Active Membership Inference Attack under Local Differential Privacy in Federated Learning

Truc Nguyen
University of Florida
truc.nguyen@ufl.edu

Phung Lai
New Jersey Institute of Technology
tl353@njit.edu

Khang Tran
New Jersey Institute of Technology
kt36@njit.edu

NhatHai Phan
New Jersey Institute of Technology
phan@njit.edu

My T. Thai*
University of Florida
mythai@cise.ufl.edu

Abstract

Federated learning (FL) was originally regarded as a framework for collaborative learning among clients with data privacy protection through a coordinating server. In this paper, we propose a new active membership inference (AMI) attack carried out by a dishonest server in FL. In AMI attacks, the server crafts and embeds malicious parameters into global models to effectively infer whether a target data sample is included in a client's private training data or not. By exploiting the correlation among data features through a non-linear decision boundary, AMI attacks with a certified guarantee of success can achieve severely high success rates under rigorous local differential privacy (LDP) protection; thereby exposing clients' training data to significant privacy risk. Theoretical and experimental results on several benchmark datasets show that adding sufficient privacy-preserving noise to prevent our attack would significantly damage FL's model utility.

1 INTRODUCTION

Federated Learning (FL) has emerged as a promising large-scale collaborative learning framework in recent years. By design, FL enables participating clients to collaboratively train a global model through a coordinating server. Although training data never leaves clients' devices, a dishonest server can still infer the membership information of any client's training data through observing their local model updates by using (passive or active) membership inference attacks [Shokri et al., 2017, Salem et al., 2019, Song and Mittal, 2021, Nasr et al., 2019]. For that reason, FL in its primitive form offers little to no privacy protection.

To address the problem, several privacy-preserving mechanisms, such as local differential privacy (LDP), have been developed to challenge membership inference (MI) attacks in general and active membership inference (AMI) attacks in particular by effectively protecting the membership information of client's training data with upper-bounded privacy leakage [Arachchige et al., 2019, Sun et al., 2021, Lai et al., 2021, Lyu et al., 2020]. Recent studies apparently show that LDP protection is effective in mitigating MI and AMI attacks [Rahman et al., 2018, Bernau et al., 2021, Gu et al., 2022]. The key reason for this result is that existing attacks have not fully conveyed privacy risks in FL by under-exploiting the correlation among data features and LDP protection. That poses previously unexplored privacy risk to the clients' local training data.

Key Contributions. To tackle that problem, we first formalize a new AMI threat model equipping with an AMI attack from a dishonest server. The key idea is that, given a target data sample, the server carefully crafts malicious weights of the global model such that the model updates from the clients would expose the membership information of the target data sample through the behavior of a chosen neuron. A chosen neuron is only activated given the target data sample controlled by a non-linear decision boundary embedded inside the malicious weights. With our non-linear decision boundary, the server can infer this membership information with severely high success rates. Furthermore, the server effortlessly achieves this result with a minimal change to the global model parameters within one training iteration.

In addition, we take a step forward and devise an AMI attack strategy under LDP protection to significantly amplify the

Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206.
*Corresponding author

privacy risk in FL. By adding a certain amount of privacy-preserving noise to the local data before training, LDP can protect the data with formal privacy leakage bounds (controlled by a privacy budget ε) [Dwork et al., 2014]. The key advantage of our AMI attack is exploiting the correlation among data features to distinguish the target data sample from others under LDP protection (Eq. 7). If LDP-preserving noise is insufficient to break this correlation (i.e., large privacy budgets ε), clients' local data will be at risk of our AMI attack with certified guarantees of success. Meanwhile, large privacy-preserving noise (i.e., small privacy budgets ε) can significantly damage the FL's model utility.

Our theoretical and experimental results in many benchmark datasets show that our AMI attacks stress-test the fundamental trade-offs between model utility and privacy risk in FL to a new level. That reflects through notably high success rates under rigorous LDP protection (i.e., small privacy budgets ε which significantly degenerate FL's model utility).

Organization. The remainder of the paper is structured as follows. Section 2 presents background and establishes our threat model. Section 3 introduces our AMI attack from a dishonest server in FL. We devise an attack strategy under LDP protection with certified guarantees of success in Section 4. Section 5 evaluates the performance of AMI attacks in several benchmark datasets. Section 6 discusses related work and Section 7 provides concluding remarks.

2 BACKGROUND AND THREAT MODEL

In this section, we briefly review the background of federated learning and differential privacy, and then introduce our active membership inference threat model.

2.1 Background

Federated Learning (FL). We focus on a horizontal setting of FL in which different clients hold the same set of features but different sets of samples. We denote $f_\theta : \mathbb{R}^d \to \mathbb{R}^k$ as a k-class neural network model that is parameterized by a set of weights θ. The aim of f_θ is to map a data point $x \in \mathbb{R}^d$ to a vector of posterior probabilities $f_\theta(x) = \mathcal{Y}$ over k classes, where the sum of all values in $\mathcal{Y}$ is 1.

FL is an iterative learning framework for training a global model f_θ on distributed data owned by N different clients $\{u_j\}_{j=1}^N$. A central server coordinates the training of f_θ by iteratively aggregating gradients computed locally by the clients. Let $i \in \mathbb{Z}_{\geq 0}$ be the current iteration of the FL protocol, and θ^i be the set of parameters at iteration i. At iteration $i = 0$, the global θ^i is initialized randomly by the central coordinating server. At every iteration i, a subset of $M < N$ clients is randomly selected to participate in the training. Each of the selected clients u_j receives f_{θ^i} from the central server and calculates the gradients G_j^i for f_{θ^i} using their local training batch $\mathcal{D}_j$. Specifically, $G_j^i = \nabla_{\theta^i}\mathcal{L}(\mathcal{D}_j, \theta^i)$ where $\mathcal{L}$ is a loss function. Then, each u_j uploads its gradients to the central server, who averages all of these gradients to compute the global model's parameters with a learning rate η:

$$G^i = \frac{1}{M}\sum_{j=1}^{M} G_j^i, \quad \theta^{i+1} = \theta^i - \eta G^i \tag{1}$$

The training continues until f_{θ^i} converges.

