Efficient Privacy Auditing in Federated Learning

Hongyan Chang
National University of Singapore

Brandon Edwards
Intel Corporation

Anindya S. Paul
University of Florida

Reza Shokri
National University of Singapore

Abstract

How much information does the federated learning (FL) process leak about a participating party's local training data? Under privacy regulations such as GDPR, participating parties have the right to know the potential privacy risks to their data. Membership inference attack (MIA) is a popular auditing tool for assessing such privacy risks in machine learning. However, adapting existing MIA solutions to FL often demands substantial computation resources (e.g., for training deep learning attack models), potentially impeding FL and burdening parties. Moreover, they typically focus on isolated snapshots of the model, failing to exploit the evolving membership information leaked during the FL process, thus underestimating privacy risks. To address these challenges, we provide an efficient and effective solution for parties to audit privacy during FL. The idea is to compute the *slope* of certain model performance metrics (e.g., prediction confidence, loss, and rescaled logit of the local data) across FL rounds and then use the outcome to differentiate members and non-members. As these metrics are *automatically computed* in the process in FL, our solution imposes very little computation overhead, and a party can seamlessly integrate it into FL without disrupting the training process. We validate the effectiveness of the slope signal under a wide spectrum of FL settings and real-world datasets. Finally, built upon the slope signal, we present a more advanced MIA solution for further enhancing the accuracy of privacy auditing, catering to parties seeking a more precise evaluation of privacy risks through increased computation investment.

1 Introduction

Federated Learning (FL) [28] enables multiple parties (namely, data holders) who do not trust each other to train a global model collaboratively using their sensitive datasets. During the training process, each party repeatedly downloads the global model, updates the model using her local dataset, and sends the *local update* back to the server. While the training data remains unseen, the shared local updates can inadvertently leak substantial information about the local sensitive data [29, 31, 45], leading to *privacy concerns*. At the same time, recent data policies such as the European Union's General Data Protection Regulation (GDPR) [1] also legalize the safeguarding of data, including Data Protection Impact Assessment (DPIA). These situations underscore the pressing need for privacy auditing in federated learning, which has been widely used in sensitive domains, including healthcare, recruitment, and loan evaluation [34, 40].

In this work, we study the problem of *auditing* privacy risk in federated learning, from the perspective of a participating party. Membership inference attack (MIA) [35] is a canonical framework for assessing such privacy risks in machine learning and has been adopted by popular machine learning platforms such as TensorFlow [1]. In particular, MIA ascertains whether a target data point was part of the training set, based on some metrics computed on it. Despite its comprehensive literature, existing MIA solutions often fall short of auditing privacy risks in federated learning. In FL, the adversary, who may be the server or a party, participates in the whole FL training process and observes multiple model snapshots during this process. How to extract membership information from the (sequence of) model snapshots effectively and efficiently is a challenging problem that is yet unsolved.

Most MIA solutions extract membership information from an isolated single model snapshot, leading to underestimation of privacy risks in FL (e.g., differentiate a member/non-member target point based on its loss computed on the final model [42]). On the other hand, an adversary in FL can observe multiple model snapshots through which the membership information gradually leaked in the FL process. Some other solutions try to leverage such multi-round information [16, 22]. However, such solutions still fall short in terms of effectiveness–despite that more model snapshots are utilized, they are still processed in an isolated fashion (i.e., the average of some metrics across FL rounds), and the correla-

[1] https://www.tensorflow.org/responsible_ai/privacy/tutorials/privacy_report

tion among these models is not fully exploited. Due to similar reasons, naively adapting centralized attacks to FL also results in poor effectiveness. Finally, training a deep learning attack model to make use of all the observed information in FL such as the per-sample gradient/activation/loss history [31] demands substantial computation resources, potentially impeding FL. Such computation investment may also not be affordable for some parties in the first place. We provide a more comprehensive review of the MIA literature in Section 3 and highlight their limitations of privacy auditing in FL regarding two performance metrics, effectiveness and efficiency.

Free training attack. To address these limitations, we present the *free raining attack* (FTA) for auditing privacy risks in federated learning. Regarding efficiency, our solution does not need to train any additional complicated learning models or perform costly computations. It can be integrated by the parties seamlessly, without disrupting the FL process.

In particular, FTA only takes the performance metrics of the FL model, such as the loss of the local training/validation datasets (which is already computed in the performance monitoring routine), as the inputs and performs elementary math operations on the inputs, incurring very little overhead. This flexibility, in turn, empowers the auditing party to choose her preferred means of privacy protection/intervention at any time to address the privacy risk she is facing. For example, a party with high privacy risk may want to inject DP noises into her local updates; and a party who observes a sudden change in the privacy risk may also choose to adjust the level of DP protection enforced on her local update. This is different from the traditional approach of applying a universal privacy-enhancing protocol to all parties throughout the entire process of FL, without understanding the empirical privacy risks that the parties face.

The effectiveness of our auditing algorithm relies on a simple but effective membership signal, called the *slope signal*. The slope signal exploits the fact that the model performances for members and non-members change at different speeds during the training process–the performance for members increases more rapidly, whereas the performance of non-members increases at slower rates. Our FTA predicts the membership of a target data point based on its slope signal computed across multiple FL model snapshots.

We validate the performance of our FTA on a wide spectrum of settings and real-world datasets. First, we want to highlight that FTA outperforms the efficient baselines, which are also free of training additional models, by a clear margin. In addition, to our surprise, FTA also outperforms more costly baselines that train additional models. For instance, on the benchmark dataset CIFAR100, our FTA successfully identifies 54.4% of members while making only an error of 1%, outperforming all other baselines, which at most identify 15.2% of members under the same error. Supported by its effectiveness and efficiency, our FTA could also be used to benchmark the privacy risks in different FL settings. We will release our code for better reproducibility.

Enhanced auditing. Built upon the slope signal, we further enhance the performance of privacy auditing by integrating it with the strong centralized MIA algorithm which trains additional reference models [5, 41]. The overall idea is to train additional reference models to fit the distributions of the slope signal evaluated on models trained with and without the target data point. If the computed slope signal (regarding the target data point on the target model) looks more like a sample from the distribution of models trained with the target data point, we will predict it as a member, and vice versa. Such integration, however, is not trivial, as the parties in FL often have limited access to the underlying training data distributions, making it difficult to obtain reference models that emulate the target model well. To resolve this challenge, we incorporate knowledge transfer to improve the quality of the reference models and also propose further optimization to reduce computation costs. We call this solution Knowledge Transfer Attack (KTA). Validated on real-world datasets, KTA further improves the effectiveness of FTA, catering to parties who are willing to invest more computation resources for more precise privacy risk assessment. Notably, KTA identifies 62.6% of members while making only an error of 1% on the CIFAR100 dataset, which provides further supporting evidence for the effectiveness of the *slope signal*.

Contributions. We summarize the contribution of this paper as follows. **(i)** We propose a novel membership signal that effectively exploits membership information from the models in multiple FL rounds. **(ii)** Based on this signal, we present an efficient auditing algorithm to quantify the privacy risk for any party. Our algorithm incurs negligible overhead and does not disrupt the FL process. **(iii)** We also provide an enhanced auditing algorithm for parties who are willing to spend more computation resources for better estimation of privacy risks. **(iv)** We conduct comprehensive experiments in various FL settings to validate the performance of our proposed solutions, in terms of effectiveness and efficiency.

2 Preliminaries and Problem Formulation

Federated learning and notations. We consider K parties in total. Each party $k \in [K]$ has access to her local dataset, denoted as S_k with $|S_k| = n_k$ where each data point of S_k is independently sampled from the underlying local data distribution π_k. A T-round federated learning process is summarized as follows. In each round t ($t \in [T]$), the server sends the current global model $\bar{\theta}^t$ to all parties. Next, each party k updates the local model along the direction that minimizes the loss on her local dataset, written as follows.

$$\theta_k^{t+1} = \bar{\theta}^t - \eta \sum_{(x,y)\in S_k} \nabla \ell(\bar{\theta}_t; x, y), \tag{1}$$

where ℓ is some pre-specified loss function and η is the local learning rate. Next, each party sends the local update to the server, which then computes a weighted average of the local updates, and obtains the new global model $\bar{\theta}^{t+1}$ as $\sum_{k\in[K]}\left(\frac{n_k}{N}\cdot\theta_k^{t+1}\right)$, where $N=\sum_{k\in[K]} n_k$.

Membership inference attack. The membership inference attack (MIA) aims to determine if a target point is used for training the target model. In particular, given a target point $z=(x,y)$ and the trained target model, the adversary outputs 0 or 1, predicting z as a non-member or a member, respectively.

In the context of federated learning, an adversary's position can take one of two forms: it can either be the server or a participating party. An adversarial server can observe all local model updates from each party during the FL process as well as the global models, while an adversarial party is restricted to observing only the global model sent from the server. In either case, the adversary always has access to *multiple* model snapshots (local or global) across different FL rounds. We focus on semi-honest adversaries, who conduct passive inference attacks. Namely, the adversary infers membership without actively manipulating or interfering with the FL process.

Problem statement. Our goal is to design an algorithm for each party for auditing privacy risk during FL using MIA, explained as follows. Consider any round t, before sending the updated local model θ_k^{t+1} to the server, the party k may want to know how much privacy information has been leaked to decide if she/he still participates in the training process. To answer that question, the party runs an MIA algorithm and examines its effectiveness (i.e., accuracy) in distinguishing members and non-members with respect to her/his local dataset S_k. Higher accuracy indicates higher privacy risk, and vice versa. The MIA algorithm takes inputs from the global models sent from the server, namely, $\{\bar{\theta}^0,\ldots,\bar{\theta}^t\}$. In addition, when considering an adversarial server, the algorithm also takes inputs from the local updates shared by the party k, namely, $\{\theta_k^0,\ldots,\theta_k^t\}$. Our goal is to provide the party with an *effective and efficient* MIA algorithm so that the parties can obtain accurate privacy assessments on the fly without disrupting FL.

3 Related work

First, we review the existing representative and state-of-the-art MIA algorithms under different settings, including the centralized setting, online setting, and FL setting, followed by discussions on their limitations for auditing privacy in FL.

3.1 MIA in Centralized Setting

Adversaries under the traditional centralized setting attack the final model. When the adversary has *total access* to the parameters of the target model, we refer to this type of attacks as *white-box attacks*. Notably, Nasr et al. [31] computes the gradient norm of the target data point (x,y) and decides it as a member if its norm is less than a certain threshold, and vice versa.

