth noting that most if not all existing MIAs against FL require access to the model's internal information or modification of the training process, yielding them unlikely to be performed in practice. In this paper, we present with TEAR the first evidence that it is possible for an honest-but-curious federated client to perform MIA against an FL system, by exploring the Temporal Evolution of the Adversarial Robustness between the training and non-training data. We design a novel adversarial example generation method to quantify the target sample's adversarial robustness, which can be utilized to obtain the membership features to train the inference model in a supervised manner. Extensive experiment results on five realistic datasets demonstrate that TEAR can achieve a strong inference performance compared with two existing MIAs, and is able to escape from the protection of two representative defenses.**

***Index Terms*—Federated learning, membership inference attack, adversarial robustness, temporal evolution.**

I. Introduction

Federated learning (FL) is a privacy-aware machine learning paradigm that enables multiple clients to create a unified model without revealing their private training data [1], [2]. The basic process involves training local models on each client's data and exchanging model updates (such as the parameters and gradients of a learning model) between federated clients to generate a unified model shared by all clients. FL embodies the principles of focused collection and data minimization [3], and can reduce the storage complexity and computational costs associated with collecting distributed data. Without sharing data, FL can thus mitigate data privacy and user confidentiality issues commonly encountered with traditional centralized machine learning [4], [5]. Due to its privacy-preserving capability, FL has been widely used in many realistic applications, such as mobile keyboard prediction [6], smart city [7], and recommendation systems [8].

However, recent works have demonstrated that FL models are vulnerable to different attacks, including source inference attacks [9], model inversion attacks [10], attribute inference attacks [11], and property inference attacks [12], which leak sensitive information present in the training dataset. In this paper, we concentrate on the so-called membership inference attacks (MIAs) against the FL model, where the adversary aims to infer whether a given sample (i.e. the target sample) was used as part of the training data of the given model. Successful MIAs can lead to severe privacy risk to federated clients. For instance, if an FL model is trained on the data distributed across multiple medical institutions, then knowing the target sample participating the model training process may reveal the disease history as well as the treatment history of the corresponding victim.

Despite extensive research efforts on MIAs against FL models, most if not all existing studies make use of the inherent overfitting property of FL models. Naturally, an FL model would have more confidence in its prediction of the training data compared with the non-training data. This prediction confidence is reflected in the FL model's prediction probabilities [13]–[15], intermediate computation results [16], and parameter gradients [17]–[19]. However, the success of these MIAs relies on the access to the FL model's interior, through which the adversary can obtain its architecture and parameters that is required to get the model's prediction probabilities and the intermediate computations. In addition, a part of MIAs needs the ability to eavesdrop and manipulate the client updates or gradients of FL model parameters to perform MIA [17], [18].

Nevertheless, it is difficult for existing MIAs to obtain the essential information to perform the inference in FL systems. In practice, the model internal parameters and the client updates are secured through Homomorphic Encryption (HE) [20] or Differential Privacy (DP) [21] during both the training and inference stages [3], [22]. In such a case, it is almost impossible for an adversarial client to obtain the internal information of the FL model, not to mention decrypting and

This work was supported in part by the National Natural Science Foundation of China under Grants 62272183, 62171189, 62002104 and 62071192; by the Key Research and Development Program of Hubei Province under Grant 2021BAA026; and by the special fund for Wuhan Yellow Crane Talents (Excellent Young Scholar). *(Corresponding author: Chen Wang.)*

G. Liu, Z. Tian, J. Chen and C. Wang are with the Hubei Key Laboratory of Smart Internet Technology, School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, China. G. Liu is also with the School of Computing Science, Simon Fraser University, British Columbia, Canada. Email: {liugaoyang, zhtian, jianchen, chenwang}@hust.edu.cn.

J. Liu is with School of Computing Science, Simon Fraser University, British Columbia, Canada. Email: jcliu@cs.sfu.ca.

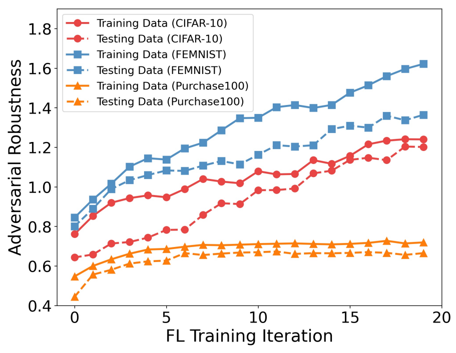


Fig. 1. The key observation on the temporal evolution of adversarial robustness concerning the training and testing data of the FL model. The settings are the same as described in Section V. It can be observed that the adversarial robustness for training and testing samples demonstrates dissimilar temporal evolution trends between the training and testing samples as the FL training process proceeds. This difference implies that the temporal evolution of adversarial robustness can serve as a viable indicator for distinguishing training and testing samples.

manipulating the parameter updates and gradients. Against this background, we consider a restricted scenario in which an honest-but-curious client can only access to the local training information of the FL model. To enhance the usability of MIA in practice, we further assume the adversarial client can merely obtain the label prediction interface of the current FL model at the beginning of each local training round. Then a key question arises naturally: is it possible to launch MIA against FL models in restricted scenario where only the predicted label of the queried sample is available?

In this paper, we present a novel MIA by exploring the Temporal Evolution of the Adversarial Robustness (TEAR) between the training and non-training data. With TEAR, we for the first time demonstrate that it is possible for an honest-but-curious federated client to perform MIA against an FL system *at the training stage*. Our key observation is that the convergence trend (called the temporal evolution) of an FL model's adversarial robustness, which refers to the tolerance of a perturbation causing a misclassified result of the target model, is different between its training and testing data (c.f. Fig. 1). Specifically, the decision boundary of the FL model is gradually fitted to the training data during the training process, whereas this is not the case for the testing data. As a result, the distance between the decision boundary and the training data gradually increases, which makes them less susceptible to adversarial perturbations [23], [24], thereby yielding a higher adversarial robustness. By continuously collecting the adversarial robustness of target samples with respect to the FL model for different training rounds during the training process, the temporal evolution of adversarial robustness can be obtained and further utilized as features to train the membership inference model in a supervised manner.

We have to emphasize that although the idea sounds simple, the realization of TEAR confronts one major challenge: how to quantify the adversarial robustness of the target sample to strengthen the membership information hidden in it? Some recent MIA works [25]–[27] leverage adversarial attacks [28]–[30] to quantify the target sample's adversarial robustness. However, existing adversarial attacks usually craft the target sample along the direction of increasing the corresponding prediction loss, which cannot directly reflect the relationship between the target model's decision boundary and the target sample. Furthermore, a recent work [27] has demonstrated that the selection of adversarial attacks largely determines the performance of the inference attacks. Therefore, with merely the label prediction interface of the target model, we need to design a novel adversarial example generation method targeted at MIAs. Inspired by the work [31] which finds that deep neural network (DNN) models are biased toward low-frequency functions and flatten decision boundaries, we leverage the direction consistency of the normal vector of the decision boundary and the vector from the target sample to its adversarial version to craft the target sample. Then we regard the distance between the target sample and the generated adversarial example as the quantitative proxy for the corresponding adversarial robustness.

We conduct extensive experiments on five datasets, and the results demonstrate that TEAR can achieve strong performance, and in most cases can even outperform the white-box MIAs [17], [25]. We also evaluate TEAR against two defense mechanisms: DP-SGD [21] and MemGuard [32], and the results show that TEAR can escape the protection of both defenses. The results reveal the membership information hidden behind the temporal evolution of adversarial robustness of FL models.

We summarize our major contributions as follows:

- We present TEAR, a novel client-side MIA against FL models with only black-box access. TEAR does not rely on the internals of the FL model (i.e. model parameters, gradients, prediction probabilities, intermediate computations), nor manipulating the training process.
- We observe the difference in the convergence tendency of adversarial robustness between the training and non-training data with respect to a given FL model, and show for the first time how the temporal evolution of adversarial robustness can be leveraged as an important membership feature to facilitate MIA against FL models.
- We design a novel adversarial example generation method tailed for MIAs to measure the distance of the target sample to the decision boundary of the target FL model, which serves as a proxy for quantifying the adversarial robustness of the given sample.
- We evaluate the performance of TEAR on five realistic datasets, and experiment results demonstrate that TEAR can achieve a strong inference performance compared with two existing MIAs. Further experiments show that TEAR can threaten the membership privacy of FL models even with the protection of DP-SGD [21] and MemGuard [32]. The code of TEAR has been released for reproducibility purposes[1].

The remainder of this paper is organized as follows. Section II describes some preliminary knowledge on FL and adversarial examples. Section III defines the threat model, followed by our design details in Section IV. Section V

[1]https://www.dropbox.com/s/u4r1s6kxexdvrcx/TEAR-Code.zip?dl=0

presents the performance evaluation and Section VI provides some discussions on defenses. Section VII reviews some related works. Finally Section VIII concludes this paper.

II. Preliminary

A. Federated Learning

FL is a distributed ML training paradigm that allows for data training by coordinating multiple clients without requiring access to the clients' local data [33], [34]. In a typical FL system, there are K federated clients $C_1, C_2, \cdots, C_K$, and one central server S which organizes the FL training process and gets a converged global FL model $\mathcal{M}$ iteratively (c.f. Fig. 2).

Specifically, at the t_{th} training iteration ($t \in T$, where T is the number of FL training iterations), the training process of the FL model $\mathcal{M}$ consists of the following four steps:

Step 1. The central server S disseminates the current global FL model $\mathcal{M}^t$ to the participating federated clients C_k ($k \in 1, 2, \cdots, K$).

Step 2. Each client C_k locally trains and refines the received model $\mathcal{M}^t$ with its own data D_k in parallel. After the local training is completed, the client C_k sends its updated model parameters U_k^t to the central server.

Step 3. The central server S collects the updated parameters $U^t = [U_1^t, U_2^t, \cdots, U_K^t]$ from all participating clients.

Step 4. The central server S updates the global model $\mathcal{M}^t$ on the basis of the aggregation of the collected parameter updates U^t. Then the newly updated model $\mathcal{M}^{t+1}$ will be disseminated to federated clients in the next training iteration.

The central server and clients of FL will execute the above training process iteratively, until the termination criterion has been satisfied (e.g. a maximum number of iterations is reached or the model accuracy is greater than a threshold). After that, the central server obtains the converged FL model $\mathcal{M}$ and disseminates it to the clients.

In practice, FL frameworks such as FATE[2] and Pysyft[3] potentially use multiple strategies to protect the confidentiality of the exchanged information as well as the integrity of FL model training process. For instance, FATE integrates TEE technique into its framework to prevent the adversary from compromising the training process, and Pysyft makes use of DP, SMPC and HE to protect the model updates of federated clients and the parameters of the disseminated model.