FL with Local Differential Privacy (LDP). Recent attacks have shown that clients' training data samples can be extracted from the shared gradients [Zhu et al., 2019, Yin et al., 2021]. These attacks underscore privacy risks in FL. Therefore, privacy-preserving mechanisms are needed to control and mitigate the privacy risks introduced by gradient sharing while optimizing utility.

Local differential privacy (LDP) [Dwork et al., 2006, Erlingsson et al., 2014] is one of the auspicious solutions, given its formal protection without an undue sacrifice in computation efficiency. LDP builds on the ideas of *randomized response* [Warner, 1965], which was initially introduced to allow survey respondents to provide their inputs while maintaining their confidentiality. The definition of ϵ-LDP is as follows:

Definition 1. *ϵ-LDP. A randomized algorithm $\mathcal{M}$ fulfills ϵ-LDP, if for any two inputs x and x', and for all possible outputs $\mathcal{O} \in \mathrm{Range}(\mathcal{M})$, we have: $Pr[\mathcal{M}(x) = \mathcal{O}] \leq e^\epsilon Pr[\mathcal{M}(x') = \mathcal{O}]$, where ϵ is a privacy budget and $\mathrm{Range}(\mathcal{M})$ denotes every possible output of $\mathcal{M}$.*

The privacy budget ϵ controls the amount by which the distributions induced by inputs x and x' may differ. A smaller value of ϵ enforces a stronger privacy guarantee but reduces model utility.

2.2 Active Membership Inference Threat Model

Previous studies typically focus on a scenario in which the central server is interested in uncovering client information by examining local updates from the clients, but still abiding by the system protocol. This threat model is commonly referred to as honest-but-curious or semi-honest. However, this threat model undermines the vulnerability of the FL system as in practice, the server can deviate from the protocol to strengthen the privacy attacks [Boenisch et al., 2021, Nguyen et al., 2022, Fowl et al., 2021]. In this work, we are thus interested in explicitly malicious (or actively dishonest) servers that may modify the model architecture and/or model parameters before dispatching them to the clients. In this regard, we propose an active membership inference threat model, in which a dishonest server maliciously adjusts the model parameters to determine whether a target data sample is in the local training dataset of a client.

$\underline{\mathsf{Exp}(\mathcal{A}, \mathcal{L}, \mathbb{D})}$:
$\mathcal{D} \sim \mathbb{D}^n$ # Sample n data points from $\mathbb{D}$ into $\mathcal{D}$
$b \xleftarrow{\$} \{0, 1\}$ # Flip a bit b uniformly at random
if $b = 1$ **then**
  $t \xleftarrow{\$} \mathcal{D}$ # Choose t uniformly from $\mathcal{D}$
end
else
  $t \sim \mathbb{D} \setminus \mathcal{D}$ # Sample t from $\mathbb{D}$ s.t. $t \notin \mathcal{D}$
end
$\theta \leftarrow \mathcal{A}^{\mathbb{D}}_{\mathsf{INIT}}(t)$ # The adversary receives t and returns a set of parameters θ
$G \leftarrow \nabla_\theta \mathcal{L}(\mathcal{D}, \theta)$ # Compute the gradients from θ and $\mathcal{D}$
$b' \leftarrow \mathcal{A}^{\mathbb{D}}(t, G)$ # The adversary receives t, G and returns a bit b'
Ret $[b' = b]$ # The game returns 1 if $b' = b$ (the adversary wins), 0 otherwise

Figure 1: AMI Threat Model as a Security Game.

We describe the active membership inference threat model as follows. We denote $\mathcal{A}$ as the central server in FL, which is also the adversary. Note that this threat model represents an attack at an arbitrary iteration that targets a specific client. Let $\mathcal{D} = \{(x_i, y_i)\}$ be the batch of training data of the target client. The set $\mathcal{D}$ contains sample $x_i \in \mathbb{R}^d$ and its ground-truth label $y_i \in \{1, ..., k\}$ with k classes. Suppose that $\mathcal{D}$ is sampled from a distribution $\mathbb{D}$ on (x_i, y_i) that the adversary $\mathcal{A}$ has knowledge of (i.e., similar to existing studies [Carlini et al., 2022, Yeom et al., 2018, Shokri et al., 2017]). This is practical in the real world since the server can collect a massive amount of data that covers the local data distribution of a sufficient number of clients [Shokri et al., 2017]. The adversary outputs maliciously crafted model parameters θ to the target client. The client sends the local gradients $G = \nabla_\theta \mathcal{L}(\mathcal{D}, \theta)$ back to the adversary. By observing the local gradients G, the goal of the server's attack is to determine whether a target (data) sample $t \in \mathbb{R}^d$ is included in the local training set $\mathcal{D}$. More formally, the adversary can be defined as the following function:

$$\mathcal{A}^{\mathbb{D}} : t, G \rightarrow \{0, 1\} \tag{2}$$

where $\mathcal{A}^{\mathbb{D}}$ denotes the query access to $\mathbb{D}$, 1 means $t \in \mathcal{D}$, and 0 otherwise.