A more realistic threat model (leading to a weaker adversary) is when the adversary can only *query* the target model through APIs. We call such an attack a *black-box attack*. Given target (x,y), *Modified Entropy* [36] queries the confidence for the label predictions on x and computes $-(1-p_y)\log(p_y)-\sum_{y'\neq y}p_{y'}\log(1-p_{y'})$, where p_y is the confidence of the model on predicting y. The adversary decides (x,y) as a member if the score is less than a certain threshold, and vice versa (note that the score is 0 when $p_y=1$ and the score is ∞ when $p_y=0$). *Loss-based attack* [41, 42] queries the loss for the target data point (x,y), written as $\ell(\theta;(x,y))$. The adversary decides (x,y) as a member if the score is less than a certain threshold, and vice versa. The intuition is that, due to overfitting, the losses for training data points are smaller compared to non-training data points. *Merlin* [18] compares the loss on the target point (x,y) with the losses on some other noisy examples with the same label y to get the membership scores. In particular, each noisy example is obtained as x perturbed with some random Gaussian noise. The score is computed as a simulation for $\mathbb{E}_{\xi\sim\mathcal{N}(0,\sigma^2 I)}\left[\mathbb{1}\{\ell(\theta;(x+\xi,y))>\ell(\theta;(x,y))\}\right]$. The adversary decides (x,y) as a member if the score is higher than a certain threshold, and vice versa. The intuition of Merlin is that the loss at the perturbed neighborhood near a member will be higher than that member, while both the loss for a non-member and the loss at the perturbed neighborhood near the non-member are consistently high.

Apart from accessing the target model through APIs, the adversary may also invest additional computation resources to *train reference models*, hoping that the reference models, to which the adversary has full white-box access, can simulate the target model. The main idea is to train two sets of reference models, the IN models, which are trained with the target data point, and the OUT models, which are trained without the target data point. Then, the adversary compares the behaviors of the target model with the IN and OUT models and makes predictions. Next, we review the representative algorithms.

After obtaining the IN and OUT models, *LiRA* [5] computes the rescaled logit of the target data point (x,y) on the IN models and OUT models separately and fits two Gaussian distributions, denoted as $\mathcal{N}_{IN}$ and $\mathcal{N}_{OUT}$. Next, the adversary queries the rescaled logit r computed on target data point (x,y) using the target model and computes $\frac{\Pr[r \text{ is a sample from } \mathcal{N}_{IN}]}{\Pr[r \text{ is a sample from } \mathcal{N}_{OUT}]}$. The adversary decides (x,y) as a member if the score is higher than a certain threshold, and vice versa. *Distillation attack* and *reference attack* [41] only train the OUT models. After obtaining a set of reference models $\{\theta_{sh_1},\ldots,\theta_{sh_M}\}$. The adversary computes the fraction of reference models whose loss for the target data point (x,y) is larger than that computed on the target, written as $\frac{\sum_{m=1}^{M} \mathbb{1}\{\ell\left(\theta_{sh_m};(x,y)\right)>\ell\left(\theta;(x,y)\right)\}}{M}$. The adversary then decides (x,y) as a member if the fraction is higher than a certain threshold, and vice versa. Finally, [21, 25, 35] trains an additional attack model to differentiate the IN/OUT reference models, hoping that it will also perform well in differentiating whether the target data point is used or not for training the given target model.

Adaptation to FL and limitations. To adapt the aforementioned attacks to FL, the adversary runs attacks on the observed model snapshot in each round t independently. Depending on the threat model, the adversary either attacks the global model $\bar{\theta}_t$ or the local model θ_t^k sent from party k. One major limitation of those attacks is their ineffectiveness. In particular, as the adversary only attacks one model snapshot at each round, she/he fails to exploit the membership information that is encoded in the incremental FL process, leading to an underestimation of the privacy risk. Moreover, the more advanced attacks in centralized settings often demand additional computation resources, e.g., for training reference models [5], and computing the per-sample gradient [31]. Such computation overhead burdens participating parties and also significantly slows down the FL process.

3.2 MIA in Online Learning Setting

In the online learning setting, the target model is updated on the arrival of new data points in each round. In particular, at round $t \geq 0$, the updated model is written as $\theta_{t+1} = \theta_t - \eta\nabla\ell(\theta_t; D_{t+1})$, where D_{t+1} represents the new data points arrived at round t. Jagielski et al. [17] consider an adversary who has access to the *model snapshots* (before and after the updates) through APIs and is interested in the membership of the target data point in any dataset of $\{D_1, \ldots, D_{t+1}\}$. In contrast to the centralized setting, this online setting naturally applies to FL, where the adversary also observes multiple model snapshots,

In their proposed *Back-front attack* [17], the adversary computes the difference between the losses for the first model snapshot θ_0 and the final snapshot θ_T, namely, $\ell(\theta_0; (x,y)) - \ell(\theta_T; (x,y))$. Alternatively, the adversary could also compute the loss ratio $\frac{\ell(\theta_0;(x,y))}{\ell(\theta_T;(x,y))}$. The adversary decides (x,y) as a member if the outcome is larger than a prefixed threshold, and vice versa. Intuitively, if (x,y) is used for training, the difference/ratio between the ending and starting losses should be large. *Delta attack* is a more fine-grained version of *Back-front attack*. In particular, the adversary computes the loss difference/ratio between model snapshots in *consecutive rounds*, namely, $\ell(\theta_{t-1}; (x,y)) - \ell(\theta_t; (x,y))$, or $\frac{\ell(\theta_{t-1};(x,y))}{\ell(\theta_t;(x,y))}$. The adversary decides (x,y) as a member if the loss difference/ratio at any two consecutive rounds is larger than the prefixed threshold.

Limitations. Although the above attacks utilize multiple model snapshots, the adversary's decision is ultimately determined using the information obtained from two rounds, rather than the entire FL process. As a result, they still fall short of effectiveness, as we will see in Section 5.

3.3 MIA in Federated Learning

Existing MIA solutions in FL settings leverage all snapshots of the models and we can directly apply them for privacy auditing. Nasr et al. [31] propose to train a *deep learning attack model*, which takes all the information one can compute on all model snapshots to make the prediction, including the concatenation of the per-sample gradient across all FL rounds. As the dimensions of such information change over time, in each round, a new attack model needs to be trained, demanding huge computation resources. Li et al. [22] find that, for overparameterized models, the *gradients* of different data points are nearly orthogonal. Based on this observation, they measure the cosine similarity between the gradient for the target data point and the local model update, written as $\frac{\langle\nabla\ell(\bar{\theta}_{t-1};(x,y)),\theta_t^k-\bar{\theta}_{t-1}\rangle}{\|\nabla\ell(\bar{\theta}_{t-1};(x,y))\|_2\|\theta_t^k-\bar{\theta}_{t-1}\|_2}$, where $\bar{\theta}_{t-1}$ represents the global model sent to the parties and θ_t^k represents the updated local model of party k. The adversary decides the target data point (x,y) as a member if the cosine similarity is larger than a threshold, and vice versa (according to their observation that the gradients of non-members should be orthogonal to those of members). Alternatively, the adversary can also compute $\|\theta_t^k - \bar{\theta}_{t-1}\|_2 - \|\theta_t^k - \bar{\theta}_{t-1} - \nabla\ell(\bar{\theta}_{t-1}; (x,y))\|_2$, and decide the target data point (x,y) as a member if the difference is larger than a certain threshold, and vice versa. The intuition is similar–when (x,y) is a non-member, the local update and the gradient on (x,y) are orthogonal, which gives the largest $\mathcal{L}_2$ norm $\|\theta_t^k - \bar{\theta}_{t-1} - \nabla\ell(\bar{\theta}_{t-1}; (x,y))\|_2$.

Limitations. The main limitation of the above approaches is their computation overhead, as both approaches require computing per-sample gradients, which is far more expensive than local training itself. On top of that, the deep learning-based approach also trains different attack models at each round, further burdening the parties and slowing down FL.

3.4 Other Works

We conclude this section with other directions in privacy-preserving federated learning.

Other privacy risks. Other than membership information, there are also other types of privacy risks in FL [7, 12, 23, 29, 45]. For instance, the adversary may also try to reconstruct the training data [4, 6, 9, 45] or infer the properties (e.g., label distribution [38]) of the training data. Auditing such privacy risks are left as future work directions.

Auditing DP algorithms. Auditing the differentially private (DP) algorithms aims to ensure that the algorithm preserves the privacy levels as claimed [2, 15, 27, 30, 32]. This line of work differs from ours in the sense that DP considers the *worst case* scenario, preventing the adversary from distinguishing any *any* pair of neighboring datasets with high confidence. In contrast, our auditing algorithm focuses on the empirical privacy risk that can be exploited by an adversary who is computationally bounded.

4 Proposed Solution for Privacy Auditing

In this section, we present an effective and efficient MIA solution for auditing privacy, called free training attack (FTA).

4.1 Extracting Membership Efficiently

To avoid excessive overhead, an efficient MIA strategy is to compute some membership signal solely based on the target models and then compare it with some pre-fixed threshold, without computing high-dimensional statistics or training any additional models. The binary outcome of the comparison leads to the membership prediction.

Other than obtaining the model snapshots from different FL rounds, we note that in the standard FL routine, the parties also need to evaluate the model snapshots (namely, the target models from the adversary's perspective) on her/his local datasets, including the training and validation datasets, for monitoring the performance of FL. A (by)product of such evaluation is the calculation of the logit, confidence of labels, and losses of the data points. To be more specific, for each point (x,y) in the local training and validation sets, the party computes the outputs produced by the final network layer before the softmax function, which are referred to as the logit. Rescaled logit is referred to as the logit associated with the predicted label, normalized with respect to the values of other logit. A higher rescaled logit indicates that the model is more confident in the prediction confidence of the input data point. From the logit, the party proceeded to compute the prediction confidence and loss for the data point.

In what follows, we will present a membership signal that effectively extracts membership information from the above-mentioned performance statistics using straightforward arithmetic operations, thereby circumventing excessive computation overhead.

Trajectory of model performance. The foundation of our method hinges on a key observation–the differences between member and non-member data are revealed not only through the model's performance in isolated rounds but also through the performance trajectory leading up to each round. This is illustrated in Figure 1 (left), where we notice a marked difference in local model prediction confidence between members and non-members. This difference is manifested not just in the outcomes of each FL round but also in the *pace at which prediction confidence escalates*. In particular, members show a swift ascent in prediction confidence, whereas non-members (namely, the examples from the validation dataset) show a more stable rise.

Slope of trajectory. Building on this observation, we propose to use the rate of change in the model performance metric (namely, the slope) as the membership signal. In the middle of Figure 1, we plot the slope of average confidence for members and non-members. Just after the 10-th round, we observe a clear separation between them, and the separation remains noticeable in the final round. For a more detailed investigation, we draw the histogram of the slope signals for members and non-members at round 30 in the right of Figure 1. As the non-overlapping region of members and non-members is quite noticeable, the adversary could accurately infer membership based on this slope signal. Moreover, another advantage of such a membership signal lies in its simplicity of computation–computing the slope of a trajectory only involves elementary operations, as we will see later.