Note that FL can be roughly separated into the horizontal and vertical ones. The former is applicable to the scenarios in which the data owned by federated clients share the same feature space, while the latter often handles the cases in which client datasets come from different feature spaces. Considering the relatively simplicity of the horizontal FL and the resulting wide application in practice, we focus on analyzing the leakage risk of membership privacy for the horizontal FL.

B. Membership Inference Attacks

The goal of our paper is to investigate the leakage risks of membership privacy with respect to FL models. For ease

[2]https://fate.fedai.org/
[3]https://github.com/OpenMined/PySyft

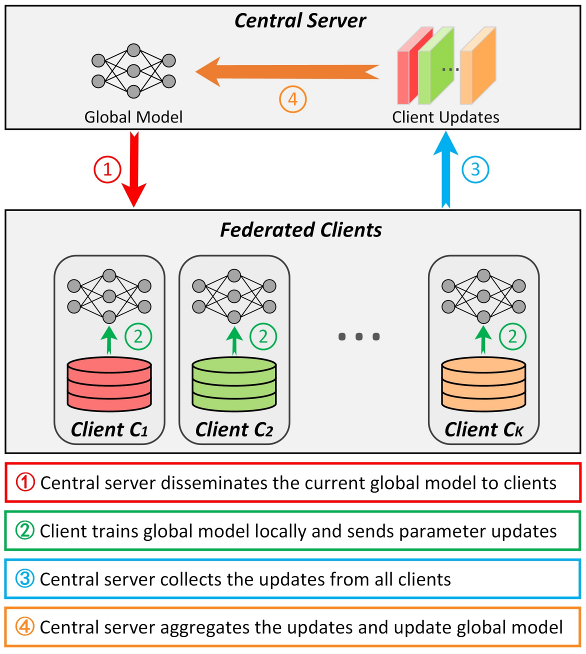

Central Server
Global Model
④
Client Updates
①
③
Federated Clients
②
Client C1
Client C2
Client CK
① Central server disseminates the current global model to clients
② Client trains global model locally and sends parameter updates
③ Central server collects the updates from all clients
④ Central server aggregates the updates and update global model

Fig. 2. The framework of FL.

of following our paper, we introduce the background of MIA against machine learning (ML) models in this section. Generally, MIAs exploit the inherent property that ML models often behave differently on the data that they were trained on versus the data that they meet for the first time. By leveraging the prediction behavior of an ML model $\mathcal{M}$ (called the *target model*), the inference adversary attempts to determine the membership property of a target sample $\mathbf{x}$, i.e., whether this sample was used to train the target model or not. Successful MIAs can breach the membership privacy of an ML model's training data, and pose new threats to training data providers.

More formally, the purpose of MIAs can be expressed as:

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

where $\mathcal{A}$ represents the attack model of MIAs, and **In** (resp. **Out**) means that $\mathbf{x}$ is a member (resp. non-member) of $\mathcal{M}$'s training data. Here, Ω is the external information about the target model and its training data that $\mathcal{A}$ can obtain.

The attack model $\mathcal{A}$ is essentially a binary classification model, and can be constructed in different ways depending on the external information Ω. According to previous works, Ω can be the target model's internals (e.g., the model structure [35], [36], parameters [17], gradients [17], and the training loss [37]), or its training data (e.g., data samples [35], [38], [39], or data distribution [40]).

Recently, several MIAs have been proposed in FL scenarios, which require the external information Ω of the internals of FL models [13]–[16], [18], [19] and even the manipulation of the FL training process [17], [18]. Considering these information is hard to obtain in FL systems, we thus aim to design an MIA in a more restricted scenario, in which we only have the access to the predicted labels of FL models.

III. Threat Model

We consider an honest-but-curious federated client as an adversary, who will not deviate from the normal FL training process but will attempt to infer the membership information of other clients' data from legitimately received FL models. The details of our threat model are described as follows.

Target model. We focus on assessing the membership privacy of a target FL model, regardless of what model structure or type is. When the central server of FL disseminates the current global model $\mathcal{M}^t$ to federated clients, the adversarial client can access to its local training process and obtain the prediction interface of the received model $\mathcal{M}^t$. Then the adversary can query $\mathcal{M}^t$ locally with data samples and get the corresponding predicted labels. We formalize the received global model as: $\mathcal{M}: \mathbf{x} \to y$, where $\mathbf{x}$ is the queried data, y is the predicted label. Different from the traditional MIAs which attacks against one trained ML model, our adversarial client performs MIA during the iterative training process of FL. In other words, the target model of our attack is the combination of a series of FL global models $\mathcal{M}^1, \mathcal{M}^2, \cdots, \mathcal{M}^T$.

Adversary's knowledge. The objective of our paper is to reveal the membership privacy risks in FL systems, where the local training process occurs within a trusted execution environment and model parameters are encrypted in practice [3]. Since the FL model's prediction probabilities are usually noised for the security purpose [35], [41], [42], we thus consider a restricted scenario in which the adversarial client *only* requires its local data and the prediction label of the target model which would remain unaffected by these protections. In such a case, the adversary *cannot* obtain any information about the following aspects:

- *Target model structure:* including the type and the structure of the target model.
- *Target model internals:* including the model parameters, intermediate computations such as activation values and gradients, and prediction probability vectors of the model's output layer.
- *Global training process of FL:* including the global training process and the hyper-parameter settings (such as client weight and global learning rate) controlled by the FL central server.
- *Samples or distribution of other clients' training data:* including any samples of the other benign clients, or the distribution of the training data for each individual client or whole clients.

Adversary's capability. In our threat model, we assume that the inference adversary is an honest-but-curious federated client, who can access but cannot manipulate its local training process of FL. Therefore, at each training iteration, the adversary can observe the current global model $\mathcal{M}^t (t \in [1, 2, \cdots, T])$. In our paper, we consider a restricted label-only scenario, in which the adversary can query the global model $\mathcal{M}^t$ with a sample $\mathbf{x}$ and obtain the predicted label y:

$$\mathcal{M}^t(\mathbf{x}) = y \tag{2}$$

During the FL training process, the current global FL model is transmitted to every FL client at the start of each epoch.

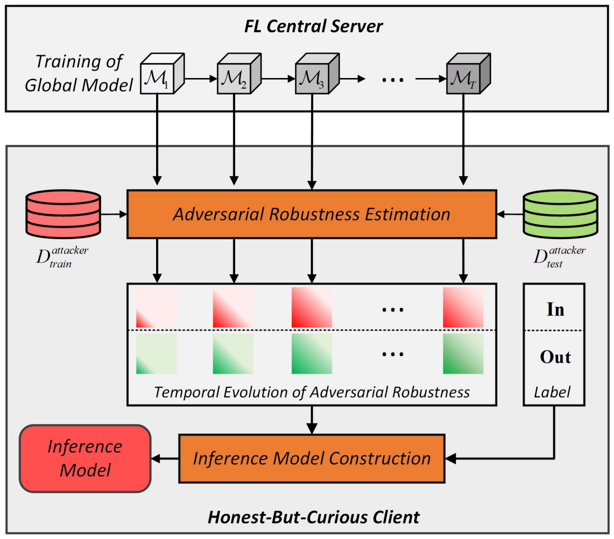


Fig. 3. The framework of TEAR.

However, this process has a vulnerability that a malicious client can save these models on its local storage [?]. This action allows the adversary to obtain a series of *intermediate parameter snapshots* of the global model as it progresses through training. To clarify, we denote the corresponding prediction interfaces as $[\mathcal{M}^1, \mathcal{M}^2, \cdots, \mathcal{M}^T]$. Consequently, these snapshots enable us to perform MIAs on any target sample, whose membership property the adversary aims to infer, after the FL training process has completed.

Adversary's goal. Given a target sample $\mathbf{x}_t$ and the observed global model $[\mathcal{M}^1, \mathcal{M}^2, \cdots, \mathcal{M}^T]$, the adversary's goal is to infer whether $\mathbf{x}_t$ is in other clients' training sets or not:

$$\mathcal{A}_{fed}(\mathbf{x}_t | [\mathcal{M}^1, \mathcal{M}^2, \cdots, \mathcal{M}^T]) \to \mathbf{In/Out} \tag{3}$$

where $\mathcal{A}_{fed}$ represents the attack model of TEAR.

IV. Design of TEAR

In order to assess the membership privacy of an FL model, we design an MIA dubbed TEAR from the perspective of an honest-but-curious federated client. TEAR is involved in the training process of the FL model, and leverages the difference of the temporal evolution of adversarial robustness between the given FL model's training and non-training (i.e. testing) data to estimate the membership attribution. Overall, TEAR mainly includes the following two parts (c.f. Fig. 3):

(1) Adversarial Robustness Estimation. At every iteration of the FL training process, TEAR measures the adversary robustness of a given record $\mathbf{x}$ with respect to the current global FL model $\mathcal{M}$. An adversarial example generation method is tailed to estimate the adversarial robustness of $\mathbf{x}$ to $\mathcal{M}$, which follows the normal vector on the $\mathcal{M}$'s decision boundary closest to $\mathbf{x}$ to generate minimal perturbations that are sufficient to mislead the prediction of $\mathcal{M}$. By doing so, the adversarial version of $\mathbf{x}$ (denoted as $\tilde{\mathbf{x}}$) can thus be generated. Then we find the sample $\bar{\mathbf{x}}$ that lies on the decision boundary between $\mathbf{x}$ and $\tilde{\mathbf{x}}$, and the distance between $\mathbf{x}$ and $\bar{\mathbf{x}}$ is regarded

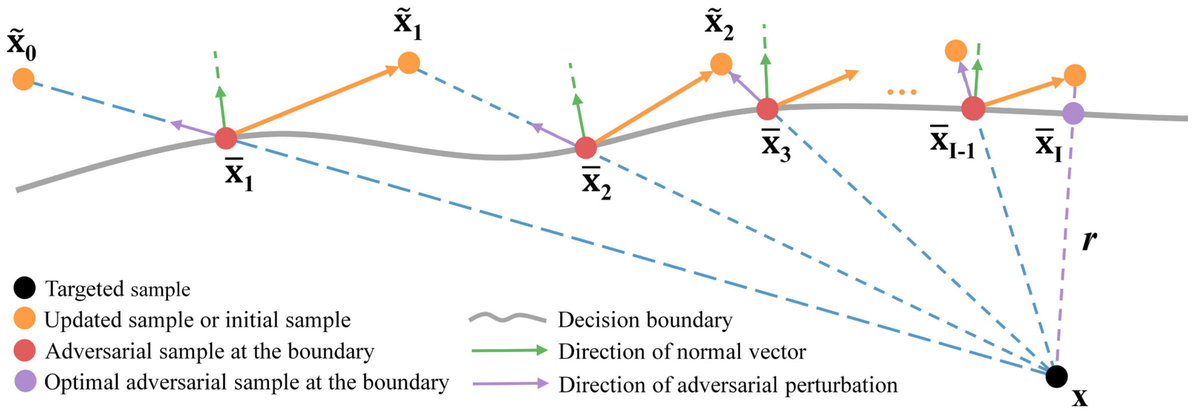


Fig. 4. Illustration of our adversarial example generation. (a) Initializing an adversarial point $\tilde{\mathbf{x}}_0$ at the first iteration. (b) Binary search to find the boundary point $\bar{\mathbf{x}}_i$ with $\tilde{\mathbf{x}}_{i-1}$ and $\mathbf{x}$. (c) Estimate normal vector at $\bar{\mathbf{x}}_i$ and compute $\tilde{\mathbf{x}}_{i-1} - \mathbf{x}$, and then get the loss of the objective function by Equation (5). (d) Optimize the objective function and update $\bar{\mathbf{x}}_i \rightarrow \tilde{\mathbf{x}}_i$. (e) The adversarial point $\tilde{\mathbf{x}}_i$ is used as the initial point for the next iteration.

as the quantification of adversarial robustness with respect to $\mathcal{M}$.