We formalize this threat model as a security game $\mathsf{Exp}(\mathcal{A}, \mathcal{L}, \mathbb{D})$ between a challenger and the adversary in Fig. 1. From that, the adversary's advantage, or the attack success rate, is defined as follows:

$$\begin{aligned} \mathrm{Adv}^{\mathcal{A}} &= \Pr[\mathsf{Exp}(\mathcal{A}, \mathcal{L}, \mathbb{D}) = 1] \\ &= \frac{1}{2}\Pr[b' = 1 | b = 1] + \frac{1}{2}\Pr[b' = 0 | b = 0] \end{aligned} \tag{3}$$

where $\Pr[b' = 1 | b = 1]$ is the True Positive Rate (TPR), and $\Pr[b' = 0 | b = 0]$ is the True Negative Rate (TNR). The success rate $\mathrm{Adv}^{\mathcal{A}}$ should be greater than 0.5, which is the probability of random guessing.

3 ACTIVE MEMBERSHIP INFERENCE (AMI) ATTACK

This section first discusses the technical intuition of membership inference through gradients. Based on this concept, we then describe our proposed strategy to launch the AMI attack from a dishonest server.

3.1 Inferring Membership via Gradients

As shown in Fig. 1, the adversary $\mathcal{A}$ receives the gradients G that was computed on the training set $\mathcal{D}$, and $\mathcal{A}$ wishes to determine whether $t \in \mathcal{D}$. This section discusses how the membership information can be inferred through gradients.

Suppose that, on an input data point $x \in \mathbb{R}^d$, the output of the ***first fully-connected layer*** is expressed as $\mathrm{ReLU}(Wx + b) = \max(0, Wx + b)$ where $W \in \mathbb{R}^{r \times d}$ is the weight matrix of that layer and $b \in \mathbb{R}^r$ is the bias vector (r is the number of neurons in the layer). To express the output of the i-th neuron of that layer, we denote W_i as the corresponding row in the weight matrix and b_i as the corresponding component in the bias vector. We observe that, when $W_i x + b_i \leq 0$, the ReLU outputs zero, in other words, the neuron i is not activated by x. As a result, the gradient of neuron i, denoted by $G_i^{(x)}$, is zero at the data point x. Otherwise, when $W_i x + b_i > 0$, the gradient $G_i^{(x)}$ is non-zero.

As the gradient G is computed over the whole training set $\mathcal{D}$, the gradient of a neuron i received by the adversary is the average of gradients over all data points $x \in \mathcal{D}$, i.e., $G_i = \frac{1}{|\mathcal{D}|}\sum_{x \in \mathcal{D}} G_i^{(x)}$. If there exists a neuron i that is ***activated only*** by a target data sample t ($t \in \mathbb{R}^d$), and ***not activated*** by any other data samples $x \neq t$, then we have:

$$\begin{cases} \sum_{j=1}^{d} W_{ij} t_j > 0 \\ \sum_{j=1}^{d} W_{ij} x_j \leq 0, \quad \forall x \in \mathcal{D} \setminus t \end{cases} \tag{4}$$

note that we suppress the bias term for simplicity.

If $t \in \mathcal{D}$, then G_i will be non-zero; otherwise, $t \notin \mathcal{D}$ results in G_i being zero. From that, the adversary upon seeing G_i can easily infer whether the target data sample t was a part of the training set $\mathcal{D}$ or not. The formulation in Eq. (4) is similar to the framework proposed by [Boenisch et al., 2021] for conducting data reconstruction attacks where a neuron that is activated only by one sample can be used to perfectly reconstruct that sample. However, we shall see below that

this simple formulation is actually inapplicable to the membership inference attack, thus requiring a more advanced strategy to launch the attack.

3.2 Attack Strategy: Manipulating Model Parameters via Training a Chosen Neuron

As aforementioned, if there exists a neuron that is activated only by the target data sample t, it is sufficient to determine whether t is in the training set $\mathcal{D}$ or not. From the threat model, the adversary can determine the model parameters θ, which includes the weight matrix W. Obviously, the adversary can choose some neuron i and try to solve Eq. (4) for W to realize the conditions of the chosen neuron. Since the adversary does not know $\mathcal{D}$, it can only approximate Eq. (4) for all of its observed data samples x that are different from t (i.e., $x \neq t$ and $x \notin \mathcal{D}$). However, that makes Eq. (4) infeasible due to the linearity of the functions $\sum_{j=1}^{d} W_{ij} t_j$ and $\sum_{j=1}^{d} W_{ij} x_j$, as shown in Appendix A.

To address this issue, we introduce non-linearity into the equation. To do so, instead of relying on the first layer, the adversary can choose a neuron in the ***second fully-connected layer*** such that the neuron is activated only by the target data sample t, and not activated by any other data samples $x \neq t$. Let us denote $h \in \mathbb{R}^r$ as the weight vector of the chosen neuron in the second layer, the attack is successful if we can find (h, W) such that:

$$\begin{cases} \sum_{i=1}^{r} h_i \operatorname{ReLU}(\sum_{j=1}^{d} W_{ij} t_j) > 0 \\ \sum_{i=1}^{r} h_i \operatorname{ReLU}(\sum_{j=1}^{d} W_{ij} x_j) \leq 0, \quad \forall x \neq t \end{cases} \tag{5}$$

To solve Eq. (5), we can train the chosen neuron to be activated only by the target data sample t. For the training, we first put forth a logistic sigmoid function (σ) on the output of the chosen neuron. As a result, the function of the chosen neuron becomes:

$$\begin{aligned} s(x) &= \sigma(\sum_{i=1}^{r} h_i \operatorname{ReLU}(\sum_{j=1}^{d} W_{ij} x_j)) \\ &= \sigma(h \cdot \operatorname{ReLU}(Wx)) \end{aligned} \tag{6}$$

Next, we sample a dataset $\mathcal{X} \sim \mathbb{D}^m$, in which we assign a label 1 for the target data sample t and label 0 for all other data samples $x \in \mathcal{X} \setminus t$. After that, we train the chosen neuron using cross-entropy loss. The key idea is that the training process tries to make $s(t) = 1$, and $s(x) = 0$ for $x \neq t$. When $s(t) > 0.5$, it means that $h \cdot \operatorname{ReLU}(Wt) > 0$; otherwise, when $s(x) < 0.5$, we have $h \cdot \operatorname{ReLU}(Wx) < 0$ and that conforms to Eq. (5).