Next, we present how to extract the membership signal based on the trajectory of prediction confidence history. Consider any target data point (x,y) associated with the prediction confidence c_t at each round t, the slope up to the $t-$th round is computed as follows.

$$b_t = \sum_{u=1}^{t} w_u \cdot c_u, \tag{2}$$

where w_u is defined

$$w_u := 6 \cdot \frac{2tu - t^2 + 1}{t^4 - t^2}, \tag{3}$$

for u taking values from 1 to t. We refer interested readers to the well-established regression literature [8] for more detailed derivation.

Given our intuition, the value b_t for a member is expected to be larger than that for a non-member. Hence, the adversary decides the target data point as a member if b_t is larger than some threshold, and vice versa. Similar computations also apply to other performance metrics, such as rescaled logit and loss. Here we omit the formulations for brevity and their corresponding experimental results are shown in Section 5.

Summary. Our slope signal achieves effectiveness and efficiency at the same time. In Section 3, we have extensively discussed the limitations of adapting existing membership inference attacks for auditing privacy risks in federated learning from both effective and efficient perspectives. Compared to the centralized MIA solutions, where the membership signal is only computed on an isolated snapshot of the model, our slope signal captures the information that is encoded in the whole trajectory of the training. Compared to MIA solutions designed for online learning settings, our slope, instead of

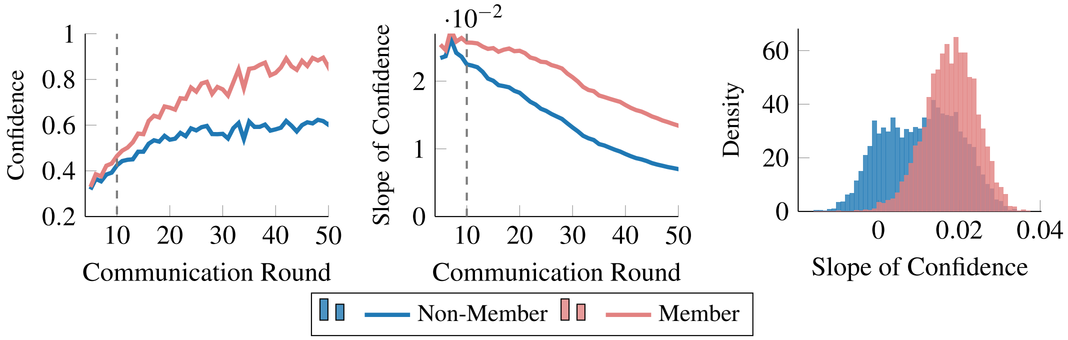

Figure 1: Prediction confidence trajectory of the local models on members and non-members, evaluated on the benchmark dataset CIFAR10. The left and middle figures show the prediction confidence and slope of the confidence for the members and non-members during FL rounds, respectively. The right figure shows the histogram of the slope of confidence for members and non-members at round 30.

looking at the changes between two snapshots of the models, leverages the whole training trajectory for auditing. Compared to the solutions in FL, which require expensive computation, our slope signal can be computed very efficiently, as the parties only need to retrieve the performance metrics for each data point and compute a weighted average of these statistics across FL rounds.

4.2 Free Training Attack

Given the slope b_t computed on the target point, our free training attack strategy is straightforward, that is, comparing this value with a threshold. The output of our attack algorithm FTA, denoted as $\mathcal{A}_{\text{FTA}}$, is computed as follows:

$$\mathcal{A}_{\text{FTA}} := \mathbb{1}\left[b_t \geq \tau\right]. \tag{4}$$

Considering a set of target data points for privacy auditing, we note that for every selected threshold τ, there are two error types regarding the whole set of data. False Positive Rate (FPR) represents the fraction of non-members that are incorrectly identified as members, while False Negative Rate (FNR) represents the fraction of members that are incorrectly identified as non-members. Note that the auditor (namely, the participating party) has access to the ground truth for the membership of the target data points and hence, she/he can obtain the TPR and FPR. In practice, the auditor may be interested in a strict FPR tolerance γ such as 1%. To achieve that, we iterate through a spectrum of τ and pick the τ that results in the highest TPR with FPR below the tolerance γ. The highest TPR, constrained by the acceptable FPR, is reported as the privacy risk.

To avoid iterating infinitely many τ's from an unbounded range, in practice, the auditing party may select different quantiles (e.g., $1\%, 25\%, 50\%$) of some observed empirical distribution of the slope signal, e.g., on her/his local training/validation dataset, and use those quantiles as the candidates for τ. In the experiments, we use a non-overlapping dataset for computing such quantiles. As we will demonstrate, this dataset can be as small as $\frac{1}{50}$ of the training dataset, without affecting the performance of FTA. The above technique for determining the threshold τ for membership prediction is prevalent in the MIA literature [21, 22, 26, 41]. For a fair comparison, we adopt the above selection process of τ for all baselines and our solution in the empirical evaluation.

Adversary's observation. Regarding an adversarial party in FL, she/he could launch FTA based on the observed global model snapshots. On the other hand, regarding an adversarial server, she/he can launch two separate FTAs on the observed global and local model snapshots. To deal with the latter type of adversary, the auditing client may consider the higher attack performance (measured as TPR) as the privacy risk.

5 Empirical Evaluation

In this section, we evaluate the effectiveness and efficiency of FTA compared to existing baselines.

5.1 Baselines

Among many MIA solutions (see Section 3), we include the most representative ones as the competitors for our solution. For the attacks that were originally designed for other settings, we also adapt them to the FL setting. To ensure a fair comparison, we do not tune the hyperparameters in favor of any particular solution.

Efficient MIAs. Efficient attack algorithms that do not demand additional computation resources naturally fit as a good candidate for efficient privacy auditing in FL. We include

Table 1: Setup for experiments on the different datasets. For each setup, we uniformly split the datasets into four clients and use the Adam optimizer to train the local model.

Dataset	CIFAR10	CIFAR100	Skin	Retinal	Texas	Purchase	Medical-MNIST	Pneumonia	Kidney
Model	Resnet56	Resnet56	AlexNet	AlexNet	5-layer-FC-NN	5-layer-FC-NN	LeNet	AlexNet	AlexNet
Rounds	100	200	500	200	500	100	50	200	200
Learning rate	0.001	0.001	0.0001	0.0001	0.0001	0.001	0.0001	0.001	0.001
Number of classes	10	100	23	4	100	100	6	2	4
Dataset size	60,000	60,000	6,095	35,472	67,330	197,324	53,724	3,166	5,508
Train accuracy	1.0	1.0	1.0	1.0	0.99	0.99	1.0	1.0	1.0
Test accuracy	0.72	0.34	0.25	0.82	0.43	0.9	1.0	0.91	0.96

the following ones as baselines in our evaluation: Modified Entropy [36], Population (01-Loss) [41, 42], Back-Front-Diff [16], and Back-Front-Ratio [16]. We also include the Fed-Loss that is also used in prior work [22]. Fed-Loss computes the average loss of the target data point across all FL rounds and predicts it as a member if the outcome is lower than a threshold.

Inefficient MIAs. We also include the algorithms which demand additional computation resources, including Merlin [18], Gradient Norm [31], Gradient-Cosine [22], and Gradient-Diff [22]. For Merlin, which evaluates the model performance on noisy perturbations of the input target data point, we follow the same hyperparameters as in the original implementation [18], which generates 100 noisy samples with $\sigma = 0.01$. For the Gradient Norm attack, we compute the norm in the last layer, as suggested in the original paper [31] since including more layers does not improve the performance of the attack. Gradient-Cosine and Gradient-Diff attacks [22] (recall Section 3) attack on a single layer only. To select the best layer to attack, they use a non-member dataset to find the layer that gives the smallest variance (across all data points) for the designated membership signal. In addition, for each target data point, they average the signals computed across different rounds and use that for predicting its membership. In our evaluation, we follow the same procedure.

Finally, as we have mentioned in Section 4.2, for all baselines, we also compute the membership signal on a non-member dataset to calibrate the threshold τ.

5.2 Setup

Dataset and model. Following prior works [5, 31, 41], we primarily focus on evaluating the two benchmark datasets CIFAR10 and CIFAR100 [20] and train a resnet56 model [10]. Besides, we also evaluate additional datasets in our extended experiments, including the Purchase dataset from Kaggle, the Texas hospital discharge dataset, the medical MNIST [3], the Pneumonia dataset, and the Retinal OCT image dataset [19], the CT Kidney dataset [13], and the skin disease dataset (http://www.dermnet.com/).

By default, we consider homogeneous data partitioning (in short, IID setting). For each dataset, we uniformly split it among 4 parties (we also provide ablation studies later). After data partitioning, each party takes 30% of their sampled data for training (referred to as members in privacy auditing) and another 30% for validation (referred to as non-members in privacy auditing). For the relatively small Pneumonia dataset, each party takes 40% of their sampled data for training and another 40% for validation. We train the models until convergence using FedAvg, the standard training algorithm used in FL [28], where each party trains the local model for one epoch and then sends the updated local model to the server. The detailed description of the model structures and hyperparameters is in Table 1.

Variants for FTA. For our FTA, we evaluate three of its variants on different model performance metrics, including the slope computed on loss, prediction confidence (in short, confidence), and rescaled logit (in short, logit). For loss, the members are expected to have a smaller slope than the non-members as the loss for members often decreases more rapidly compared with non-members. Accordingly, the direction of the inequality sign in Eq. (4) is flipped. We would also like to emphasize that due to the efficiency of our FTA, the auditing party can choose to assess the privacy risk using *all three variants* and use the *best performance* among all three variants as the estimation for the privacy risk.

Metrics. We are interested in two metrics, effectiveness and efficiency. The efficiency is measured by the time spent on the auditing in each round over the time spent on the local training in each round for one epoch. We measure the efficiency as the *GPU time* required for conducting the MIA/privacy auditing algorithm in each round. To ensure a fair comparison, the GPU time is computed with the same machine configuration with 1 NVIDIA Titan RTX GPU. On the other hand, the effectiveness is measured by the attack success rate. Following prior work [5, 41], we use the TPR at low a FPR as the measurement for effectiveness. A higher TPR at a low FPR indicates a better estimation of the privacy risks. All results are averaged across all parties over 5 independent runs. For clearer demonstration, in our experiments, we report the ef-

Table 2: Effectiveness and efficiency of auditing the privacy risk for local models trained on CIFAR10 and CIFAR100 (abbreviated as C-10 and C-100, respectively). We focus on the local models with respect to the whole FL process and use benchmark datasets CIFAR10 and CIFAR100 (abbreviated as C-10 and C-100, respectively) with IID data partitioning. Efficiency is measured as the ratio between the GPU time used for auditing versus training. For each solution, we mark whether it uses multiple model snapshots by 'Final' or 'Multiple' in the 'Snapshots' column.