(2) Inference Model Construction. During the training process of the FL model $\mathcal{M}$, TEAR could observe a series of intermediate parameter snapshots of the global model $[\mathcal{M}^1, \mathcal{M}^2, \cdots, \mathcal{M}^T]$. By assessing the observed intermediate global models, the adversarial robustness evolutions of its local training data $D_{train}^{attacker}$ and testing data $D_{test}^{attacker}$ could be obtained. Then TEAR constructs the binary inference model in a supervised manner, which takes the adversarial robustness evolutions of the $D_{train}^{attacker}$ and $D_{test}^{attacker}$ as input and the membership properties of the corresponding data as supervised output. After TEAR trained the attack model, the inference model can be directly utilized to determine whether the target sample $\mathbf{x}$ is in the other clients' training data or not.

A. Adversarial Robustness Estimation

The key idea of the adversarial robustness estimation is to estimate the distance from the target record to the decision boundary of the target model, which would serve as the quantitative proxy of the adversarial robustness. In order to measure the decision boundary distance, we adopt adversarial attacks to generate samples that lie on the decision boundary of the target model. Then we treat the distance between the target sample and its adversarial version as the estimation of adversarial robustness.

Since the attacker can only get the predicted label of the received FL model, we consider a targeted adversarial example generation method against a black-box model, which aims to generate the adversarial example $\bar{\mathbf{x}}$ to change the predicted label of the target record $\mathbf{x}$ to a certain label y_k:

$$\begin{aligned} &\min_{v} \ \mathcal{D}(\mathbf{x}, \tilde{\mathbf{x}}) \\ &s.t. \ \mathcal{M}(\tilde{\mathbf{x}}) = y_k \end{aligned} \tag{4}$$

where $\mathcal{M}$ is the target model, y_k is the targeted label which could be any label but $\mathcal{M}(\mathbf{x})$, and $\mathcal{D}$ is a distance function for $\mathbf{x}$ and $\tilde{\mathbf{x}}$, such as cosine distance and $L2$ distance.

To generate $\tilde{\mathbf{x}}$, we are motivated by the geometry property of the adversarial perturbation $v\,(v = \tilde{\mathbf{x}} - \mathbf{x})$: the direction of the minimal v for $\mathbf{x}$ equals the normal vector direction of the decision boundary of the FL model $\mathcal{M}$ at the boundary sample $\bar{\mathbf{x}}_i$, which is located on the decision boundary of the target model and falls within the range of $\tilde{\mathbf{x}}_i$ and x. Therefore, we use cosine similarity to measure the direction deviation and design the objective function as follows:

$$\begin{aligned} &\arg\min_{\tilde{\mathbf{x}}} \ \mathcal{L}(\tilde{\mathbf{x}}) \\ &s.t. \ \mathcal{M}(\tilde{\mathbf{x}}) = y_k \\ &\mathcal{L}(\tilde{\mathbf{x}}) = -\frac{< \tilde{\mathbf{x}} - \mathbf{x}, \mathbf{N}|_{\bar{\mathbf{x}},\mathcal{M}} >}{\|\tilde{\mathbf{x}} - \mathbf{x}\|_2 \ \|\mathbf{N}|_{\bar{\mathbf{x}},\mathcal{M}}\|_2} \end{aligned} \tag{5}$$

where $<,>$ represents the inner product of vectors, $\|\cdot\|_2$ represents the length of the vector, $\mathbf{N}|_{\bar{\mathbf{x}},\mathcal{M}}$ is the normal vector of $\mathcal{M}$ at $\bar{\mathbf{x}}$, and $\tilde{\mathbf{x}} - \mathbf{x}$ is the adversarial perturbation.

To optimize the above objective function, we leverage the binary-search and gradient descent mechanisms. The optimization process of our adversarial example generation is described in Algorithm 1. The generation algorithm takes the target sample $\mathbf{x}$ and the target label y_k as inputs. At the i_{th} optimization iteration, we first take the adversarial example $\tilde{\mathbf{x}}_{i-1}$ generated in the previous iteration, and project $\tilde{\mathbf{x}}_{i-1}$ to the decision boundary by binary search and get $\bar{\mathbf{x}}_i$ (Line 2 of Algorithm 1). Then we estimate the normal vector $\mathbf{N}|_{\bar{\mathbf{x}}_i,\mathcal{M}}$ of the target model's decision boundary at $\bar{\mathbf{x}}$ by using the Monte Carlo estimation method (Line 3 of Algorithm 1). After that, we calculate the cosine similarity between $\tilde{\mathbf{x}}_i - \mathbf{x}$ and $\mathbf{N}|_{\bar{\mathbf{x}}_i,\mathcal{M}}$ (Line 4 of Algorithm 1). At the end of i_{th} optimization iteration, we update our adversarial example $\tilde{\mathbf{x}}_i$ by using the gradient descent method. (Line 5 of Algorithm 1). It worth noting that the initial adversarial example $\tilde{\mathbf{x}}_0$ could be any random sample that satisfies the condition $\mathcal{M}(\tilde{\mathbf{x}}_0) = y_k$.

1) Generating Adversarial Example: The process of our adversarial example generation is illustrated in Fig. 4, which mainly contains three steps:

(1) Binary search. Given the target sample $\mathbf{x}$ and its adversarial version $\tilde{\mathbf{x}}$ which meets the condition of $\mathcal{M}(\tilde{\mathbf{x}}) = y_t$, we

Algorithm 1 Adversarial Robustness Estimation

Input: Target sample $\mathbf{x}$, initial adversarial example $\tilde{\mathbf{x}}_0$, target model $\mathcal{M}$, max iteration number I, binary search threshold θ, sampling size B, sampling radii δ, learning rate η.
Output: Distance r from $\mathbf{x}$ to $\mathcal{M}$'s decision boundary.

```
    Generating Adversarial Example
1: for $i = 1$ to $I$ do
2:     $\bar{\mathbf{x}}_i \leftarrow$ BinarySearch($\mathbf{x}, \tilde{\mathbf{x}}_{i-1}, \mathcal{M}, \theta$)
3:     $\mathbf{N}|_{\bar{\mathbf{x}}_i,\mathcal{M}} \leftarrow$ NormalVectorEstimation($\bar{\mathbf{x}}_i, \mathcal{M}, B, \delta$)
4:     $\mathcal{L}(\tilde{\mathbf{x}}_{i-1}) \leftarrow < \tilde{\mathbf{x}}_{i-1} - \mathbf{x}, \mathbf{N}|_{\bar{\mathbf{x}}_i,\mathcal{M}} >$
5:     $\tilde{\mathbf{x}}_i \leftarrow$ExampleUpdate($\bar{\mathbf{x}}_i, \mathcal{L}(\tilde{\mathbf{x}}_{i-1}), \eta$)
6: end for
   Quantifying Adversarial Robustness
7: $\bar{\mathbf{x}}_I \leftarrow$ BinarySearch($\mathbf{x}, \tilde{\mathbf{x}}_I, \mathcal{M}, \theta$)
8: $r = ||\bar{\mathbf{x}}_I - \mathbf{x}||_2$
9: return $r$
```

Algorithm 2 Binary Search

Input: Target sample $\mathbf{x}$, adversarial example $\tilde{\mathbf{x}}$, target model $\mathcal{M}$, threshold θ.
Output: A sample $\bar{\mathbf{x}}$ located at $\mathcal{M}$'s decision boundary.

```
1: Set $\mathbf{x}_{cln} = \mathbf{x}$ and $\mathbf{x}_{adv} = \tilde{\mathbf{x}}$
2: while $||\mathbf{x}_{cln} - \mathbf{x}_{adv}|| > \theta$ do
3:     Set $\mathbf{x}_{mid} \leftarrow \frac{\mathbf{x}_{cln}+\mathbf{x}_{adv}}{2}$
4:     if $\mathcal{M}(\mathbf{x}_{mid}) = \mathcal{M}(\mathbf{x}_{adv})$ then
5:         Set $\mathbf{x}_{adv} \leftarrow \mathbf{x}_{mid}$
6:     else
7:         Set $\mathbf{x}_{cln} \leftarrow \mathbf{x}_{mid}$
8:     end if
9: end while
10: Set $\bar{\mathbf{x}} = \mathbf{x}_{adv}$
11: return $\bar{\mathbf{x}}$
```

make use of the binary search algorithm to find the example which locates at the decision boundary of the target model along the line between $\mathbf{x}$ and $\tilde{\mathbf{x}}$. The search process will stop when a certain constrain is satisfied. The complete binary search algorithm is described in Algorithm 2, where the binary search threshold θ is set to be a small constant.

(2) Normal vector estimation. Given an example $\bar{\mathbf{x}}$ located at the decision boundary of $\mathcal{M}$, we approximate the normal vector $\mathbf{N}|_{\bar{\mathbf{x}},\mathcal{M}}$ via the Monte Carlo estimation method:

$$\mathbf{N}|_{\bar{\mathbf{x}},\mathcal{M}} := \frac{1}{B}\sum_{b=1}^{B}\phi(\bar{\mathbf{x}}+\delta u_b)u_b, \tag{6}$$

$$\phi(\bar{\mathbf{x}}+\delta u_b) = \begin{cases} +1 \ , \ \mathcal{M}(\bar{\mathbf{x}}+\delta u_b) \neq y_k \\ -1 \ , \ \mathcal{M}(\bar{\mathbf{x}}+\delta u_b) = y_k. \end{cases} \tag{7}$$

where $\phi(\bar{\mathbf{x}}+\delta u_b)$ represents whether the sample $\bar{\mathbf{x}}+\delta u_b$ is adversarial or not, $\{u_b\}_{b=1}^{B}$ are i.i.d. drawn from the uniform distribution over the d-dimensional sphere, δ is a small positive constant and is proportional to d^{-1} [30], and B is the number of disturbed groups.