When the adversary receives the gradient G, it can observe the gradient of that chosen neuron to determine whether the target data sample t is in the training set $\mathcal{D}$ ($t \in \mathcal{D}$) or not, as discussed in the previous section. In particular, the adversary extracts the gradient of the chosen neuron, denoted by g_t, from G and sees whether g_t is non-zero. ***If g_t is zero***, the adversary predicts that ***the target data sample t is not in the training set*** $\mathcal{D}$ (i.e., $t \notin \mathcal{D}$). This is because the chosen neuron was not activated during the gradient computation on the training set $\mathcal{D}$. Otherwise, ***if g_t is non-zero***, then the adversary predicts that ***the target data sample t is in the training set*** $\mathcal{D}$ (i.e., $t \in \mathcal{D}$).

adversary $\mathcal{A}^{\mathbb{D}}_{\mathsf{INIT}}(t)$:
$\mathcal{X} \sim \mathbb{D}^m$
$D_A \leftarrow \bigcup_{x \in \mathcal{X} \setminus t} \{(x, 0)\}$
$D_A \leftarrow D_A \cup \{(t, 1)\}$
Train h, W (Eq. 6) from dataset D_A
Initialize θ
$\theta \leftarrow \theta \cup (h, W)$
Ret θ

adversary $\mathcal{A}^{\mathbb{D}}(t, G)$:
Extract g_t as the gradient of the chosen neuron from G
Ret $[g_t \neq 0]$

Figure 2: AMI Attack Strategy of the Adversary $\mathcal{A}$.

Fig. 2 shows a design of the adversary $\mathcal{A}$ according to the threat model in Fig. 1. Note that our attack strategy only modifies the parameters of 1 chosen neuron in the second layer and r associated neurons in the first layer. That makes our attack feasible by enabling us to make a minimal change to the model parameters, and the attack can be carried out within one FL training iteration.

4 AMI ATTACK UNDER LDP WITH CERTIFIED GUARANTEE OF SUCCESS

LDP is generally regarded as an effective defense against privacy inference attacks given its rigorous privacy protection compared with other approaches [Wagh et al., 2021]. To tolerate such mechanisms, our AMI attack exploits the correlation among input features captured through a non-linear decision boundary to distinguish the target data sample t from others (Eq. 5) under LDP protection. If LDP-preserving noise is insufficient to break this correlation (i.e., large privacy budgets ε), clients' local data will be at risk of our AMI attack. Meanwhile, large privacy-preserving noise (i.e., small privacy budgets ε) can significantly damage the model utility. In FL, it is challenging for clients to identify suitable privacy budgets given their limited local training data. Therefore, they usually rely on the server to provide the privacy budget ε and the LDP-preserving mechanism $\mathcal{M}$. That increases the risk of exposing their local training data to a dishonest server under our AMI attack. We focus on shedding light on the fundamental trade-offs between

ε-LDP protection, model utility, and privacy risk with a certified bound for our AMI attack to be successful.

Each client independently perturbs every training data sample in their local training data $\mathcal{D}$ using an LDP-preserving mechanism $\mathcal{M}$ (Def. 1) to obtain a randomized local training set $\mathcal{D}' = \mathcal{M}(\mathcal{D}, \varepsilon) = \{\mathcal{M}(x, \varepsilon)\}_{x \in \mathcal{D}}$. The client sends the gradients derived from the randomized training set $G = \nabla_\theta \mathcal{L}(\mathcal{D}', \theta)$ to the server. The mechanism $\mathcal{M}$ and the privacy budget ε are known to the adversary $\mathcal{A}$, thus, the adversary function in Eq. (2) is re-defined under LDP context as $\mathcal{A}^{\mathbb{D},\mathcal{M}}_{LDP} : t, G, \varepsilon \to \{0, 1\}$. We enhance the threat model to reflect the use of LDP in Fig. 3.

Attack Strategy. Given the gradients G computed on LDP-perturbed data $\mathcal{D}'$, applying the same attack strategy that was discussed in Section 3.2 would not work effectively. Suppose that the target sample t is in $\mathcal{D}$, and that the adversary were able to train a neuron that is activated only by the target sample t, i.e., $h \cdot \mathrm{ReLU}(Wt) > 0$ and $h \cdot \mathrm{ReLU}(Wx) < 0$ for $x \neq t$. However, it is very likely that t is not in the randomized local training set $\mathcal{D}'$ since t was randomized under LDP. As the client uses $\mathcal{D}'$ for gradients computation, the chosen neuron may remain inactivated, i.e., $h \cdot \mathrm{ReLU}(W \cdot \mathcal{M}(t, \varepsilon)) < 0$. Hence, the attack fails to infer the correct membership of t.