Algorithm	Snapshots	Efficiency	AUC		TPR @0.1% FPR		TPR @0.5% FPR		TPR @1% FPR	
		(C-10)	C-10	C-100	C-10	C-100	C-10	C-100	C-10	C-100
Modified Entropy [36]	Final	0.56	0.691	0.902	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
01-Loss [41]	Final	0.56	0.687	0.903	0.0%	0.4%	0.0%	2.4%	0.0%	5.1%
Fed-Loss [22]	Multiple	0.56	0.712	0.901	0.1%	0.0%	0.5%	0.6%	0.9%	1.6%
Back-Front-Diff [17]	Multiple	0.56	0.66	0.879	0.3%	0.8%	0.7%	2.6%	1.2%	4.8%
Back-Front-Ratio [17]	Multiple	0.56	0.689	0.903	0.0%	0.4%	0.0%	2.6%	0.0%	5.3%
Delta-Diff [17]	Multiple	0.56	0.324	0.2	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Delta-Ratio [17]	Multiple	0.56	0.697	0.858	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Merlin [18]	Final	57.3	0.508	0.581	0.0%	0.3%	0.4%	0.7%	1.1%	2.0%
Gradient Norm [31]	Final	373	0.684	0.893	0.2%	0.3%	0.7%	2.3%	1.2%	5.2%
Gradient-Cosine [22]	Multiple	373	0.359	0.368	0.0%	0.0%	0.2%	0.0%	0.3%	0.2%
Gradient-Diff [22]	Multiple	373	0.803	0.947	0.4%	0.1%	1.4%	3.4%	2.8%	6.6%
FTA (loss)	Multiple	0.56	**0.817**	0.95	**3.1%**	0.0%	5.9%	15.6%	9.0%	26.1%
FTA (confidence)	Multiple	0.56	0.801	**0.975**	3.0%	**12.8%**	**9.2%**	**43.2%**	**12.9%**	**54.4%**
FTA (logit)	Multiple	0.56	0.783	0.971	0.3%	0.1%	0.4%	2.0%	0.8%	11.2%

fectiveness results on the local and global models in separate figures and tables.

5.3 Main Results

We first compare our auditing algorithm with baselines based on privacy auditing for the local models (considering the adversarial server setting) with IID data partitioning on the benchmark datasets CIFAR10 and CIFAR100. We focus on the privacy risks with respect to the whole FL process (i.e., up to the final round). We compare the efficiency and effectiveness of different solutions in Table 2. We show the true positive rate (TPR) at low false positive rates (FPR) (i.e., 0.1%, 0.5%, and 1%) and the area under the ROC curve (AUC). The best performance under each metric is bolded. We also measure the efficiency of auditing on CIFAR10.

As a reference, the GPU time for updating the local model for one round is around **2.3 seconds** (on average). For efficient MIA solutions, including our FTA and some baselines [16, 36, 41, 42], the additional cost is evaluating the model performance on the non-member dataset, which only takes around **1.2 seconds**. For Merlin [18], the additional cost is evaluating the model performance on 100 noisy versions of each point in both target and non-member datasets, which cost around **131.8 seconds**. For gradient-based approaches [22], the cost is computing the per-sample gradients for all samples in target datasets and non-member datasets, which cost around **860 seconds**. This is 373 times larger than the time spent on local training! Consider the case where FL takes 3 GPU hours with 4 parties. Using the auditing algorithm based on gradient information will slow down FL to more than 46 GPU days when simulating FL and auditing the privacy risk for each party, which may prevent researchers from understanding the risks in FL.

Regarding effectiveness, our FTA surpasses all the baselines by a significant margin. Notably, FTA achieves a TPR of 43.2% on CIFAR100 when FPR is 0.5%, which is more than 10 times higher than the best performance achieved by the gradient-based baseline [22]. On CIFAR10, our FTA achieves a TPR of 9.22% when FPR is 0.5%, which is 7 times larger than the best performance [22]. As claimed in the original paper [22], the key observation of the gradient orthogonality is highly dependent on the model structure. Furthermore, only after enough training epochs will the overparameterized model gradients of different data points be nearly orthogonal. Such limitations may contribute to the performance gap between our FTA and [22]. In addition, [22] requires more than 700 times more computation resources than our FTA.

Compared with efficient baselines, our improvement in effectiveness is even more significant. While FTA achieves a TPR of 43.2% on CIFAR100 when FPR is 0.5%, the most efficient baseline [17] only achieves a TPR of 0.8%. Although multiple model snapshots are utilized in [17], they are still processed in an isolated manner. In contrast, our FTA exploits the membership information from the entire FL trajectory, leading to much higher effectiveness. In conclusion, FTA not only stands out as the most effective approach in our experiments, but it also maintains this superior performance without imposing any significant computational burden, establishing itself as a practical auditing algorithm accessible to any party

Table 3: Effectiveness of auditing global models on CIFAR10 and CIFAR100 with IID data partitioning.

	TPR@0.1%FPR		TPR@0.5%FPR	
Algorithm	CIFAR10	CIFAR100	CIFAR10	CIFAR100
Population	0.00%	0.00%	0.00%	1.76%
Fed-Loss	0.11%	0.11%	0.64%	0.69%
Back-Front-Diff	0.36%	0.71%	0.67%	3.09%
Gradient-Cosine	0.27%	0.07%	1.40%	2.78%
FTA (confidence)	**2.58%**	**15.22%**	**8.44%**	**27.16%**
FTA (loss)	1.00%	2.42%	3.31%	10.60%
FTA (logit)	0.04%	0.22%	0.31%	4.31%

Table 4: Effectiveness of FTA (confidence) for auditing global models under different sizes of non-member set for selecting the threshold τ. We measure TPR when FPR is fixed at 0.5%.

Non-member	Size of Dataset						
Dataset	100	500	1000	2000	3000	4000	5000
CIFAR10	0.5%	5.5%	6.1%	6.3%	6.3%	6.5%	6.5%
SVHN	0.0%	0.0%	0.0%	1.8%	3.2%	3.8%	3.9%

in FL.

Auditing global models. We next present the effectiveness results for auditing the privacy risk of the global models Table 3. For each solution, as the computation for auditing privacy is necessarily applied to the global model instead of the local model, the efficiency remains the same, and hence, is omitted from the table. We also omit the baselines, which give 0 for all settings. As we can see from Table 3, our FTA still outperforms the baselines, by a large margin. In addition, compared with the results for auditing local models in Table 2, our FTA can be more effective for auditing global models. We suspect that this is because, for global models, the performance trajectory is more stable, leading to a more stable slope signal for inferring membership. We support this hypothesis with Figure 2, where we plot the average prediction confidence of the global model and the local models on members and non-members. As we can see, the confidence for the global model is more stable, whereas there are more fluctuations for the local model. Finally, as we have discussed in Section 4.2, when auditing the risks with respect to a malicious server, the actual privacy risk should be computed as the *higher* one between the local and global model snapshots, as the adversary can observe both during FL process.

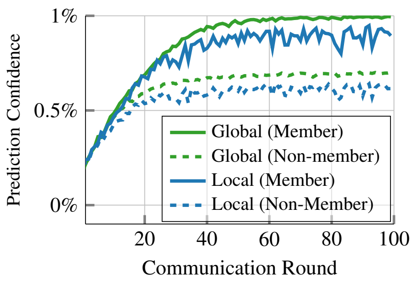

Figure 2: Prediction confidence of local and global models on CIFAR10 with IID data partitioning.

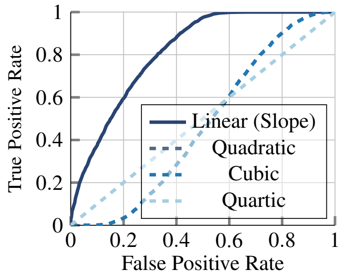

Figure 3: Comparison of fitting different polynomial functions for privacy auditing.

5.4 Ablation Studies and Extended Results

Effect of the validation data. Recall from Section 4.2 that, our FTA uses a non-member dataset for selecting the threshold τ. We first show how the size of this dataset influences the performance of our FTA (confidence) in Table 4. Remarkably, only 500 data points of CIFAR10 are enough for effective privacy auditing, achieving a TPR of 5.9% with an FPR of 0.5%, for the CIFAR10 dataset. As a reference, the best TPR of the baselines is only 1.4%. This suggests that parties need not reserve large validation datasets for auditing, further underscoring the practicality of the FTA approach. Instead of taking validation data from the target dataset CIFAR10, the auditor party may also use data from external sources, such as SVHN [33], to audit the privacy risk regarding CIFAR10. In particular, FTA produces a TPR of 3.9% with an FPR of 0.5%. This observation also suggests the high applicability of FTA even when access to validation data is limited.

Effect of linear fitting. In our approach, we have modeled the performance history as a linear function with respect to the FL round t. However, one might consider fitting other types of functions to the performance history, such as the quadratic function represented by $\ell_t = a + b \cdot t + c \cdot t^2$. To explore such kinds of alternatives, we apply more complex functions from the polynomial function family, including the quadratic (second-order function of the FL round t), cubic (third-order), and quartic functions (fourth-order), on the loss history. The results are shown in Figure 3. Interestingly, the application of more complex functions, as opposed to linear ones, appears to reduce the effectiveness of the attacks (the curves of quadratic and cubic functions overlap). This finding

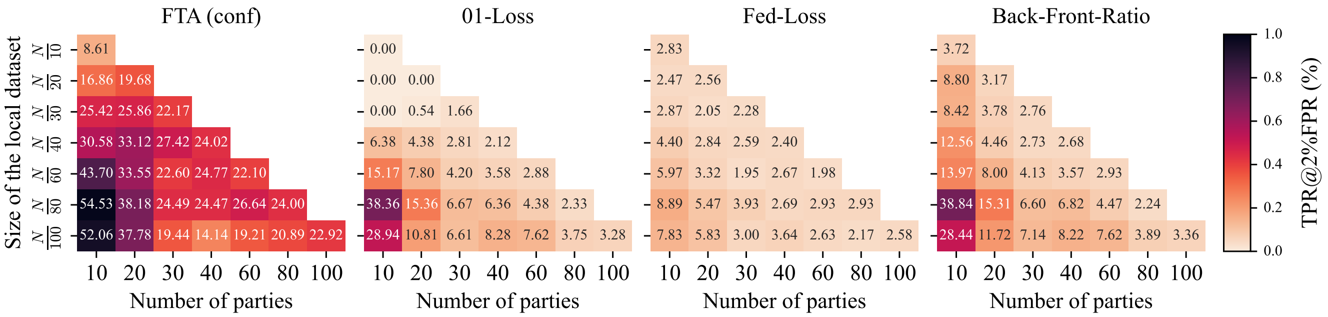

Figure 4: Effectiveness of auditing global models with varying numbers of parties and sizes of local datasets.