(3) Example update. We implement our data updates based on the objective function in Equation (5). We get the gradient $\nabla\mathcal{L}(\tilde{\mathbf{x}}_i)$ directly and use the gradient descent algorithm for the adversarial example updates as follows:

$$\tilde{\mathbf{x}}_i = \bar{\mathbf{x}}_{i-1} - \eta\nabla\mathcal{L}(\tilde{\mathbf{x}}_{i-1}) \tag{8}$$

where η represents the learning rate of the updates, and we can achieve high optimization efficiency by adjusting the learning rate and obtaining a suitable step size. However, in certain cases, the updated sample $\tilde{\mathbf{x}}_i$ may end up on the same side of the decision boundary as the target sample, i.e., $\mathcal{M}(\tilde{\mathbf{x}}_i) = \mathcal{M}(\mathbf{x})$. To ensure that $\tilde{\mathbf{x}}_i$ always belongs to the targeted label y_k, we employ the following equation to project $\tilde{\mathbf{x}}_i$ along the line connecting $\tilde{\mathbf{x}}_i$ and $\mathbf{x}$:

$$\tilde{\mathbf{x}}_i \leftarrow \tilde{\mathbf{x}}_i + \beta\frac{\tilde{\mathbf{x}}_i - \mathbf{x}}{||\tilde{\mathbf{x}}_i - \mathbf{x}||_2^2} \tag{9}$$

where $\beta \in [0, +\infty)$. We gradually increase β from 0 until the projected $\tilde{\mathbf{x}}_i$ is predicted to the target label by the target model. Furthermore, since the updated adversarial example typically does not lie on the decision boundary of the target model, we need to apply binary search again to project it onto the target model's decision boundary.

2) Quantifying Adversarial Robustness: After generating the adversarial example $\tilde{\mathbf{x}}_I$, TEAR next quantifies the adversarial robustness of $\mathbf{x}$ with respect to $\mathcal{M}$. To this end, we first find the example $\bar{\mathbf{x}}_I$ located at the decision boundary of $\mathcal{M}$ (Line 7 of Algorithm 1), and then compute the $L2$ distance r between $\bar{\mathbf{x}}_I$ and $\mathbf{x}$ (Line 8 of Algorithm 1). At the end, we regard the distance r as the quantitative measurement for the adversarial robustness of $\mathbf{x}$.

B. Inference Model Construction

Now that we have got the adversarial robustness of the target sample $\mathbf{x}$, the next step of TEAR is to infer the membership property of $\mathbf{x}$. Though the adversarial robustness of $\mathbf{x}$ alone does not provide a distinguishable membership feature [27], we find that the temporal evolution of the adversarial robustness between the training and testing data can be easily distinguished, given a series of the intermediate parameter snapshots of the FL global model $[\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_T]$, where T is the number of FL training iteration. As a consequence, TEAR can construct the membership inference model by exploiting the temporal evolution of the adversarial robustness.

Constructing Inference Model. Considering the adversarial client can access to its local training data $D_{train}^{attacker}$ and testing data $D_{test}^{attacker}$, we next construct a binary inference model in a supervised manner. The reason for we can doing so is that $D_{train}^{attacker}$ and $D_{test}^{attacker}$ can be regarded as a sampled subset from the aggregated dataset of all FL clients, whose distribution is not significantly different from the other clients. According to ML-Leaks [38], even it utilizes an existing dataset that comes from a different distribution than the target model's training data to train the shadow model, the separation boundary of member and non-member samples of the shadow model could generalize to the target model. For the scenarios considered in our work, we can directly treat the target model as the shadow model and the condition of the data distribution is much better than that of ML-Leaks and. Therefore, with $D_{train}^{attacker}$ and $D_{test}^{attacker}$, the attack model could learn the separation boundary of member and non-member samples which is shared across other FL clients.

Specifically, for each sample $(\mathbf{x}, y_{gt})$ in $D^{attacker}_{train}$, we measure its adversarial robustness of all possible labels with respect to every model in $[\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_T]$. For clarity, we denote the adversarial robustness of label y_k against $\mathcal{M}_t$ as r^t_k, where k is the predicted label that the FL model can take except for y_{gt}. Then for the attacker's local training data, the temporal evolution $\mathbf{r}$ is denoted as:

$$\begin{pmatrix} r^1_1 & r^2_1 & r^3_1 & \cdots & r^T_1 \\ r^1_2 & r^2_2 & r^3_2 & \cdots & r^T_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r^1_K & r^2_K & r^3_K & \cdots & r^T_K \end{pmatrix} \tag{10}$$

Then we label the temporal evolution $\mathbf{r}$ of all training samples with $\mathbf{In}$ and combine them together to get the collection of temporal evolutions $\mathbf{R_{In}}$. Then through the same operation, we can get the same combination $\mathbf{R_{Out}}$ for the attacker's testing data. It should be noted that the amount of training data used to generate $\mathbf{R_{In}}$ should be the same as the amount of testing data used to generate $\mathbf{R_{Out}}$. At the final step of TEAR, we utilize a supervised training algorithm on $\mathbf{R_{Out}}$ and $\mathbf{R_{In}}$ to obtain the inference model $\mathcal{A}_{MI}$.

Inferring Membership. Using the inference model $\mathcal{A}MI$, an adversarial client can easily determine whether a given sample belongs to the training data of other clients by directly querying the temporal evolution of adversarial robustness $\mathbf{r}$ obtained through $\mathcal{A}MI$. It's important to note that with $\mathcal{A}_{MI}$, we can conduct MIAs on multiple target samples simultaneously, as long as we can estimate the adversarial robustness of those samples.

C. Discussion: TEAR in White-box Case

In this paper, we assume the attacker is an honest-but-curious client, who can only access to the prediction interface of the global model during the training process. To this end, we design an adversarial example generation method by merely leveraging the predicted label of the target model.

In some cases, there may exist a part of stronger attackers in FL such as malicious clients, honest-but-curious central servers, or intermediary eavesdroppers, who can infiltrate the FL model to obtain some white-box information like the prediction probabilities or even internal model parameters. Then existing adversarial attacks, such as Deepfool [43] and C&W Attack [29], can be directly adopted in TEAR to measure the adversarial robustness of the target sample.

In our evaluation section, we compare the performance of our original TEAR with the informed TEAR that integrates Deepfool, SurFree, FMN, and C&W Attack. Please refer to Section V-D for the experimental settings and results.

V. Performance Evaluation

A. Experimental Settings

1) Datasets: We use the following five datasets for evaluation.

Adult[4]. Adult dataset includes $48,842$ records and each record contains 14 attributes, such as age, gender, education, marital status and occupation. The classification task is to predict whether a person makes over \$50K a year based on the census attributes.

MNIST[5]. MNIST dataset is a collection of $70,000$ handwritten digital images. Each image consists of 28×28 pixels and is normalized so that the digits are located at the center of the image. The label is a one-dimensional array with a length of 10, representing the specific digital value of the image.

CIFAR10[6]. CIFAR10 dataset is widely used for evaluating image recognition algorithms, which is composed of 10 classes of images with $6,000$ images per class. Each image consists of 32×32 pixels. Overall, there are $50,000$ training images and $10,000$ testing images.

Purchase[7]. Purchase dataset contains shopping histories of several thousand shoppers over one year, including many fields such as product name, store chain, quantity, and date of purchase. In particular, Purchase dataset has $197,324$ records but does not contain any class labels. Following Shokri el al. [35] and Salem el al. [38], we adopt K-Means algorithm to assign each data record with a class label. The numbers of classes include 2, 10, 20, 50, and 100, and each class corresponds to a purchase style.

FEMNIST[8]. Federated Extended MNIST dataset is an image classification dataset which consists of $805,263$ images with 62 classes. This dataset is constructed by dividing the data in Extended MNIST [44] into $3,550$ parts according to the source of the digit/character images. This dataset is inherently non-IID with data quantity and class distributions heterogeneity.

2) Target Models: We use the famous FedAvg algorithm [45] to train the target FL model. We set the number of federated clients to 5 and the iteration number of FL training to 10. During every training iteration, federated clients train the received FL model no more than 10 rounds on their own data. As for the data separation, we equally divide the used dataset into 5 parts and randomly assign each part to one client.

For different datasets, we use different global FL models, and the model structures are shown in Table I. Particularly, FC layer represents fully connected layer in DNN models. Conv. layer means convolutional layer and Pool. layer means

[4]https://archive.ics.uci.edu/ml/datasets/Adult
[5]http://yann.lecun.com/exdb/mnist
[6]http://www.cs.toronto.edu/ kriz/cifar.html
[7]https://www.kaggle.com/c/acquire-valued-shoppers-challenge/data
[8]https://leaf.cmu.edu

TABLE I
THE ARCHITECTURES OF FEDERATED MODELS.

Dataset	Model Architecture
Adult	2 FC layers
MNIST	2 Conv., 2Pool., and 3 FC layers
CIFAR10	2 Conv., 2Pool., and 3 FC layers
Purchase	3 FC layers
FEMNIST	2 Conv., 2Pool., and 2 FC layers

maxpooling layer in convolutional neural network (CNN) models.

3) Settings of TEAR: **Adversarial robustness estimation.** As discussed in Section IV-C, TEAR can also adopt existing adversarial attacks to obtain the adversarial robustness, when the attacker has the white-box access to the target FL model. Therefore, we perform TEAR in both white-box and black-box scenarios.

For the black-box case we mainly consider in this paper, we set the initial maximum number of iterations $I = 50$ and the number of gradient queries $B = 5000$. We use the cosine similarity as a measure of the loss function and SGD algorithm to update the samples. We set the learning rate $\eta = 0.3$ for CIFAR10 dataset and $\eta = 0.005$ for other datasets.

In the white-box case, we use Deepfool [43] to measure the distance from the target sample to the decision boundary of the target model. For the hyper-parameters of Deepfool, we set the maximum number of iterations to 500 and the amplification factor to 0.02.