Therefore, it is necessary for the adversary to ensure that the chosen neuron is activated only by the randomized target sample $\mathcal{M}(t, \varepsilon)$. Similar to Eq. (5), we formulate this observation as finding (h, W) such that:

$$\begin{cases} h \cdot \mathrm{ReLU}(W\mathcal{M}(t, \varepsilon)) > 0 \\ h \cdot \mathrm{ReLU}(Wx) \leq 0, \quad \forall x \neq \mathcal{M}(t, \varepsilon) \end{cases} \tag{7}$$

To develop an effective attack for Eq. (7), it is essential for the chosen neuron to be activated if the client uses $\mathcal{M}(t, \varepsilon)$ regardless of the randomness introduced by the LDP-preserving mechanism $\mathcal{M}(\cdot, \varepsilon)$. To achieve our goal, we strengthen our attack by generating a set $\mathcal{T}$ of l perturbations $\mathcal{M}(t, \varepsilon)$, that is, invoking $\mathcal{M}(t, \varepsilon)$ l times with independent draws of LDP-preserving noise. Next, we sample a training set $\mathcal{X} \sim \mathbb{D}^m$ such that $\mathcal{X} \cap \mathcal{T} = \emptyset$. Then, we assign label 1 for samples in $\mathcal{T}$ (which contains randomized versions of t) and label 0 for samples in $\mathcal{X}$. Finally, we train the chosen neuron using cross-entropy loss. The remaining steps follow Section 3.2. Fig. 4 shows the design of the adversary $\mathcal{A}_{LDP}$ with respect to the threat model in Fig. 3.

Certified Guarantee of Success for AMI. Now, we derive certified guarantees for the adversary (Fig. 4) to be successful under ε-LDP protection. The AMI attack $\mathcal{A}^{\mathbb{D},\mathcal{M}}_{LDP}$ is successful in determining the membership of the target sample t if it can ensure that the chosen neuron is activated only by the LDP-preserving $\mathcal{M}(t, \varepsilon)$. Following the expected output stability property in DP [Lecuyer et al., 2019],

$\mathsf{Exp}_{LDP}(\mathcal{A}_{LDP}, \mathcal{L}, \mathbb{D}, \mathcal{M}, \varepsilon)$:
$\mathcal{D} \sim \mathbb{D}^n$
$b \xleftarrow{\$} \{0, 1\}$
if $b = 1$ **then**
  $t \xleftarrow{\$} \mathcal{D}$
end
else
  $t \sim \mathbb{D} \setminus \mathcal{D}$
end
$\theta \leftarrow \mathcal{A}^{\mathbb{D},\mathcal{M}}_{LDP,\mathsf{INIT}}(t, \varepsilon)$
$\mathcal{D}' \leftarrow \mathcal{M}(\mathcal{D}, \varepsilon)$ # Apply the LDP mechanism on $\mathcal{D}$
$G \leftarrow \nabla_\theta \mathcal{L}(\mathcal{D}', \theta)$
$b' \leftarrow \mathcal{A}^{\mathbb{D},\mathcal{M}}_{LDP}(t, G, \varepsilon)$
Ret $[b' = b]$

Figure 3: AMI Threat Model under LDP Mechanisms.

adversary $\mathcal{A}^{\mathbb{D},\mathcal{M}}_{LDP,\mathsf{INIT}}(t, \varepsilon)$:
Choose $l \in \mathbb{N}$
$\mathcal{T} \leftarrow \emptyset$
for $i = 1$ *to* l **do**
  $\mathcal{T} \leftarrow \mathcal{T} \cup \{\mathcal{M}(t, \varepsilon)\}$
end
$\mathcal{X} \sim \mathbb{D}^m \setminus \mathcal{T}$ # Sample $\mathcal{X} \sim \mathbb{D}^m$ s.t. $\mathcal{X} \cap \mathcal{T} = \emptyset$
$D_A \leftarrow \bigcup_{x \in \mathcal{X}} \{(x, 0)\}$
$D_A \leftarrow D_A \cup \left(\bigcup_{x \in \mathcal{T}} \{(x, 1)\}\right)$
Train h, W (Eq. 6) from dataset D_A
Initialize θ
$\theta \leftarrow \theta \cup (h, W)$
Ret θ

adversary $\mathcal{A}^{\mathbb{D},\mathcal{M}}_{LDP}(t, G, \varepsilon)$:
Extract g_t as the gradient of the chosen neuron from G
Ret $[g_t \neq 0]$

Figure 4: Attack Strategy of the Adversary $\mathcal{A}_{LDP}$.

in which the expected value of an ε-LDP algorithm with bounded output is not sensitive to small changes in the input, the trained attack $\mathcal{A}^{\mathbb{D},\mathcal{M}}_{LDP}$ is certifiably robust to $\mathcal{M}(\cdot, \varepsilon)$ if the following condition holds:

$$\begin{cases} \mathbb{E}\big[v(t)\big] > 0 \\ \mathbb{E}\big[v(x)\big] \leq 0, \quad x \neq \mathcal{M}(t, \varepsilon) \end{cases} \tag{8}$$

where $v(t) = h \cdot \mathrm{ReLU}(W\mathcal{M}(t, \varepsilon))$ and $v(x) = h \cdot \mathrm{ReLU}(Wx)$ are the values of the chosen neuron, given the randomized target sample $\mathcal{M}(t, \varepsilon)$ and any other data samples $x \neq \mathcal{M}(t, \varepsilon)$, respectively.