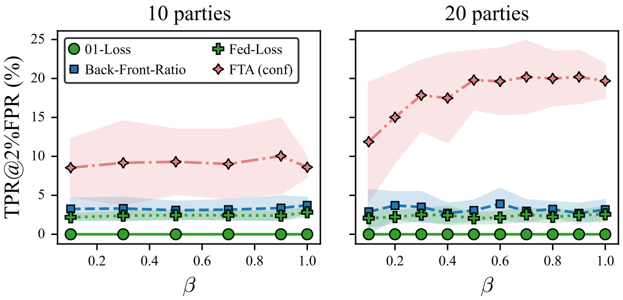

Figure 5: Effectiveness of auditing global models on CIFAR10 with varying data discrepancies. The error bar is computed on the standard deviation across all parties.

is remotely related to the Neural Tangent Kernel theory, highlighted by Jacot et al. [14], which posits that model outputs may evolve linearly during training.

Equipped with the powerful and efficient auditing tool of FTA, we further evaluate the privacy risks in FL under different settings. The following results are based on auditing the global models.

Results on different data partitioning settings. We vary the size of the local datasets in each party from $\frac{N}{10}$ to $\frac{N}{100}$ (N is the overall number of training data points) and the overall number of participating parties from 10 to 100 for the CIFAR10 dataset with IID data partitioning. Overall, there are 28 different settings. We include results of the Back-Front-Ratio baseline, which achieves the best performance compared to the Back-Front-Diff, Delta-Diff, and Delta-ratio baselines. We have omitted the expensive baselines due to their timely costs. We present the resulting TPR at FPR 2% for different solutions for auditing the global models (as there are fewer data points, we increase the FPR tolerance) in Figure 4.

First, we note that our method outperforms the baselines *in all settings*, as indicated by the colors. Next, we fix the number of parties while reducing the size of local datasets (inspect each column separately), and observe an increase in privacy risks. This is because when the training dataset size is smaller, each data point has a larger impact on the model, leading to a higher privacy risk. In addition, we fix the size for each party while increasing the number of parties in FL (inspect each row separately), and observe a decrease in the privacy risks. For instance, when each party has $\frac{1}{100}$ of the whole training dataset, increasing the number of parties from 10 to 100 reduces the TPR of FTA (confidence) by almost a half (from 52.1% to 22.9%). Both results highlight the benefit of having more parties/data points in FL to reduce privacy risks.

Discrepancy in local datasets. We evaluate how the data discrepancies in the parties' local datasets may affect the privacy risk. Specifically, we consider the common heterogeneous setting (in short, non-IID setting). In particular, for each label $y \in \mathcal{Y}$, the distribution for the number of local samples of label y among parties is controlled by the Dirichlet distribution, as is done in prior works [24, 39, 43]. We use $\mathrm{Dir}_K(\beta)$ to represent a K-dimensional Dirichlet distribution of parameter β. Here, K represents the number of parties and β controls the degree of heterogeneity. For every class y, we first sample as $p_y \sim \mathrm{Dir}_K(\beta)$. Next, each dimension in the vector p_y is used to allocate a proportion $p_{y,k}$ of instances that belong to class y, to party k. This partitioning strategy was initially introduced by [43] and has been subsequently employed in several recent FL studies, e.g., [24, 39].

We fix the whole training dataset of CIFAR10 size and set the number of parties to 10 and 20. In both setups, we vary the discrepancy parameter β from 0.1 to 1.0. A smaller β indicates higher data discrepancy in class labels among the parties' partitions for CIFAR 10, and vice versa. When $\beta = 1.0$, it is the same as the setting of IID data partitioning.

In Figure 5, we report the performance (measured as TPR when FPR is fixed at 2%) using our FTA (confidence) as an illustration. The first observation is that the variance of privacy risks for parties is larger in the non-IID partitioning ($\beta < 1$) than in the IID setting ($\beta = 1$). This indicates that in FL with heterogeneous local data, parties are experiencing different levels of risk, especially in the Non-IID setting. The average privacy risk also reduces when increasing the discrepancies

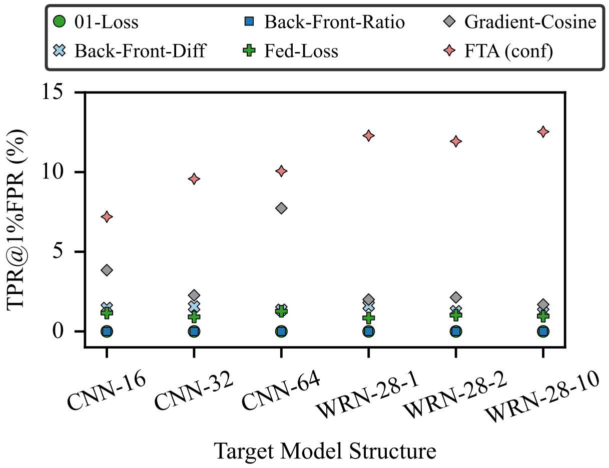

Figure 6: Effectiveness of auditing global models with different model structures.

in the data between the parties. Such an effect cannot be observed using the existing baselines. These findings further highlight the importance of having an efficient and effective auditing algorithm that is available for all parties.

Varying model structures. We include the results for different models on CIFAR10 with IID data partitioning for 4 parties. Specifically, we consider three CNN models (with 16, 32, and 64 convolutional filters), and three Wide ResNets (WRN) with widths 1, 2, and 10. The results are shown in Figure 6. Under all setups, our FTA (confidence) consistently achieves the highest performance, highlighting the wide applicability of the *slope signal* regardless of the model structure. In addition, with more complicated models, the privacy risks increase. This is aligned with the observations in [5].

Table 5: Effectiveness of auditing global models in FedSGD.

	AUC	TPR@Low FPRs		
		FPR=0.1%	FPR=0.5%	FPR=1%
Modified-Entropy	0.77	0.00%	0.00%	0.00%
01-Loss	0.77	0.07%	0.71%	1.42%
Back-Front-Diff	0.69	0.11%	0.38%	0.91%
Back-Front-Ratio	0.77	0.07%	0.78%	1.29%
Fed-Loss	0.62	0.00%	0.62%	1.00%
Merlin	0.64	0.00%	0.00%	0.00%
FTA (conf)	**0.79**	0.56%	1.91%	3.53%
FTA (loss)	0.76	**0.67**%	**2.69**%	**5.33**%
FTA (logit)	0.75	0.00%	0.47%	0.84%

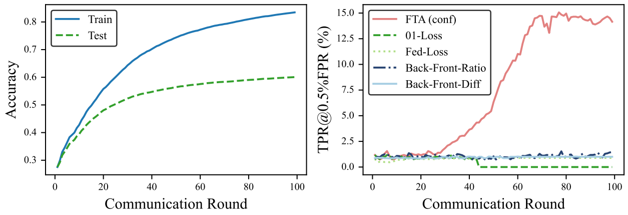

Figure 7: Classification accuracy of the global model and effectiveness of auditing global models during training on CIFAR10 with IID partitioning.

Auditing FedSGD. So far, our focus is on the FL algorithm of FedAvg. Next, we audit the privacy risk in FedSGD. We implement the FedSGD based on canonical settings, where each party computes the gradient on her/his local dataset and shares the gradient with the server in each communication round. We train the model on CIFAR10 that is uniformly partitioned among 4 parties for 2000 communication rounds. Regarding the evaluation of attack algorithms, we have omitted the gradient-based approaches in [22], which compute individual gradients and cost almost 20 GPU days in total (as a comparison, our FTA only costs 0.66 GPU hours in total). The results for auditing the global model are shown in Table 5. We have neglected the baselines that perform close to random guessing. Overall, our FTA (confidence) consistently achieves much better performance under all settings. In addition, we note that the privacy risk for FedSGD is low. We suspect this may be contributed by the fluctuations in training (as demonstrated in Figure 8 in Appendix). In particular, for members and non-members, the model performs similarly at each FL round. However, we note that FedSGD does not obtain a model with high accuracy. Even with 2000 rounds, the test accuracy for the model is 0.536. Due to the unsatisfactory model performance and large communication costs, we suspect that FedSGD is not an interesting subject to study for privacy-preserving FL.

Different datasets. We show the performance of our FTA (confidence) and different baselines for auditing the global models on the nine datasets introduced in Section 5.2 in Table 6. Overall, the results show that our FTA consistently archives the best performance on *almost all datasets*, except for one trivial case where the inherent privacy risk is low. To be more specific, on the relatively easy dataset Medical-MNIST, the highest AUC is 0.51, which is only slightly better than random guessing (corresponding to an AUC of 0.5); and the highest TPR is 2.11% which is also only slightly better than the random baseline of 2%. Despite such a trivial case, the results demonstrate the universal effectiveness of our proposed FTA on different datasets and model structures. Such a difference in the privacy risks also highlights the importance of having an efficient auditing tool that is applicable to all kinds of datasets and model structures.

Privacy risks dynamics. Finally, we fix the FPR to 0.5% and see how the TPR varies as the training proceeds in FL.

Table 6: Effectiveness of auditing the global models on different datasets. We report the AUC and the TPR when FPR is fixed at 2%. When FPR is fixed at 2%, the random-guessing baseline for TPR is 2%.