Inference model construction. We adopt the method of supervised member inference attack, using sample-decision boundary distance matrix as the input value of the attack model, and the data belonging to the member (or not) as the label to train the binary classification inference model. We randomly select 500 samples from the adversarial client (half from training set and half from testing set) and use the corresponding adversarial robustness to construct the inference model. We train a gradient boosting model as our inference model with the open-source library Xgboost[9]. Then we construct an evaluation dataset by randomly selecting a part of records from the federated clients' training and testing data on which we evaluate the performance of our inference model.

4) Metrics: Since the inference attack essentially is a binary classification task, we adopt the standard metrics in ML filed, including *accuracy*, *precision* and *recall*, to evaluate the performance of TEAR. Specifically, *accuracy* represents the proportion of target samples whose membership properties are correctly inferred. *Precision* presents the proportion of the data samples predicted as members of the federated clients' training data that are indeed in. *Recall* presents the fraction of the training samples that we can correctly infer as the training samples of the federated clients.

To further compare the performance of our attack and the comparisons, we also report *F1-score* of MIAs which is primarily used to compare the performance of two classification models in ML field. *F1-score* is the harmonic mean of the *precision* and *recall* of MIAs, which can be calculated as:

$$F1\text{-}score = \frac{2(Precision \times Recall)}{Precision + Recall} \tag{11}$$

Generally, precision (resp.recall) measures the accuracy (resp. coverage) of MIAs. With F1-score, we could evaluate the performance of TEAR with both precision and recall.

5) Comparison Methods: We consider the following four MIAs for comparisons. The first one is proposed by Nasr et al. [17], which employs an auto-encoder architecture to reconstruct the signals associated with the membership property using the final layer outputs of the target FL model. The auto-encoder consists of an encoder with four MLP layers and a decoder with two MLP layers. Subsequently, the output of the encoder serves as the membership score for the target samples. With the membership score, this method utilizes the unsupervised spectral clustering model to cluster the target samples into two parts, and predicts the cluster with a lower prediction uncertainty as the member of training set.

[9]https://github.com/dmlc/xgboost

The second comparison is proposed by Choquette-Choo et al. [25]. For the target models that only output the predicted labels, this method uses a label-only adversarial attack, namely "HopSkipJump" [30], and sets a query budget of 1,000 to estimate the distance of the target sample to the target model's decision boundary. Specifically, HopSkipJump generates a random direction for the perturbation and iteratively improves this perturbation by employing gradient-based optimization to locate the direction that maximizes the loss function. If the distance is greater than a predefined threshold, this sample would be considered as a member of the training set. We determine the threshold with the local data of the adversarial FL client in our experiments.

The third comparison is proposed by Del et al. [46], which involves Auto-Attack [47] to generate an adversarial example for a given target sample from a trained model. Then the l_p $(p \in \{1, 2, \infty\})$ distance between the given sample and the generated adversarial example is calculated to determine the membership by a preset threshold value. In our experiments, we set the budget of iterations of Auto-Attack to 500 and determine the threshold by evaluating the attack accuracy on the local data of the adversarial FL client with varying values.

The fourth comparison is proposed by Liu et al. [48], which employs the knowledge distillation technique and a shadow dataset to replicate the target model's training process. Then the distilled model is treated as the shadow model and the intermediate training states are saved at various distillation epochs. The shadow dataset is evaluated on the sequence of intermediate shadow models, and the resulting training loss trajectories are recorded and used to construct an attack model to perform MIAs. In our experiments, we randomly select half of the training and testing data from all FL clients to create the shadow dataset for this attack. We designate the final trained FL model as the target model and construct the shadow model with 100 distillation epochs.

It is noteworthy that the first comparison method [17] is designed for FL models, while the rest comparison is specifically intended for FL models, the remaining comparison methods [25], [46], [48] are intended for MIAs against centralized models. In order to ensure fairness in our comparison, all comparison attacks are conducted against the trained FL global model. As for constructing the attack model and determining the attack threshold, we leveraged adversarial FL client's local training and testing data labeled with membership properties to provide the necessary information. We perform the above four attacks as well as TEAR on randomly selected samples from the benign federated clients' training and testing datasets, where the number of training samples is set equal to the number of testing samples, in order to maximize the uncertainty of inference (thus the baseline accuracy is 0.5, equivalent to

TABLE II
ACCURACY, PRECISION, RECALL, AND F1-SCORE (%) OF OUR METHODS AND FOUR COMPARISON METHODS (**BOLD** INDICATES THE HIGHEST ONE, AND $\underline{\text{UNDERLINE}}$ INDICATES THE SECOND HIGHEST ONE).

Attack	Metric	Adult	MNIST	CIFAR-10	Purchase2	Purchase10	Purchase20	Purchase50	Purchase100	FEMNIST
TEAR (black-box)	Accuracy	**0.610**	$\underline{0.740}$	**0.830**	**0.720**	**0.730**	$\underline{0.750}$	**0.800**	**0.820**	$\underline{0.730}$
	Precision	**0.656**	$\underline{0.731}$	$\underline{0.836}$	**0.722**	$\underline{0.741}$	$\underline{0.771}$	$\underline{0.789}$	$\underline{0.809}$	$\underline{0.742}$
	Recall	**0.667**	$\underline{0.760}$	**0.852**	**0.750**	$\underline{0.755}$	$\underline{0.810}$	$\underline{0.820}$	**0.917**	**0.831**
	F1-Score	**0.661**	$\underline{0.745}$	**0.844**	**0.736**	$\underline{0.748}$	$\underline{0.790}$	$\underline{0.804}$	**0.859**	$\underline{0.784}$
TEAR w/ Deepfool (white-box)	Accuracy	$\underline{0.610}$	**0.760**	$\underline{0.810}$	$\underline{0.710}$	$\underline{0.730}$	**0.760**	$\underline{0.770}$	$\underline{0.810}$	**0.770**
	Precision	$\underline{0.649}$	**0.732**	**0.850**	$\underline{0.721}$	**0.750**	**0.787**	**0.794**	**0.813**	**0.776**
	Recall	$\underline{0.661}$	**0.820**	$\underline{0.836}$	0.646	**0.764**	**0.814**	**0.833**	$\underline{0.881}$	$\underline{0.831}$
	F1-Score	$\underline{0.655}$	**0.774**	$\underline{0.843}$	0.681	**0.757**	**0.800**	**0.813**	$\underline{0.846}$	**0.797**
Nasr et al. [17]	Accuracy	0.524	0.560	0.665	0.549	0.581	0.604	0.624	0.674	0.606
	Precision	0.502	0.548	0.655	0.504	0.552	0.597	0.633	0.641	0.556
	Recall	0.528	0.592	0.684	0.563	0.592	0.629	0.657	0.691	0.634
	F1-Score	0.515	0.569	0.669	0.532	0.571	0.612	0.645	0.665	0.592
Choquette-Choo et al. [25]	Accuracy	0.587	0.601	0.785	0.687	0.715	0.729	0.755	0.808	0.715
	Precision	0.565	0.628	0.739	0.639	0.703	0.714	0.764	0.794	0.723
	Recall	0.643	0.663	0.815	0.699	0.719	0.732	0.780	0.836	0.737
	F1-Score	0.602	0.645	0.775	0.668	0.711	0.723	0.771	0.814	0.730
Del et al. [46]	Accuracy	0.580	0.700	0.730	0.660	0.680	0.720	0.720	0.740	0.670
	Precision	0.556	0.686	0.752	0.672	0.695	0.692	0.744	0.764	0.647
	Recall	0.610	0.712	0.788	0.698	0.703	0.769	0.773	0.795	0.712
	F1-Score	0.582	0.699	0.770	0.685	0.699	0.728	0.758	0.779	0.680
Liu et al. [48]	Accuracy	0.610	0.730	0.790	0.700	0.700	0.740	0.760	0.760	0.720
	Precision	0.636	0.702	0.814	0.716	0.716	0.732	0.771	0.785	0.730
	Recall	0.644	0.759	0.835	$\underline{0.748}$	0.748	0.764	0.790	0.823	0.796
	F1-Score	0.640	0.729	0.824	$\underline{0.732}$	0.732	0.748	0.780	0.804	0.762

the random guess).

B. *Performance of TEAR*

We first evaluate the performance of TEAR on diverse types of datasets to showcase the generalization of our attack against different FL models. The parameter settings of TEAR and comparison methods are the same as those described in Section V-A unless otherwise specified. In this section, we utilized the grid search method provided by Scikit-Learn library to adjust the hyperparameters of the attack models, in order to obtain the best attack performance that TEAR can achieve. This will help us analyze the utmost risk of privacy leakage during the FL training process. The experiment results are shown in Table II.

In order to clarify the overfitting level of target models which is a vital factor for MIAs, we also record the performance of the target FL models. Table III shows the training and testing accuracy of target models on each used dataset. The difference between the model's training and testing accuracy indicates the overfitting level.

From the results in Table II, we can find that even with merely black-box access of the target models, our attack can achieve a strong inference performance compared with four comparison attacks. It is noteworthy that TEAR's performance on certain datasets with white-box access to the target FL model is poorer than that with black-box access. The main reason is that the adversarial robustness estimated by Deepfool is less accurate compared to our estimation method. The same issue arises in the comparison work [25] which uses "Hop-SkipJump" method to estimate the adversarial robustness. We will further discuss the impact of the selection of adversarial robustness estimation method in Section V-D.

As for the specific results, for Adult dataset, TEAR with black-box access can get an attack accuracy of 0.610, which is better than the two comparisons. For MNIST dataset, TEAR achieves an attacking accuracy of 0.740. For Purchase dataset, with the increase of the number of classes, the number of decision boundaries increases and more membership information leaks. As the number of class increasing, the overfitting level of the target model increases, and the attacking performance of TEAR becomes even better.

Specifically, for Purchase100 dataset, the precision and recall of TEAR can reach to 0.809 and 0.917, respectively. As for Purchase2, TEAR only achieves a precision of 0.722 and a recall of 0.750. For CIFAR10 dataset, the attack of TEAR gets a precision of 0.836 and a recall of 0.859. For the FEMNIST dataset, our MIA also gets relatively good results. The attacking precision is 0.742 and the recall is 0.831, which demonstrates that TEAR can be extended to non-iid data in a more complex federated environment.

Overall, no matter black-box or white-box access TEAR has to the target models, it demonstrates a superior attack performance against most FL models compared to the four comparisons. We compared our white-box attack (i.e., TEAR with Deepfool) with the methods proposed in [25], [46] that only estimate the adversarial robustness on the final trained model. Our comparison confirmed that the temporal evolution of adversarial robustness poses a serious threat to the membership privacy of FL models' training data. We also compare our attack with [48] that only uses the temporal trajectory of the training losses. We find that adversarial robustness can provide much more detailed membership features of the target samples. Based on the experimental results, we can

TABLE III
THE PERFORMANCES OF FEDERATED MODELS.