However, due to the potentially complex nature of the post-noise computation, we cannot precisely compute the expectations in Eq. (8). We therefore resort to Monte Carlo sampling to estimate the expectations $\hat{\mathbb{E}}(\cdot)$. This estimation is obtained by invoking $\mathcal{M}(\cdot)$ multiple times with independent draws of the noise over the input. We denote $v_p(t)$ as

the p draws of $\mathcal{M}(t,\varepsilon)$ from the target sample t and $v_q(x)$ as the q draws of $\mathcal{M}(x,\varepsilon)$ from the sample x. Then, we replace $\mathbb{E}\big[v(t)\big]$ with $\hat{\mathbb{E}}\big[v(t)\big] = \frac{1}{p}\sum_p v_p(t)$ and replace $\mathbb{E}\big[v(x)\big]$ with $\hat{\mathbb{E}}\big[v(x)\big] = \frac{1}{q}\sum_q v_q(x)$, where p an q are the number of invocations of $\mathcal{M}(\cdot)$ for t and x, respectively.

The key idea is to simultaneously ensure that the lower bound $\hat{\mathbb{E}}^{lb}\big[v(t)\big]$ is larger than 0 and the upper bound $\hat{\mathbb{E}}^{ub}\big[v(x)\big]$ is smaller than or equal to 0 with a broken probability δ. That provides a certified guarantee for the Eq. (8) to hold. We compute $(1-\delta)$-confidence the lower bound $\hat{\mathbb{E}}^{lb}\big[v(t)\big]$ and the upper bound $\hat{\mathbb{E}}^{ub}\big[v(x)\big]$ by using Hoeffding's inequality [Hoeffding, 1963], as follows:

$$\hat{\mathbb{E}}^{lb}\big[v(t)\big] \triangleq \hat{\mathbb{E}}\big[v(t)\big] - \text{Range}\big(v(t)\big)\sqrt{-\frac{\ln(\delta)}{2p}} \tag{9}$$

$$\hat{\mathbb{E}}^{ub}\big[v(x)\big] \triangleq \hat{\mathbb{E}}\big[v(x)\big] + \text{Range}\big(v(x)\big)\sqrt{-\frac{\ln(\delta)}{2q}} \tag{10}$$

where $\text{Range}\big(v(\cdot)\big)$ is the range of $v(\cdot)$.

By replacing the bounds in Eqs. (9) and (10) to the expectations in Eq. (8), we derive the certified guarantee so that Eq. (8) holds as in Theorem 1.

Theorem 1. *(Certified Guarantee Condition) Suppose that clients in FL apply the LDP-preserving $\mathcal{M}(\cdot,\varepsilon)$ to their local data. $\hat{\mathbb{E}}^{lb}\big[v(t)\big]$ and $\hat{\mathbb{E}}^{ub}\big[v(x)\big]$, computed as in Eqs. 9 and 10, are the $(1-\delta)$-confidence lower and upper bounds, respectively. The AMI attack $\mathcal{A}^{\mathbb{D},\mathcal{M}}_{LDP}$ is successful in inferring the membership of the target sample t in $\mathbb{D}$ if the following condition is satisfied:*

$$\begin{cases} \hat{\mathbb{E}}^{lb}\big[v(t)\big] > 0 \\ \hat{\mathbb{E}}^{ub}\big[v(x)\big] \leq 0, \quad x \neq \mathcal{M}(t,\varepsilon) \end{cases} \tag{11}$$

Proof of Theorem 1 is in Appendix B. At the attack time, we implement a *certified guarantee of success* as a search to return the minimal privacy budget ε and broken probability δ so that the condition in Theorem 1 holds, as follows:

Corollary 1. *Given a well-trained model f_θ and the target sample t, the AMI attack $\mathcal{A}^{\mathbb{D},\mathcal{M}}_{LDP}$ is guaranteed to be successful up to the privacy budget ε^* and the broken probability δ^* for which the condition in Theorem 1 checks out:*

$$(\varepsilon^*,\ \delta^*) = \arg\min_{\varepsilon,\delta} \textit{ s.t. } \begin{cases} \hat{\mathbb{E}}^{lb}\big[h\cdot \text{ReLU}(W\mathcal{M}(t,\varepsilon))\big] > 0 \\ \hat{\mathbb{E}}^{ub}\big[h\cdot \text{ReLU}(Wx)\big] \leq 0, x \neq \mathcal{M}(t,\varepsilon) \end{cases} \tag{12}$$

Since the mechanism $\mathcal{M}(\cdot)$ and the privacy budget ε are known to the adversary $\mathcal{A}$, we only do a line search to find the minimal δ for a given ε^*.

5 EVALUATION

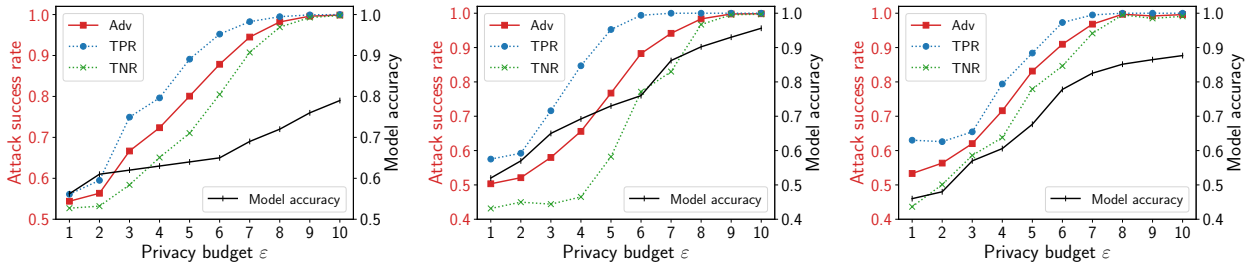
This section validates the effectiveness of our AMI attack by gauging its success rate. We particularly focus on evaluating how well it performs under LDP protection. Our implementation of the attack is available at https://github.com/trucndt/ami.