	CIFAR10		CIFAR100		Skin		Retinal		Texas		Purchase		Medical-MNIST		Pneumonia		Kidney	
Algorithm	AUC	TPR	AUC	TPR	AUC	TPR	AUC	TPR	AUC	TPR	AUC	TPR	AUC	TPR	AUC	TPR	AUC	TPR
Modified-Entropy	0.68	0.96%	0.93	0.00%	0.91	0.00%	0.65	0.00%	0.80	0.00%	0.54	0.00%	0.51	0.00%	0.55	0.00%	0.57	0.00%
Merlin	0.58	0.00%	0.72	0.00%	0.79	0.00%	0.57	0.00%	0.70	0.00%	0.51	0.00%	0.50	0.00%	0.54	0.00%	0.52	0.00%
01-Loss	0.67	1.25%	0.93	15.27%	0.90	8.75%	0.64	0.00%	0.80	1.80%	0.54	0.00%	0.51	0.00%	0.54	0.00%	0.57	0.00%
Fed-Loss	0.60	2.44%	0.85	2.87%	0.84	3.06%	0.58	2.18%	0.73	2.18%	0.53	2.12%	0.50	1.74%	0.53	1.58%	0.57	2.66%
Back-Front-Diff	0.61	2.67%	0.86	7.47%	0.89	6.56%	0.64	3.31%	0.84	8.40%	0.55	2.68%	0.51	2.08%	0.52	4.11%	0.54	1.69%
Back-Front-Ratio	0.67	1.39%	0.93	15.44%	0.90	8.75%	0.64	0.00%	0.80	2.02%	0.54	0.00%	0.51	0.00%	0.54	0.00%	0.57	0.00%
Delta-Diff	0.42	0.19%	0.26	0.00%	0.39	0.00%	0.43	0.00%	0.38	0.00%	0.47	0.72%	0.50	1.69%	0.48	0.00%	0.43	0.00%
Delta-Ratio	0.59	1.37%	0.81	6.64%	0.74	0.00%	0.54	0.00%	0.69	0.00%	0.52	1.74%	0.51	1.34%	0.50	0.00%	0.47	0.00%
Gradient-Cosine	0.59	3.80%	0.79	12.67%	0.67	8.10%	0.55	3.57%	0.78	31.27%	0.52	2.47%	0.51	2.06%	0.49	0.00%	0.49	2.91%
Gradient-Diff	0.38	0.37%	0.09	0.00%	0.14	0.00%	0.41	0.00%	0.23	0.00%	0.47	0.05%	0.50	**2.11**%	0.43	0.00%	0.43	0.00%
FTA (confidence)	0.69	**9.82**%	0.95	**49.62**%	0.90	**19.04**%	0.65	**6.39**%	**0.90**	**34.11**%	**0.58**	2.82%	0.50	1.86%	0.57	**4.75**%	0.57	2.42%
FTA (loss)	0.67	6.33%	0.91	28.31%	**0.90**	11.16%	0.64	5.41%	0.88	26.04%	0.56	**2.99**%	0.50	1.89%	**0.57**	3.48%	0.56	2.91%
FTA (logit)	**0.69**	2.16%	**0.97**	44.58%	0.89	9.63%	**0.65**	2.14%	0.87	5.41%	0.54	1.84%	**0.51**	1.59%	0.54	2.22%	**0.58**	**2.91**%

We focus on CIFAR10 with IID partitioning. From Figure 7, we see that FTA outperforms other baselines. Comparing the model performance (train and test accuracy) in Figure 7 (left) with the exhibited privacy risk in Figure 7 (right), we find that while the train and test accuracies of the global model stop rising at some point, the attack performance of our FTA (confidence) continues to increase. On the contrary, the TPR of other baselines barely changes over time. This further provides evidence that the proposed *slope signal* is a robust membership indicator throughout the FL process and also highlights its applicability, in particular, for discovering the privacy risks in late training phases.

6 Enhanced Auditing

Certain parties in FL may invest more computation resources in exchange for a more accurate assessment of the privacy risk. To that end, we propose a more advanced solution that is built upon the *slope signal*, called Knowledge Transfer Attack (KTA), which combines the existing state-of-the-art centralized MIA algorithm [5] and our *slope signal*. We show that KTA further improves the effectiveness of FTA, validating the effectiveness and broad applicability of the *slope signal*.

6.1 Algorithm

Adapting slope signal to reference model attack. The original idea of [5] is to train reference models that simulate the behavior of the target model. In particular, they first obtain a set of reference models, referred to as the IN models, using datasets with the target data point; and then train another set of reference models, referred to as the OUT models, using datasets without the target data point. Next, the most straightforward way to decide if the target model was trained using the target data point is to look at the distributions of model parameters of the IN and OUT models, and then determine which distribution the target model's parameters are sampled from.

Indeed, estimating the distribution for the whole model parameters is intractable in practice. Thus, existing solution [5] estimates the distributions of the rescaled logit in the IN and OUT models, referred to as IN and OUT distributions (for rescaled logit). The adversary can query the rescaled logit on the target model and determine whether it is a sample of the IN or OUT distribution.

Now we adapt our proposed *slope signal* to this process for auditing privacy in FL. Without loss of generality, we focus on the performance metric of loss. In particular, the adversary fits two Gaussian distributions to the empirical slope of loss distributions for the IN and OUT reference models, and then determines which one is more likely–the slope computed for the target data point on the target model (referred to as the target slope) is sampled from the IN distribution or the OUT distribution. As we will explain next, such an adaptation is highly non-trivial in FL settings.

Improved training. In practical FL settings, the auditor can only train reference models using her/his own dataset, whereas an adversary may also have access to other datasets, leading to difficulties in obtaining IN and OUT reference models with fidelity. Consider an extreme case: party 1, tasked with auditing, possesses only truck images from CIFAR10, while other parties hold images of dogs. Accurately estimating the true slope distributions demands data access to both truck and dog images whereas the auditing party, could only obtain models relying on the truck image.

In what follows, we overcome this challenge by adding a regularization that *forces* the local reference models trained on truck images only obtained by the auditing party to be similar to the models trained on both images. In each round t, given the observed target model θ^t, the auditor trains the reference model θ_s to optimize the following objective.

$$\min_{\theta_s}(1-\alpha)L(\theta_s;S_{sh})+\alpha R(\theta_s,\theta^t), \qquad (5)$$

where the S_{sh} represents the dataset used for training the local reference model (termed as reference data) and R is a regularizer that measures the closeness between the reference model θ_s and the target model θ^t. The overall objective is to minimize the reference model's loss on the reference data and the distance between the reference model and the target model at the same time. Parameter α controls the relative level of importance of the two minimization objectives. We study the effect of α is studied in our evaluation.

Choice of the regularizer. The regularizer R serves the objective of aligning the reference model closely with the target model based on the model outputs, which is a critical aspect of knowledge transfer (KT). The idea of KT is to transfer knowledge from a larger model to a smaller one, even in the absence of shared training data. Among the literature, the Knowledge Distillation (KD) regularizer [11], which employs the Kullback-Leibler divergence [37], to measure the output distribution discrepancies between the two models, is particularly effective. We adopt this measurement as our choice of R in Eq. (5).

Optimizing training efficiency. To save the computation time for obtaining reference models, in each round t, the auditor fine-tunes the reference models from the previous round, instead of training from scratch. Accordingly, the reference models are updated as follows.

$$\theta_s^{t+1} = \theta_s^t - \eta\Big((1-\alpha)\nabla L(\theta_s^t; S_{sh}) + \alpha\nabla R(\theta_s^t, \theta^t)\Big), \quad (6)$$

where η is some pre-fixed learning rate, and θ_s^t and θ_s^{t+1} represent the old and newly-updated reference model at round t, respectively.

Knowledge transfer attack. We put everything together and present the complete attack procedure called Knowledge Transfer Reference Attack (KTA). Specifically, the party samples multiple reference datasets such that each point in the target dataset (union of local training and validation datasets) is included in exactly half of the reference datasets. Then the party trains one reference model for each reference dataset. This ensures that the target point is used by exactly half of the models (i.e., IN models) and not used by the rest (i.e., OUT models). The party computes the slope for the target point for all reference models. Based on the computed slope from the IN and OUT models, the adversary fits two Gaussian distributions, referred to as the IN Gaussian and OUT Gaussian distributions. As shown in Figure 1, the empirical distributions of the slope indeed look like Gaussian distributions. The idea of fitting the empirical distribution to a Gaussian was also used in [5]. Lastly, given the estimated slope distributions, we compute the likelihood that the target slope is a sample from the IN Gaussian and the OUT Gaussian and make the corresponding membership prediction.

Comparison with [41] and [5]. Knowledge distillation is also employed by Ye et al. [41] and Liu et al. [25] for MIA in the centralized setting with a single target model (the final model). The main difference is that our KTA extracts membership information from multiple model snapshots. To further optimize efficiency, we repeatedly fine-tune reference models from previous rounds, instead of training models from scratch. Due to the high computational demands of Liu et al.'s method in auditing privacy risks in FL (recall Section 3), we exclude this baseline from our experiments. The original approach in [5] does not involve the regularizer. Namely, the only objective is to minimize the loss of the reference model on the reference dataset. As we will see next, such a design is unfavorable under the FL setting, compared with training reference models with regularization.

Extend to advanced centralized MIAs When adopting the reference model based attack in centralized setting into FL, we propose to use slope signal and apply the signal and applying knowledge distillation regulation for training the reference models. Those adaption is compatible with more advanced attacks in centralized setting in the future. For instance, the concurrent work [44] propose a new test to improve the robustness and efficiency of privacy auditing in the centralized setting, which achieves the latest state-of-the-art performance. Such attack can be easily integrated into FL based on the two modifications we suggested in the paper.

6.2 Empirical Results

Setup. The evaluation setup is the same as in Section 5. We empirically compare KTA with existing baselines with the existing state-of-the-art attacks [5, 41] in the centralized setting, both of which train reference models. For the baselines and our KTA, we train 16 reference models in total using the same optimizer and model structure as in Section 5. In practice, if parties possess greater computational resources, they can enhance their estimation of privacy risks by increasing the number of reference models, as demonstrated by the findings in [5]. By default, we set α as 0.8, which consistently performs better than other values across different settings.

Results. Table 7 summarizes the effectiveness results of our algorithms and the baselines on auditing local models trained on CIFAR10 and CIFAR100 under IID data partitioning, considering the whole FL process. We also include the best performance of FTA (among loss, confidence, and rescaled logit) for comparison.

Overall, our KTA outperforms the baselines [5, 41] under all settings. Among the three model performance metrics, confidence and rescaled logit lead to the highest performance for KTA. For example, on CIFAR100, KTA (rescaled logit) achieves a TPR of 32.4% with an FPR of only 0.1%, which is ten times larger than that of LiRA. The same conclusion applies to the same datasets with non-IID partitioning, as

Table 7: Effectiveness of auditing local models with IID data partitioning. We show the TPR at low FPR rates, 0.1%, 0.5%, and 1%, as well as the AUC (area under the ROC curve). The highest performance under each setting is bolded.

Algorithm	AUC		TPR @0.1% FPR		TPR @0.5% FPR		TPR @1% FPR	
	CIFAR10	CIFAR100	CIFAR10	CIFAR100	CIFAR10	CIFAR100	CIFAR10	CIFAR100
FTA	**0.817**	0.975	5.0%	12.8%	9.2%	43.2%	12.9%	54.4%
Distillation [41]	0.608	0.583	1.0%	0.1%	2.1%	1.1%	3.7%	2.1%
Lira [5]	0.721	0.882	2.2%	3.1%	6.1%	6.9%	8.3%	15.2%
KTA (loss)	0.788	0.943	4.7%	7.0%	8.4%	19.1%	11.8%	29.4%
KTA (confidence)	0.796	0.975	**5.0%**	18.6%	**11.3%**	46.6%	**15.5%**	56.4%
KTA (logit)	0.682	**0.983**	3.0%	**32.4%**	9.6%	**53.5%**	12.5%	**62.6%**

shown in Tables 9 in Appendix B.