Dataset	Training Accuracy	Testing Accuracy
Adult	0.8505	0.8490
MNIST	0.9909	0.9859
CIFAR10	0.7801	0.6201
Purchase2	0.9830	0.9801
Purchase10	0.9518	0.9301
Purchase20	0.9276	0.8782
Purchase50	0.9235	0.8103
Purchase100	0.9151	0.7503
FEMNIST	0.9492	0.8608

conclude that both our method and Liu et al. [48] exceed the performance of the other three comparison methods. This is because during the training process, the attacker can utilize multiple snapshots of target models and obtain the temporal evolution of membership features.

C. Impact of Quality of Adversarial Example

In TEAR, the influence of parameters (especially the numbers of adversarial iterations and queries) will affect the quality of the generated adversarial samples, so that the distance from the original sample to the decision boundary will be changed. In order to achieve a fair comparison of different adversarial iterations and queries, we construct our attack models using the same set of hyperparameters. This approach ensures that any performance differences are primarily due to variations in the parameters of the adversarial sample generation, rather than differences in the hyper-parameters of the attack models. In the following, we evaluate the two parameters using Purchase100 and CIFAR10 datasets.

Impact of number of adversarial iterations. The number of adversarial iterations refers to the times of updating adversarial samples through SGD optimization. In order to obtain the adversarial sample closest to the original sample, we set the gradient query times as 5,000 through multiple experiments, and the number of iterations starts from 10 to 90.

Fig. 5 shows the result of impact of adversarial iterations. We can see that for both Purchase100 and CIFAR10 datasets, the more adversarial iterations, the better the inference performance. For example, the precision and recall are 0.800 and 0.825 for 90 iterations compared to 0.633 and 0.655 for 10 iterations on CIFAR10 dataset. When the number of iterations is small, the adversarial attack model will not converge. Thus the overall cosine similarity between the normal vector of the decision boundary and the vector from the target sample to its adversarial example would be small. This leads to the higher adversarial disturbance and lower inference performance. The attack accuracy reaches the highest when the number of iterations comes to a certain value, where the adversarial attack has converged and gets the closest point to the target sample.

Impact of number of adversarial queries. The number of adversarial queries refers to the number of disturbed samples generated when we use Monte Carlo algorithm to estimate the gradient of adversarial data on the decision boundary. As

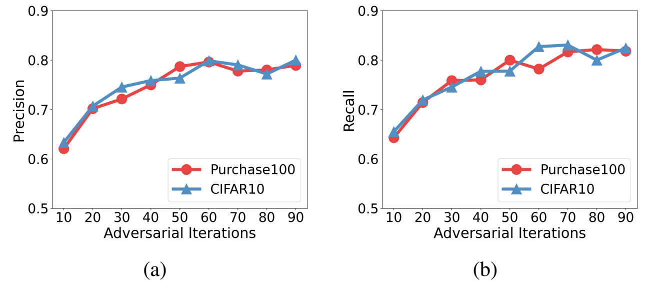


Fig. 5. Precision and recall of TEAR vs. number of iterations.

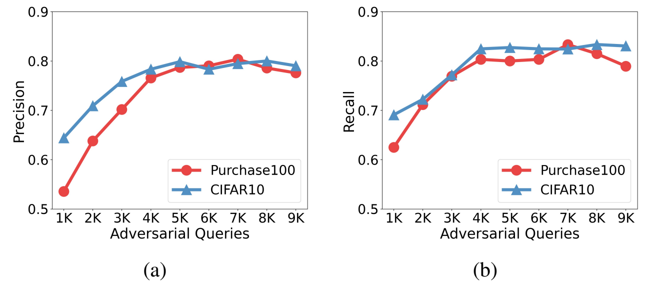


Fig. 6. Precision and recall of TEAR vs. number of queries.

stated before, through repeated experiments, we set the optimal number of adversarial iterations as 50 for the Purchase100 dataset and 60 for CIFAR10 dataset. The number of adversarial queries is evaluated every 1,000 rounds from 1,000 until it reached 9,000.

From the results in Fig. 6, we can see that better inference performance can be achieved with the increase of query times. For CIFAR10 dataset, the precision increases from 0.644 to 0.800 and the recall increases from 0.691 to 0.833. While the query times continue to increase, the attack performance started to be stable. Small query times cause large gradient estimation deviation of the adversarial sample at the decision boundary and the calculation error of loss value, which reduce the quality of adversarial sample and interfere with TEAR.

D. Comparison of Adversarial Attacks

In this section, we dive into the performance of our adversarial attack. The performance of the adversarial attack directly determines the quality of adversarial samples, thus affecting the distance estimation from data to decision boundary. We select a white-box attack, i.e. C&W [29], as our baseline. In order to evaluate the performance of our adversarial example generation, we compare the cosine similarity between the normal vector of the decision boundary and the vector from the target sample to its adversarial example generated by TEAR, SurFree [49], FMN [50], and Deepfool [43]. Then we briefly introduce each comparison.

SurFree. This method initially identifies the adversarial example to the target sample and then seeks a perturbation that shifts the target sample to the opposite side of the decision boundary of the target model.

FMN. This attack estimates the gradient of the loss function in the feature space, and then perturbs the input along the estimated gradient direction.

Deepfool. This algorithm regards the target model's decision

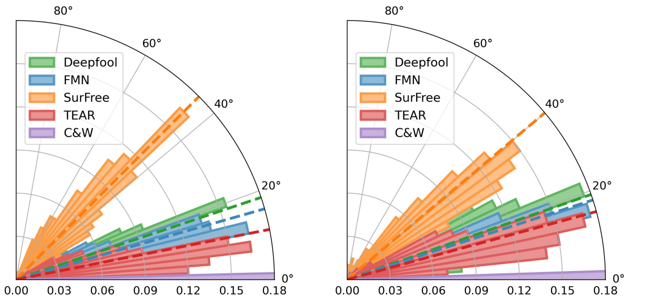
TABLE IV
ACCURACY, PRECISION, RECALL, AND F1-SCORE (%) UNDER DIFFERENT ITERATIONS.

(a) Purchase100 Dataset

Attack Types		**TEAR (black-box)**				**TEAR w/ Deepfool (white-box)**			
Metrics		Accuracy	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score
Observed Iteration	1, 2, ..., 10	0.820	0.809	0.917	0.859	0.810	0.813	0.881	0.846
	10	0.720	0.722	0.867	0.788	0.730	0.750	0.836	0.791
	5	0.660	0.703	0.750	0.726	0.730	0.714	0.784	0.748

(b) CIFAR10 Dataset

Attack Types		**TEAR (black-box)**				**TEAR w/ Deepfool (white-box)**			
Metrics		Accuracy	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score
Observed Iteration	1, 2, ..., 10	0.830	0.836	0.852	0.844	0.810	0.850	0.836	0.843
	10	0.800	0.804	0.833	0.818	0.730	0.727	0.842	0.781
	5	0.760	0.759	0.815	0.786	0.690	0.702	0.741	0.721



(a) Purchase100 Dataset (b) CIFAR-10 Dataset

Fig. 7. Comparison of adversarial attacks. The radius of each sector represents the data proportion of the corresponding angle, and the dashed line represents the average angle.

boundary as a hyperplane that separates the input data space into regions associated with different output classes. Then it iteratively generates slight perturbations that move the target sample towards the vertical direction of the decision boundary.

Fig. 7 visualizes the histogram of deviation: $\theta = \arccos \frac{s_t s_o}{||s_t||_2||s_0||_2}$, where s_t is the vector between the original data and the generated adversarial data and s_0 is the baseline vector. The results show that our adversarial attack in TEAR is much closer to the baseline than SurFree, FMN, and Deepfool, which presents that TEAR achieves a better performance. The complexity and randomness of the data result in a wide range of angle differences between vectors in both TEAR and the comparison attacks, but the average angle of TEAR is much smaller than that of comparisons. The main reason for TEAR's ability to generate adversarial examples with direction closer to the baseline attributed to its use of the geometric relationship between the target sample and the decision boundary of the target models. By constraining the direction of the adversarial examples along the normal vector of the target model's decision boundary, TEAR is able to directly find the adversarial examples around the target sample. This approach, which is not taken into account in current adversarial attacks, leads to a smaller direction deviation between the target sample and the adversarial example generated by TEAR.

Taking Purchase100 dataset as an example, the mean angle deviation between TEAR and C&W attack reaches to about 10° (the red dash line), while FMN's and Deepfool's are close to 15° (the blue dash line) and 20° (the green dash line) respectively. In such a case, the angle deviation of SurFree even is larger than 40°. The angles of TEAR are mainly distributed within 15° difference from that of C&W attack. However, for the Surfree, FMN and Deepfool attacks, there are a lot of angle deviations which are more than 20°. In addition, we can observe a similar phenomenon from the experiment results of CIFAR10 dataset. In general, TEAR can generate the adversarial examples with a smaller angle deviation from the target model's decision boundary. The main reason is that TEAR uses the geometric relationship between the target sample and the decision boundary of the target models to directly find the adversarial examples along the decision boundary around the target sample. This approach constrains the direction of the adversarial examples and is not taken into account in current adversarial attacks.

E. Validity of TEAR

For TEAR, we use the information from models on each global epoch. We capture model member leakage by means of the distance difference between the training and testing data to the decision boundaries of the multiple intermediate FL models. In this section, we discuss validity of TEAR by just using one observed snapshot of FL model and performing the attack by distance features of this FL model. We take the global model of the last round and some rounds in the middle of FL for experiments, in both black-box and white-box.

Table IV shows the results under different observed iterations. In any case, compared to the attack using all model snapshot observed through the whole training process, the attack on one model snapshot is much less effective. If we use the model disseminated at the end of FL training process, there is a little bit better performance than the middle model, mainly because the overfitting degree of the model becomes higher with the training of the model. For instance, in Purchase100 dataset, TEAR can get 0.82 accuracy and the white-box version gets 0.81. However, the results become 0.72 and 0.73 if we merely use the FL model disseminated by the central server at 10_{th} iteration and the accuracy is even lower for the model grasped at 5_{th} training iteration. Overall, we can find that

TABLE V
ACCURACY, PRECISION, RECALL, AND F1-SCORE (%) UNDER THE DEFENSE OF DP-SGD AND MEMGUARD.