Experimental Settings. We evaluate AMI attack with three benchmark datasets, including CIFAR-10 [Krizhevsky et al., 2009], ImageNet [Deng et al., 2009], and CelebA [Liu et al., 2015]. Each dataset includes a training set and a validation set. The training set is used to sample the local training set $\mathcal{D}$ in the threat model (Figs. 1 and 3), while the validation set is used as the sampled set $\mathcal{X}$ of the adversary. Our experiment follows the security game in Figs. 1 and 3, and the success rate is calculated using Eq. (3) after executing the game 10,000 times. The batch size $|\mathcal{D}|$ is chosen to be 20 for the CelebA dataset according to its specification (i.e., 20 face images per person). As in [Fowl et al., 2021, Boenisch et al., 2021, Geiping et al., 2020], we set $|\mathcal{D}|$ to 64 and 100 for the ImageNet and CIFAR-10 datasets, respectively. We modify $r = 1,000$ neurons in the first layer and 1 neuron in the second layer to carry out AMI attacks. Further details on the experimental settings can be found in Appendix D.

To realize $\mathcal{M}$ in Fig. 3, we use two different LDP mechanisms: OME [Lyu et al., 2020] and BitRand [Lai et al., 2021]. These mechanisms add LDP noises to the embeddings of data samples. Such embeddings are obtained via a pre-trained Resnet-18 model which results in feature vectors of 512 dimensions [He et al., 2016]. We show the results when using BitRand in this section and refer the readers to Appendix D for the results on OME. A background on BitRand and OME is provided in Appendix C.

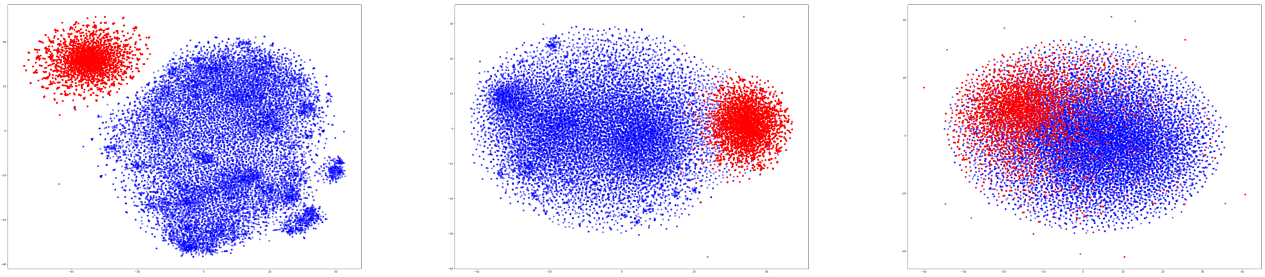
Attack Performance without LDP. In three datasets, our attack achieves near 100% success rate. The key reason behind this impressive success rate is that our attack strategy can easily train the chosen neuron to satisfy the attack objective, i.e., Eq. (5). Intuitively, the problem formulation in Eq. (5) is equivalent to finding a decision boundary overfitting to t in a way that can distinguish t against all other samples. As a result, increasing the number of neurons in the first layer (r) helps improve the attack performance, as it makes the model more over-fitting. In our experiments, we can achieve a 100% success rate with as few as 5 neurons ($r = 5$) in the first layer. We refer the readers to Appendix D for further analysis of this scenario as we focus the rest of this section on evaluating the attack under LDP.

Attack Performance under LDP. Fig. 5 shows that our attack introduces severe privacy risk to clients' local training data through strong attack success rates under LDP protection. Large privacy budgets ε (e.g., $\varepsilon \geq 5$) does minimal to



(a) CelebA (b) ImageNet (c) CIFAR-10

Figure 5: Attack success rate of AMI against an ε-LDP mechanism on CelebA, ImageNet, and CIFAR-10 datasets. The success rate is represented via the advantage (Adv), true positive rate (TPR), and true negative rate (TNR) according to Eq. 3. The baseline of random guessing is 0.5. The model accuracy illustrates the utility loss of the data when using LDP.



(a) $\varepsilon = 10$ (b) $\varepsilon = 7$ (c) $\varepsilon = 5$

Figure 6: Visualizing the distribution of the target sample t among other samples in the training set $\mathcal{D}$ using t-SNE embeddings. The red dots denote the target sample t and a multitude of its LDP noises $\mathcal{M}(t, \varepsilon)$, while the blue dots denote other non-target samples. These data samples are obtained from the CelebA dataset.

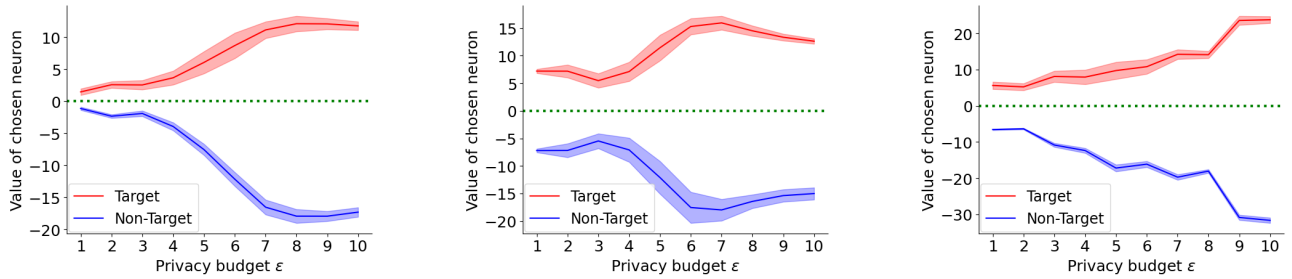
defend against our AMI attack. Across all three datasets, the model accuracy on the legitimate classification task remains acceptable given $\varepsilon \geq 5$. However, our attack imposes a severely high success rate (≥ 0.77), which approaches a near perfect success rate of 0.99 with $\varepsilon \geq 9$. When we reduce the privacy budget ($\varepsilon \in [3, 4]$), our attack still maintains a success rate of at least 0.67, 0.58, and 0.62, on CelebA, ImageNet, and CIFAR-10, respectively. With very low ε ($\varepsilon \in [1, 2]$), the model accuracy is severely damaged.