Both LiRA and distillation attacks require training reference models, which cost about 76 GPU seconds to fine-tune reference models in each round. Note that we have applied the fine-tuning strategy for obtaining the reference models in each round (without it, the overall cost would be 76t GPU seconds at the t-th round). Our KTA costs about 94 GPU seconds to fine-tune reference models in each round. The additional 20 seconds is for calibrating our model to the regularizer. As a comparison, our FTA, which does not train any reference model, only costs around **1.2** GPU seconds. Despite such expenses, these baselines still perform worse than FTA. As we have mentioned, this is, again, attributed to the fact that our *slope signal* effectively exploits the membership information across FL rounds, whereas the compared baselines, which attack only one model snapshot, fail to extract such information.

Table 8: Effectiveness of KTA under different parameter α. We show the TPR when FPR is fixed to 0.5%.

Dataset	Partition	α					
		0	0.2	0.4	0.6	0.8	1.0
CIFAR10	IID	6.6%	6.8%	7.3%	9.4%	11.7%	3.1%
	Dir(0.5)	3.0%	5.3%	4.1%	4.7%	5.8%	1.1%
	Dir(0.1)	1.8%	2.1%	2.8%	2.7%	3.7%	0.9%
CIFAR100	IID	41.5%	46.8%	50.0%	49.6%	46.1%	25.4%
	Dir(0.5)	51.6%	51.7%	51.8%	51.1%	51.4%	26.3%
	Dir(0.1)	35.7%	43.0%	41.8%	37.8%	35.4%	19.6%

Effect of parameter α. We study the effect of α, the parameter that controls the level of regularization in KTA. Recall Eq. (5), larger α means that we want the reference model to behave more similarly to the target model rather than to achieve minimal loss on the reference training dataset. The effectiveness results under different choices of α are shown in Table 8.

We first examine two extreme cases, $\alpha = 0$ and $\alpha = 1$. When $\alpha = 0$, a reference model is trained to minimize the loss on the reference dataset only (similar to LiRA [5]); and when $\alpha = 0$, the models are trained to minimize the distance between the reference model and the target model only (similar to [41]). We note that choosing $\alpha \in (0, 1)$ leads to better effectiveness. For instance, while the straightforward adaptations of LiRA and distillation attack using the *slope signal* achieve 41.5% and 25.4% TPR (with FPR fixed at 0.5%) on CIFAR100 with IID data partitioning, our KTA with $\alpha = 0.4$ further improves TPR to 50.0%. Recall from Table 7 that the original TPRs of Lira and distillation attack are lower than 7% for the same setting. Such performance improvement of KTA highlights the effectiveness of using **(i)** the *slope signal* and **(ii)** the *regularizer* for reference model training.

7 Conclusion and Future Work

We study privacy auditing for federated learning. We propose to use the *slope signal* to extract membership information from the training trajectory of FL. Built upon it, we propose FTA, which simultaneously achieves high effectiveness and efficiency for auditing privacy in FL, without disrupting the training process. We also propose an enhanced auditing algorithm called FTA, catering to parties who want to invest more computational power for a more accurate assessment of privacy risks. KTA further improves the performance of FTA, by incorporating the slope signal with the knowledge distillation technique.

For future work, we plan to utilize our solution to assess the privacy risks under different FL scenarios, including auditing the risks in personalized FL. Further improving the effectiveness of the slope signal is also a promising direction.

8 Acknowledgements

The authors would like to thank Jason Martin and Bao Ergute for their comments on the earlier versions of this work. The authors would also like to thank the reviewers for their valuable comments during the rebuttal and revision process. This research is supported by Intel within the www.private-ai.org center, Meta faculty research award, Google PDPO faculty research award, the NUS Early Career Research Award (NUS ECRA award number NUS ECRA FY19 P16), and the National Research Foundation, Singapore under its Strategic Capability Research Centres Funding Initiative. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not reflect the views of the National Research Foundation, Singapore.

References

[1] General data protection regulation. https://en.wikipedia.org/wiki/General_Data_Protection_Regulation, 2018.

[2] Galen Andrew, Peter Kairouz, Sewoong Oh, Alina Oprea, Hugh Brendan McMahan, and Vinith Menon Suriyakumar. One-shot empirical privacy estimation for federated learning. In *The Twelfth International Conference on Learning Representations*, 2024.

[3] apolanco3225. Medical mnist classification. https://github.com/apolanco3225/Medical-MNIST-Classification, 2017.

[4] Franziska Boenisch, Adam Dziedzic, Roei Schuster, Ali Shahin Shamsabadi, Ilia Shumailov, and Nicolas Papernot. When the curious abandon honesty: Federated learning is not private. In *2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)*, pages 175–199. IEEE, 2023.

[5] Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. Membership inference attacks from first principles. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1897–1914. IEEE, 2022.

[6] Liam H Fowl, Jonas Geiping, Wojciech Czaja, Micah Goldblum, and Tom Goldstein. Robbing the fed: Directly obtaining private data in federated learning with modified models. In *International Conference on Learning Representations*, 2021.

[7] Chong Fu, Xuhong Zhang, Shouling Ji, Jinyin Chen, Jingzheng Wu, Shanqing Guo, Jun Zhou, Alex X Liu, and Ting Wang. Label inference attacks against vertical federated learning. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 1397–1414, 2022.

[8] Francis Galton. Regression towards mediocrity in hereditary stature. *Journal of the Anthropological Institute*, 15:246–263, 1886.

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

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[11] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[12] Hongsheng Hu, Zoran Salcic, Lichao Sun, Gillian Dobbie, and Xuyun Zhang. Source inference attacks in federated learning. In *2021 IEEE International Conference on Data Mining (ICDM)*, pages 1102–1107. IEEE, 2021.

[13] Md Nazmul Islam, Mehedi Hasan, Md Kabir Hossain, Md Golam Rabiul Alam, Md Zia Uddin, and Ahmet Soylu. Vision transformer and explainable transfer learning models for auto detection of kidney cyst, stone and tumor from ct-radiography. *Scientific Reports*, 12(1):1–14, 2022.

[14] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

[15] Matthew Jagielski, Jonathan Ullman, and Alina Oprea. Auditing differentially private machine learning: How private is private sgd? *Advances in Neural Information Processing Systems*, 33:22205–22216, 2020.

[16] Matthew Jagielski, Stanley Wu, Alina Oprea, Jonathan Ullman, and Roxana Geambasu. How to combine membership-inference attacks on multiple updated machine learning models. *Proceedings on Privacy Enhancing Technologies*, 2023.

[17] Matthew Jagielski, Stanley Wu, Alina Oprea, Jonathan Ullman, and Roxana Geambasu. How to combine membership-inference attacks on multiple updated machine learning models. *Proceedings on Privacy Enhancing Technologies*, 3:211–232, 2023.

[18] Bargav Jayaraman, Lingxiao Wang, Katherine Knipmeyer, Quanquan Gu, and David Evans. Revisiting membership inference under realistic assumptions. *Proc. Priv. Enhancing Technol.*, 2021(2):348–368, 2021.

[19] Daniel S Kermany, Michael Goldbaum, Wenjia Cai, Carolina CS Valentim, Huiying Liang, Sally L Baxter, Alex McKeown, Ge Yang, Xiaokang Wu, Fangbing Yan, et al. Identifying medical diagnoses and treatable diseases by image-based deep learning. *cell*, 172(5):1122–1131, 2018.

[20] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[21] Klas Leino and Matt Fredrikson. Stolen memories: Leveraging model memorization for calibrated {White-Box} membership inference. In *29th USENIX security symposium (USENIX Security 20)*, pages 1605–1622, 2020.

[22] Jiacheng Li, Ninghui Li, and Bruno Ribeiro. Effective passive membership inference attacks in federated learning against overparameterized models. In *The Eleventh International Conference on Learning Representations*, 2023.

[23] Zhuohang Li, Jiaxin Zhang, Luyang Liu, and Jian Liu. Auditing privacy defenses in federated learning via generative gradient leakage. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10132–10142, 2022.

[24] Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. *Advances in Neural Information Processing Systems*, 33:2351–2363, 2020.

[25] Yiyong Liu, Zhengyu Zhao, Michael Backes, and Yang Zhang. Membership inference attacks by exploiting loss trajectory. *arXiv preprint arXiv:2208.14933*, 2022.

[26] Yunhui Long, Lei Wang, Diyue Bu, Vincent Bindschaedler, Xiaofeng Wang, Haixu Tang, Carl A Gunter, and Kai Chen. A pragmatic approach to membership inferences on machine learning models. In *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 521–534. IEEE, 2020.

[27] Samuel Maddock, Alexandre Sablayrolles, and Pierre Stock. CANIFE: Crafting canaries for empirical privacy measurement in federated learning. In *The Eleventh International Conference on Learning Representations*, 2023.

[28] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[29] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE symposium on security and privacy (SP)*, pages 691–706. IEEE, 2019.

[30] Milad Nasr, Jamie Hayes, Thomas Steinke, Borja Balle, Florian Tramèr, Matthew Jagielski, Nicholas Carlini, and Andreas Terzis. Tight auditing of differentially private machine learning. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 1631–1648, 2023.

[31] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning. In *Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP)*, pages 1–15, 2018.

[32] Milad Nasr, Shuang Songi, Abhradeep Thakurta, Nicolas Papernot, and Nicholas Carlin. Adversary instantiation: Lower bounds for differentially private machine learning. In *2021 IEEE Symposium on security and privacy (SP)*, pages 866–882. IEEE, 2021.

[33] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[34] Nicola Rieke, Jonny Hancox, Wenqi Li, Fausto Milletari, Holger R Roth, Shadi Albarqouni, Spyridon Bakas, Mathieu N Galtier, Bennett A Landman, Klaus Maier-Hein, et al. The future of digital health with federated learning. *NPJ digital medicine*, 3(1):1–7, 2020.

[35] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2017.

[36] Liwei Song and Prateek Mittal. Systematic evaluation of privacy risks of machine learning models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2615–2632, 2021.

[37] Tim van Erven and Peter Harremoës. Rényi divergence and kullback-leibler divergence. *IEEE Trans. Inf. Theory*, 60(7):3797–3820, 2014.