(a) Purchase100 Dataset

Attack Types		**TEAR (black-box)**				**TEAR w/ Deepfool (white-box)**			
Metrics		Accuracy	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score
Defense	No Defense	0.820	0.809	0.917	0.859	0.810	0.813	0.881	0.846
	DP-SGD	0.780	0.810	0.900	0.830	0.790	0.776	0.918	0.841
	MemGuard	0.820	0.809	0.917	0.859	0.810	0.813	0.881	0.846

(b) CIFAR-10 Dataset

Attack Types		**TEAR (black-box)**				**TEAR w/ Deepfool (white-box)**			
Metrics		Accuracy	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score
Defense	No Defense	0.830	0.836	0.852	0.844	0.810	0.850	0.836	0.843
	DP-SGD	0.750	0.776	0.809	0.792	0.800	0.837	0.837	0.837
	MemGuard	0.830	0.836	0.852	0.844	0.810	0.850	0.836	0.843

with all model snapshots of FL models throughout the training process, our MIA can enrich the membership information and thus get better attacking performance.

VI. DISCUSSION ON DEFENSES

A. Existing Defenses

In FL, the attacker can easily obtain user's data and model information through MIAs. In order to reduce the membership leakage, many defense methods are proposed recently. We largely classify these defenses into the following three categories:

Differential privacy. DP [51] is a widely used privacy protection method in many applications, such as location-based services [52] and healthcare services [53]. Many studies [25], [26], [32], [54], [55] have applied DP to learning models to mitigate MIAs. Current DP technologies hide real data and achieve defensive effects by adding noise to the training data of users [56], the object function [57], or the gradient computed by gradient descent during the training process [21], [54], [58]. For instance, Abadi et al. [21] propose DP version of stochastic gradient descent (DP-SGD), which adds random Gaussian noise to the gradient during model optimization with SGD. Our MIA may be affected by DP with perturbing the gradient, because the decision boundary will be changed with the noise on gradients during the FL training and TEAR will be misled for distinguishing the members and non-members.

Confidence score masking. Confidence score masking is mainly used to defend against MIAs in black-box case to provide membership privacy guarantees. It aims to hide or change the true confidence scores returned by the target classifier so that both members and non-members look like similar examples to the attack model. For instance, the target classifier can only provide top-k confidence scores instead of a complete prediction vector to the attacker. As the first defense with formal utility-loss in this category, MemGuard [32] adds noise to posterior probabilities of the target model with a certain probability and forms a defense classifier simulating the attack classifier to mislead the membership inference. It would be interesting to apply this countermeasure in FL with adding noise to posterior probabilities of the each local models to affect the global model and defend against MIAs.

Regularization. Regularization is a very common defense in MIAs and it mitigates attacks by reducing the overfitting degree of target models. It can help the model generalize better to testing data and reduce the difference of members and non-members. Existing regularization defense methods including L2-norm regularization [25], [35], [59], dropout [60], data argumentation [61], [62], model stacking [38], and adversarial regularization [59].

B. Defense Evaluation

In our experiments, we use DP-SGD and MemGuard as examples to evaluate our attacking performance against the MIA defenses. In our experiment, we implement DP-SGD using the open-source package Opacus[10] directly, and adopt the original implementation of MemGuard[11]. We observe our attack in both black-box and white-box cases.

Table V shows the results against different defenses. We find that DP-SGD reduces our attack accuracy to some extent, which suggests that DP-SGD interferes with our attack by changing the training of the local model, and thus degrades the prediction performance of the target model. However, the impact on the overall attack performance is not obvious, and in some cases it is even better than the undefended, indicating that our attack model has some stability. Black-box attacks under CIFAR10 datasets are most affected, mainly due to the complexity of the dataset and the high requirement on the predictive ability of the target model in TEAR. We also notice that MemGuard has no effect on our attacks. Although MemGuard changes the posterior probabilities of the target model, the prediction label of the model will not be affected. Our attacks only exploit the predictive labels of the inputs, so MemGuard cannot defend against our attacks at all.

It should be noted that our black-box TEAR demonstrates superior performance compared to the white-box TEAR attack with Deepfool in some cases. The main reason is that TEAR utilizes the adversarial robustness of the target sample with respect to the target model to conduct inference attacks. The attack effectiveness of TEAR is directly related to the accuracy of the estimated adversarial robustness. As discussed in the

[10]https://github.com/pytorch/opacus
[11]https://github.com/jjy1994/MemGuard

previous section (c.f. Fig. 7), although the distribution of adversarial robustness estimated by the black-box TEAR and Deepfool are similar, Deepfool performs more accurately than our method on a part of samples. As a consequence, our black-box TEAR attack can generally achieve comparable performance to the white-box TEAR attack with Deepfool.

C. Possible Defenses

From the experiment results shown in Table V, we could see that TEAR could breach MemGuard and DP-SGD by achieving reasonable attack performance. Different from the previous MIAs, TEAR performs MIA at the model training stage and uses the difference of the temporal evolution of the adversarial robustness between the training and non-training data. Therefore, the core idea of defending our MIA is to reduce the difference between the training data and the non-training data of the model during training process. The possible defense against our MIA could be noise gradient and adversarial training.

Noise gradient. Adding noise to gradients is a common defense strategy in FL. As for defending against TEAR, the client can add well-designed noise to gradients before sending them to FL server at every training round, and thus increasing the distance to the decision boundary of the training data with respect to the target model.

Adversarial training. The original goal of adversarial training is to enhance the robustness of models against the adversarial examples. As for defending against TEAR, the FL server could involve the adversarial training into the FL training process, and thus increasing the adversarial robustness of the FL model with respect to the training data.

With these possible defenses, the adversarial robustness of the training data will be enhanced and thus the difference to the non-training data would be reduced. This would increase the difficulty for TEAR to perform MIA, thereby protecting the data privacy of FL clients.

VII. Related Work

A. MIA against ML Models

Shokri et al. [35] first study MIA against ML models and present the shadow training technique, which perhaps is the most widely used attack paradigm in MIA field [63]. They use a shadow dataset, which has the similar distribution with the target model's training data, to construct multiple shadow models to mimic the prediction behavior of the target ML model. Then they construct multiple inference models based on the prediction of the shadow models, which can directly determine the membership property of the target sample. Subsequently, Salem et al. [38] present ML-Leaks and show that it is possible to perform the same attack with only one shadow model instead of multiple shadow models. Li et al. [39] also leverage the shadow training technique and present an instance-probability attack. They use the prediction of shadow and testing data on the shadow models to select a threshold for the probability of correct labels, which is then used to perform MIA. Truex et al. [64] demonstrate how shadow models can be leveraged in adversarial ML and expose MIA vulnerabilities through the perspectives of data skewness.

Except for leveraging the shadow training technique, many researchers use other information of the target model to perform MIA. A part of researchers find that the training process of ML models is to fit the whole training data, which may not generalize well on the testing data. As a consequence, there will be a difference between the training and non-training data with respect to prediction correctness [36], prediction loss [37], and parameter gradient [17]. Then they treat these information as the membership feature to perform MIA. Furthermore, Hui et al. [65] exploit the prediction probability distributions of the target model, and design a differential comparison method with a synthetic dataset to attack against a trained DNN model. Song et al. [66] propose a new metric called the privacy risk score, based on the modification of prediction entropy. With this metric, they could identify samples with high leakage risks of membership information.

More recently, a part of researchers are inspired by the adversarial example studies [27], [67] and attempt to perform MIA with the sample distance to decision boundary of ML models. The intuition behind these works follows a general observation that an ML model is more confident in predicting its training data samples. Thus it requires a larger perturbation to make the target model misclassify a member sample than a non-member sample. Choquette et al. [25] and Li et al. [26] make use of existing adversarial attacks, including HopSkipJump attack [30] and C&W attack [29] , to estimate the sample distance to the decision boundary of the target model. Then they infer the given sample as a member if the distance is larger than a predefined threshold, and vice versa. Recently, Grosso et al. [46] make use of Auto-Attack [68] to generate the adversarial example for the target sample. Then the distance between the target sample and its corresponding adversarial example was then utilized to conduct MIAs.

We notice that, in parallel to our work, a recent work TrajectoryMIA [48] also utilizes information during the training process to conduct MIAs. The authors employ knowledge distillation techniques to train a shadow model on a supplementary dataset that has the same distribution as the target model's training data. They subsequently train an attack model using the prediction loss trajectory of the auxiliary dataset acquired during the shadow model's training procedure. Although both our work and TrajectoryMIA utilize the discrepancy between training and testing data with respect to the model during the training process, TrajectoryMIA focuses solely on the prediction loss changes. In contrast, we dive into the connection between the decision boundary shift of the target model and the target sample with the training process continues, which can offer a more detailed geometric perspective for MIAs.

B. MIA against FL Models

Like ML models, FL models also confront the threat of MIAs. Correspondingly, a part of researchers apply the shadow training technique [35] to FL models, and analyze the privacy leakage risk of the training data of FL models. Pustozerova et al. [13] find that the membership information of the training data confront greater risk compared with the centralized ML models. Mo et al. [69] perform MIAs on the well-trained FL model and its last layer's outputs, and demonstrate that the FL model deployed in the mobile computing scenario still confronts the risk of membership leakage. Since the training data of each federated client are independent, Zhang et al. [14] make use of generative adversarial network (GAN) to overcome the lack of shadow data.

Since the training procedure of FL needs to exchange the parameter updates or gradients, FL provides a new avenue for MIAs. Nasr et al. [17] present a passive MIA by extracting the membership information of the target samples from the observed model updates. Pichler et al. [19] introduce a simple MIA with a malicious central server which only relies on a single training step. They observe the parameter changes of two successive training rounds with respect to the target sample, and then infer this sample's membership property with a threshold test. Then some researchers modify the FL updates and present several active MIAs. Melis et al. [16] exploit the fact that SGD optimization updates model parameters in the direction of minimizing the training loss and present the gradient ascent attack. They reverse the direction of model updates with respect to the target sample, and force the FL model to reveal more information about the target sample. Xu et al. [18] propose an MIA for conspired malicious clients to adjust their private training data strategically, so that a certain parameter of the FL model can form meaningful signals of a pattern's appearance. Yuan et al. [70] analyze the membership risks for federated recommender systems and aim to infer target client's interacted item set by observing the target client's parameter updates.

Zari et al. [15] propose a simple MIA by leveraging the observation that the temporal evolutions of scores of the true labels for members and non-members of training data are distinguishable. Correspondingly, they collect the prediction scores of a dataset which contains both training and testing samples during the FL training process, and construct a binary attack model with the collected predictions. Given a target sample, they can directly determine the membership property with the attack model.

It is observed that the most MIAs in FL require the access to the parameter updates and even the modification of the training process of the FL model. However, in practice, the model parameters and gradients are usually encrypted and the training process is executed with a secure environment. Against this background, we consider a most restricted scenario and present a practical MIA, that merely requires the access to the prediction labels of the target model, in which most if not all existing FL MIAs would fails since it is almost impossible to obtain the necessary information about the target model, not to mention manipulating the training process.