[38] Aidmar Wainakh, Fabrizio Ventola, Till Müßig, Jens Keim, Carlos Garcia Cordero, Ephraim Zimmer, Tim Grube, Kristian Kersting, and Max Mühlhäuser. User-level label leakage from gradients in federated learning. *Proceedings on Privacy Enhancing Technologies*, 2:227–244, 2022.

[39] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris S. Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[40] Wensi Yang, Yuhang Zhang, Kejiang Ye, Li Li, and Cheng-Zhong Xu. Ffd: A federated learning based method for credit card fraud detection. In *International conference on big data*, pages 18–32. Springer, 2019.

[41] Jiayuan Ye, Aadyaa Maddi, Sasi Kumar Murakonda, Vincent Bindschaedler, and Reza Shokri. Enhanced membership inference attacks against machine learning models. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 3093–3106, 2022.

[42] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st computer security foundations symposium (CSF)*, pages 268–282. IEEE, 2018.

[43] Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Nghia Hoang, and Yasaman Khazaeni. Bayesian nonparametric federated learning of neural networks. In *International conference on machine learning*, pages 7252–7261. PMLR, 2019.

[44] Sajjad Zarifzadeh, Philippe Liu, and Reza Shokri. Low-cost high-power membership inference attacks. *arXiv preprint arXiv:2312.03262*, 2023.

[45] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. *Advances in neural information processing systems*, 32, 2019.

A Full Description of Algorithms

For completeness, we present the detailed algorithmic description for FTA and KTA as in Algorithms 1 and 2. For KTA, the adversary trains reference models such that the target point is included in exactly half of the models (i.e., “IN” models) and excluded in the rest (i.e., “OUT” models) and computes the slope for the target point for all reference models (Lines 3-11 in Algorithm 2). Based on the computed slope from the “IN” models and “OUT” models, the adversary estimates the slope distributions for the target point when it is in and not in the training dataset. As we show in Figure 1, the slope distribution looks like a Gaussian. Therefore, we fit the slope distribution into a Gaussian distribution, whose mean and variance are the empirical mean and empirical variance of the slope computed on the “IN” models and the “OUT” models) (Lines 13-16 in Algorithm 2). The idea of fitting the empirical distribution to a Gaussian was also used in [5]. Lastly, given the estimated slope distributions, we compute the likelihood ratio based on two distributions to determine if the point is used for training or not.

B Additional Experiments

Datasets. For completeness, we describe the additional datasets we have used beyond the CIFAR datasets. For all datasets, we follow the same pre-processing as in [22].

Algorithm 1 Free Training Attack (FTA)

Input: A sequence of models $\theta^1, \ldots, \theta^t$, the target point z, data population pool π (can be a subset of the validation dataset).

```
Let $\mathcal{B}_{pop} = \{\}$
$b_{obs} = B((\theta^1, \ldots, \theta^t), z)$ ▷Initialize the membership signal set for non-members
for $M$ times do
    $z_{pop} \leftarrow \pi$ ▷Sample a non-member from the distribution
    $b_{pop} = B((\theta^1, \ldots, \theta^t), z_{pop}, \phi)$ ▷Compute the slope of the signal for the non-member
    $\mathcal{B}_{pop} = \mathcal{B}_{pop} \cup \{b_{pop}\}$ ▷Update the membership signal set for non-members
end for ▷Compute the slope of the signal for the target data
return $\frac{|\{b_{pop} \in \mathcal{B}_{pop} : b_{pop} \leq b_{obs}\}|}{|\mathcal{B}_{pop}|}$ ▷Compute the likelihood that the target point is a non-member.
```

Purchase: Derived from Kaggle’s “acquire valued shopper” challenge, the dataset includes records for thousands of individuals and is processed as described in [35]. It features 600 binary attributes per instance, is divided into 100 classes, and contains 197,324 instances in total. This dataset is commonly utilized to assess membership inference attacks, employing the neural network model from [31].

Texas: The dataset comprises hospital discharge records published by the Texas Department of State Health Services and features information on inpatient stays across multiple facilities. This processed dataset, as described in [35], contains 67,330 records with 6,170 binary features representing the 100 most frequent medical procedures. It is divided into 100 classes, each corresponding to a different patient type, and is employed to evaluate membership inference attacks.

Medical MNIST [3]: The dataset contains $58,954$ MNIST-style medical images in 64×64 resolution across six classes. We sampled a balanced dataset with $53,724$ images, each class having $8,954$ images, and resized all images to 32×32.

Pneumonia [19]: The dataset consists of Chest X-ray images from pediatric patients aged one to five at Guangzhou Women and Children’s Medical Center. We sub-sample a balanced dataset of $3,166$ X-rays, originally varying in size, and uniformly resize images to 64×64.

Retinal OCT Image [19]: The dataset includes $84,492$ high-resolution retinal OCT images for diagnosing conditions across four classes: CNV, DME, DRUSEN, and NORMAL. We sampled a balanced training set with $35,472$ images. We resize images to 64 uniformly because the original images are in different sizes.

CT Kidney [13]: The dataset is from PACS [13], collected from various Dhaka hospitals, includes diagnoses of "tumor," "cyst," "normal," or "stone" from coronal and axial CT scans. These images, from whole abdomen and urogram studies,

Algorithm 2 Knowledge Transfer Reference Attack (KTA).

Input: A sequence of models $\theta^1, \ldots, \theta^t$, the target point z, the data population pool π of target party k^* (the union of the local training and validation datasets).

```
$\mathcal{B}_{in} = \{\}, \mathcal{B}_{out} = \{\}$    ▷Initialize the membership signal sets for the IN models and OUT models
$b_{obs} = B(\{\theta^1, \ldots, \theta^t\}, z)$    ▷Compute the slope of the signal on the target point from the target models
for $N$ times do
    $S_{sh} \leftarrow \pi^{n_{sh}}$    ▷Sample a reference dataset of size $n_{sh}$
    $\theta^1_{in}, \theta^2_{in}, \ldots, \theta^t_{in} \leftarrow \mathcal{T}_{sh}(\{\theta^1, \theta^2, \ldots, \theta^t\}, S_{sh} \cup \{z\})$    ▷Train a reference model with the target point using Algorithm 3
    $b_{z,in} = B(\{\theta^1_{in}, \theta^2_{in}, \ldots, \theta^t_{in}\}, z)$    ▷Compute the slope of the signal based on the IN reference model
    $\mathcal{B}_{in} = \mathcal{B}_{in} \cup \{b_{z,in}\}$    ▷Update the membership signal set for IN models
    $\theta^1_{out}, \theta^2_{out}, \ldots, \theta^t_{out} \leftarrow \mathcal{T}_{sh}(\{\theta^1, \theta^2, \ldots, \theta^t\}; S_{sh})$    ▷Train a reference model without the target point
    $b_{z,out} = B(\{\theta^1_{out}, \theta^2_{out}, \ldots, \theta^t_{out}\}, z)$    ▷Compute the slope of the signal based on the OUT reference model
    $\mathcal{B}_{out} = \mathcal{B}_{out} \cup \{b_{z,out}\}$    ▷Update the membership signal set for OUT models
end for
$\mu_{in} = \mathtt{mean}(\mathcal{B}_{in})$    ▷Model the slope distribution for IN models and OUT models using Gaussian distributions
$\sigma^2_{in} = \mathtt{var}(\mathcal{B}_{in})$
$\mu_{out} = \mathtt{mean}(\mathcal{B}_{out})$
$\sigma^2_{out} = \mathtt{var}(\mathcal{B}_{out})$
Return $\Lambda = \frac{p(b_{obs} | \mathcal{N}(\mu_{out}, \sigma^2_{out}))}{p(b_{obs} | \mathcal{N}(\mu_{in}, \sigma^2_{in}))}$    ▷Compute the likelihood ratio of the target point based on estimated distributions
```

Algorithm 3 Reference model training algorithm $\mathcal{T}_{sh}$.

Input: A sequence of target global models $\bar{\theta}^1, \ldots, \bar{\theta}^t$; the reference dataset S_{sh}.

```
Initialize reference model $\theta^0_s$
for $u = 1 \ldots t$ do
Update the reference model parameter $\theta^u_s$ as in Eq. (6).
end for
return $\theta^1_s, \theta^2_s, \ldots, \theta^t_s$
```

were de-identified and converted to lossless jpg format after careful selection. A balanced dataset of 5,508 images, originally in various sizes, was uniformly resized to 64 × 64. Each image's accuracy was reconfirmed by a radiologist and a medical technologist.

Skin disease: The dataset is sourced from Dermnet (http://www.dermnet.com/), comprises approximately 19,559 images of 23 skin disease types, serving as an educational online dermatology resource. The dataset is accessible at https://www.kaggle.com/datasets/shubhamgoel27/dermnet. We formed a balanced dataset with 6,095 images.

Results in non-IID settings. In the Tabel 9, we show the performance of our algorithms and the baselines in the non-IID settings. Our algorithms, FTA and KTA, consistently outperform other baselines in all settings.

Training trajectory of FedSGD. We show the train and test performance of FedSGD in Figure 8. As we have mentioned in Section 5.4, the training and test accuracies of FedAvg are much more stable than FedAvg. This phenomenon explains why FedSGD exhibits lower privacy risks than FedAvg.

Table 9: Effectiveness of auditing local models with the Dir(0.5) data partitioning.

Method	TPR @0.1% FPR		TPR @0.5% FPR		TPR @1% FPR	
	C-10	C-100	C-10	C-100	C-10	C-100
01-loss [42]	0.0%	0.1%	0.0%	3.0%	0.0%	6.3%
Gradient-Diff [22]	0.1%	1.8%	1.2%	7.2%	2.5%	11.3%
Lira [5]	0.4%	2.4%	2.3%	7.6%	4.1%	15.8%
FTA (loss)	1.4%	10.5%	3.4%	22.1%	5.5%	33.6%
FTA (confidence)	0.2%	13.6%	4.6%	38.6%	8.2%	55.7%
FTA (logit)	0.1%	0.0%	0.5%	5.4%	0.9%	19.0%
KTA (loss)	0.8%	10.7%	3.5%	24.3%	5.7%	36.6%
KTA (confidence)	0.4%	20.3%	5.3%	51.4%	7.6%	62.5%
KTA (logit)	0.9%	38.1%	5.7%	59.1%	10.5%	70.3%

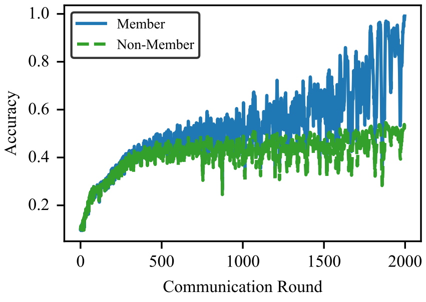

Figure 8: Classification accuracy of the global model in FedSGD during FL.