VIII. Conclusion

In this paper, we have presented TEAR, a novel MIA against FL models by exploring the temporal evolution of adversarial robustness between the training and non-training data. TEAR requires neither the modification of the standard FL training process nor the access to the FL model's parameters and gradients. Experiments on five realistic datasets demonstrate that TEAR can achieve better attacking performance compared with existing MIAs, and also bypass two classic MIA defenses. We envision our work as a solid step in FL towards revealing privacy leakage risks of the training data during the model training stage and shed light on designing more effective defense mechanisms against MIAs in FL systems.

References

[1] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Federated learning of deep networks using model averaging," *CORR, arXiv: 1602.05629*, 2016.

[2] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of ACM CCS*, 2017, pp. 1175–1191.

[3] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[4] J. Chen, X. Zhang, R. Zhang, C. Wang, and L. Liu, "De-pois: An attack-agnostic defense against data poisoning attacks," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 3412–3425, 2021.

[5] 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.

[6] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *CoRR, arXiv: 1811.03604*, 2018.

[7] 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.

[8] F. Liang, W. Pan, and Z. Ming, "Fedrec++: Lossless federated recommendation with explicit feedback," in *Proceedings of AAAI*, vol. 35, no. 5, 2021, pp. 4224–4231.

[9] H. Hu, Z. Salcic, L. Sun, G. Dobbie, and X. Zhang, "Source inference attacks in federated learning," in *Proceedings of IEEE ICDM*, 2021, pp. 1102–1107.

[10] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients-how easy is it to break privacy in federated learning?" *Proceedings of NeurIPS*, vol. 33, pp. 16 937–16 947, 2020.

[11] A. Boutet, T. Lebrun, J. Aalmoes, and A. Baud, "MixNN: Protection of federated learning against inference attacks by mixing neural network layers," *CoRR, arXiv: 2109.12550*, 2021.

[12] X. Luo, Y. Wu, X. Xiao, and B. C. Ooi, "Feature inference attack on model predictions in vertical federated learning," in *Proceedings of IEEE ICDE*, 2021, pp. 181–192.

[13] A. Pustozerova and R. Mayer, "Information leaks in federated learning," in *Proceedings of NDSS*, 2020.

[14] 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.

[15] O. Zari, C. Xu, and G. Neglia, "Efficient passive membership inference attack in federated learning," in *Proceedings of NeurIPS PriML Workshop*, 2021.

[16] 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.

[17] 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.

[18] X. Xu, J. Wu, M. Yang, T. Luo, X. Duan, W. Li, Y. Wu, and B. Wu, "Information leakage by model weights on federated learning," in *Proceedings of Workshop on PPMLP*, 2020, pp. 31–36.

[19] G. Pichler, M. Romanelli, L. R. Vega, and P. Piantanida, "Perfectly accurate membership inference by a dishonest central server in federated learning," *CoRR arXiv: 2203.16463*, 2022.

[20] L. Lyu, H. Yu, and Q. Yang, "Threats to federated learning: A survey," *CoRR, arXiv:2003.02133*, 2020.

[21] 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.

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

[23] S. Tian, G. Yang, and Y. Cai, "Detecting adversarial examples through image transformation," in *Proceedings of AAAI*, vol. 32, no. 1, 2018.

[24] S. Hu, T. Yu, C. Guo, W.-L. Chao, and K. Q. Weinberger, "A new defense against adversarial images: Turning a weakness into a strength," *Proceedings of NeurIPS*, vol. 32, 2019.

[25] C. A. Choquette-Choo, F. Tramer, N. Carlini, and N. Papernot, "Label-only membership inference attacks," in *Proceedings of ICML*, 2021, pp. 1964–1974.

[26] Z. Li and Y. Zhang, "Membership leakage in label-only exposures," in *Proceedings of ACM CCS*, 2021, pp. 880–895.

[27] S. Rezaei and X. Liu, "On the difficulty of membership inference attacks," in *Proceedings of IEEE/CVF CVPR*, 2021, pp. 7892–7900.

[28] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *CoRR, arXiv: 1412.6572*, 2014.

[29] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proceedings of IEEE S&P*, 2017, pp. 39–57.

[30] J. Chen, M. I. Jordan, and M. J. Wainwright, "Hopskipjumpattack: A query-efficient decision-based attack," in *Proceedings of IEEE S&P*, 2020, pp. 1277–1294.

[31] L. Grinsztajn, E. Oyallon, and G. Varoquaux, "Why do tree-based models still outperform deep learning on tabular data?" *CoRR, arXiv: 2207.08815*, 2022.

[32] J. Jia, A. Salem, M. Backes, Y. Zhang, and N. Z. Gong, "Memguard: Defending against black-box membership inference attacks via adversarial examples," in *Proceedings of ACM CCS*, 2019, pp. 259–274.

[33] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu, "Federated learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 13, no. 3, pp. 1–207, 2019.

[34] X. Yin, Y. Zhu, and J. Hu, "A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions," *ACM Computing Surveys*, vol. 54, no. 6, pp. 1–36, 2021.

[35] 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.

[36] A. Sablayrolles, M. Douze, C. Schmid, Y. Ollivier, and H. Jegou, "White-box vs black-box: Bayes optimal strategies for membership inference," in *Proceedings of ICML*, 2019, pp. 5558–5567.

[37] 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.

[38] A. Salem, Y. Zhang, M. Humbert, P. Berrang, 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.

[39] J. Li, N. Li, and B. Ribeiro, "Membership inference attacks and defenses in supervised learning via generalization gap," *CoRR, arXiv: 2002.12062*, 2020.

[40] G. Liu, C. Wang, K. Peng, H. Huang, Y. Li, and W. Cheng, "SocInf: Membership inference attacks on social media health data with machine learning," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 5, pp. 907–921, 2019.

[41] A. El Ouadrhiri and A. Abdelhadi, "Differential privacy for deep and federated learning: A survey," *IEEE Access*, vol. 10, pp. 22 359–22 380, 2022.

[42] Q. Li, Z. Wen, Z. Wu, S. Hu, N. Wang, Y. Li, X. Liu, and B. He, "A survey on federated learning systems: vision, hype and reality for data privacy and protection," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[43] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of IEEE/CVF CVPR*, 2016, pp. 2574–2582.

[44] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik, "Emnist: Extending mnist to handwritten letters," in *Proceedings of IEEE IJCNN*, 2017, pp. 2921–2926.

[45] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of AISTATS*, 2017, pp. 1273–1282.

[46] 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.

[47] F. Croce and M. Hein, "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks," in *Proceedings of ICML*. PMLR, 2020, pp. 2206–2216.

[48] 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.

[49] T. Maho, T. Furon, and E. Le Merrer, "Surfree: A fast surrogate-free black-box attack," in *Proceedings of IEEE/CVF CVPR*, 2021, pp. 10 430–10 439.

[50] M. Pintor, F. Roli, W. Brendel, and B. Biggio, "Fast minimum-norm adversarial attacks through adaptive norm constraints," in *Proceedings of NeurIPS*, 2021, pp. 20 052–20 062.

[51] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," *Journal of Privacy and Confidentiality*, vol. 7, no. 3, pp. 17–51, 2016.

[52] C. Xu, Y. Ding, C. Chen, Y. Ding, W. Zhou, and S. Wen, "Personalized location privacy protection for location-based services in vehicular networks," *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[53] Y. Zhao and J. Chen, "A survey on differential privacy for unstructured data content," *ACM Computing Surveys (CSUR)*, vol. 54, no. 10s, pp. 1–28, 2022.

[54] B. Jayaraman and D. Evans, "Evaluating differentially private machine learning in practice," in *Proceedings of USENIX Security Symposium*, 2019, pp. 1895–1912.

[55] K. Leino and M. Fredrikson, "Stolen memories: Leveraging model memorization for calibrated white-box membership inference," in *Proceedings of USENIX Security*, 2020, pp. 1605–1622.

[56] J. C. Duchi, M. I. Jordan, and M. J. Wainwright, "Local privacy and statistical minimax rates," in *Proceedings of IEEE FOCS*, 2013, pp. 429–438.

[57] R. Iyengar, J. P. Near, D. Song, O. Thakkar, A. Thakurta, and L. Wang, "Towards practical differentially private convex optimization," in *Proceedings of IEEE S&P*, 2019, pp. 299–316.

[58] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of ACM CCS*, 2015, pp. 1310–1321.

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

[60] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[61] D. Yu, H. Zhang, W. Chen, J. Yin, and T.-Y. Liu, "How does data augmentation affect privacy in machine learning?" in *Proceedings of AAAI*, vol. 35, no. 12, 2021, pp. 10 746–10 753.

[62] Y. Kaya and T. Dumitras, "When does data augmentation help with membership inference attacks?" in *Proceedings of ICML*, 2021, pp. 5345–5355.

[63] H. Hu, Z. Salcic, L. Sun, G. Dobbie, P. S. Yu, and X. Zhang, "Membership inference attacks on machine learning: A survey," *ACM Computing Surveys*, vol. 54, no. 11, pp. 1–37, 2022.

[64] S. Truex, L. Liu, M. E. Gursoy, W. Wei, and L. Yu, "Effects of differential privacy and data skewness on membership inference vulnerability," in *Proceedings of IEEE TrustCom*, 2019.

[65] B. Hui, Y. Yang, H. Yuan, P. Burlina, N. Z. Gong, and Y. Cao, "Practical blind membership inference attack via differential comparisons," in *Proceedings of NDSS*, 2021.

[66] L. Song and P. Mittal, "Systematic evaluation of privacy risks of machine learning models." in *Proceedings of USENIX Security Symposium*, vol. 1, no. 2, 2021, p. 4.

[67] J. Cohen, E. Rosenfeld, and Z. Kolter, "Certified adversarial robustness via randomized smoothing," in *Proceedings of ICML*, 2019, pp. 1310–1320.

[68] F. Croce and M. Hein, "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks," in *Proceedings of ICML*. PMLR, 2020, pp. 2206–2216.

[69] F. Mo, H. Haddadi, K. Katevas, E. Marin, D. Perino, and N. Kourtellis, "Ppfl: privacy-preserving federated learning with trusted execution environments," in *Proceedings of ACM Mobisys*, 2021, pp. 94–108.

[70] W. Yuan, C. Yang, Q. V. H. Nguyen, L. Cui, T. He, and H. Yin, "Interaction-level membership inference attack against federated recommender systems," *CoRR, arXiv:2301.10964*, 2023.