Furthermore, Fig. 5 depicts the TPR and TNR of our attack. Recall that TPR denotes how well the attack detects the presence of the target sample t in the training data $\mathcal{D}$, and TNR measures the ability to detect the absence of t. From the result, we can see that our attack has high TPR across all scenarios, which means it is sensitive to detecting the case where $t \in D$. Moreover, our TNR is greater than 0.5 indicating the capability of discerning the absence of t in the training data (except for the ImageNet dataset at $\varepsilon \leq 4$).

Training the Chosen Neuron under LDP. Training the chosen neuron is equivalent to determining a decision boundary that can distinguish the target sample (and its randomized variants) from any other samples. Fig. 6 visualizes how the samples in the training set $\mathcal{D}$ are distributed using t-SNE [Van der Maaten and Hinton, 2008]. At $\varepsilon = 10$, Fig. 6a shows that the t-SNE algorithm is able to group together the target sample t and its randomized variants. This is because the LDP mechanism imposes a small amount of noise such that t and its randomized variants $\mathcal{M}(t, \varepsilon)$ closely resemble one another. Hence, t-SNE models these by nearby points. Therefore, it is easy for our attack to train a neuron that can distinguish $\mathcal{M}(t, \varepsilon)$ from other samples, resulting in an attack success rate of about 0.99 as shown in Fig. 5.

At $\varepsilon = 5$, Fig. 6c shows that $\mathcal{M}(t, \varepsilon)$ blends into other samples, meaning that t-SNE is unable to group together the randomized variants of t as in the previous Fig. 6a. This is because the mechanism $\mathcal{M}(\cdot, \varepsilon)$ imposes a high amount of noise at $\varepsilon = 5$, so that all randomized variants $\mathcal{M}(t, \varepsilon)$ no longer closely resemble one another. This makes the task of finding the decision boundary between $\mathcal{M}(t, \varepsilon)$ and other samples more difficult. Nevertheless, our AMI attack can still attain a success rate of 0.80 (Fig. 5).

Certified Guarantee of Success. Given a privacy budget $\varepsilon \in [1, 10]$, in order to check the certified guanrantee conditions as in Theorem 1, we obtain the the lower and upper bounds $\hat{\mathbb{E}}^{lb}\big[v(t)\big]$ and $\hat{\mathbb{E}}^{ub}\big[v(x)\big]$ (Eqs. 9 and 10) by using $4,000$ ε-LDP target samples $\mathcal{M}(t, \varepsilon)$ and all ε-LDP non-target samples $\mathcal{M}(x, \varepsilon)$ from the validation set of each dataset. Here we use BitRand [Lai et al., 2021] as the



(a) CelebA (b) ImageNet (c) CIFAR-10

Figure 7: Certified guarantee of success for $\varepsilon \in [1, 10]$: Expectation (solid lines), upper-bound, and lower-bound (shaded areas surrounding the expectation) of the values of the chosen neuron. The larger the gap between the lower bound of the noisy target samples and the upper bound of the noisy non-target samples, the higher success rate the AMI achieves.

$\mathcal{M}(\cdot, \varepsilon)$ mechanism for the embeddings since BitRand is designed and well-suited for randomizing the embeddings.

Fig. 7 shows the certified guarantee of success for the CelebA, ImageNet, and CIFAR-10 datasets. We can derive certified guarantee of success for our AMI attack given $\varepsilon \geq 1$ with a small broken probability 10^{-8}. For rigorous privacy budgets, e.g., $\varepsilon \leq 3$, the output of the chosen neuron for both the target and non-target samples approaches the borderline associated with $v(\cdot) = 0$ (i.e., the dotted green lines), indicating a higher chance for AMI attacks to be failed given a broken probability δ. When the privacy budget ε increases, the output of the chosen neuron fits well with the attack objective. As a result, the expected value of the chosen neuron departs from the borderline, i.e., more positive given the target samples (i.e., the solid red lines) and more negative given the non-target samples (i.e., the solid blue lines). That implies a better attack success rate. Also, we observed that the overlapping area between the two distributions of the target samples and the non-target samples reduce significantly, which is consistent with our certified guarantee of success (Fig. 10, Appendix D) and our empirical results in Fig. 5.

6 RELATED WORK

Membership inference (MI) is one of the most fundamental privacy problems in machine learning [Carlini et al., 2022]. Several research has been carried out to convey the practical consequences of MI attacks [Backes et al., 2016, Pyrgelis et al., 2018] and analyze the models' vulnerability to MI [Carlini et al., 2019, Song and Mittal, 2021]. Along this direction, multiple MI attacks have been proposed in which the attacker only queries the model or observes its parameters to conduct the attacks [Shokri et al., 2017, Salem et al., 2019, Carlini et al., 2022]. Such attacks can be straightforwardly adapted to FL in which the central server is a passive adversary who tries to infer the membership information of clients' private data via inspecting their local models' parameters [Nguyen and Thai, 2022, Melis et al., 2019]. Recently, an AMI attack in FL proposed by [Nasr et al., 2019] considers a dishonest server that can interfere with the FL training protocol. However, this attack must be repeated in multiple training iterations to attain a high success rate. Furthermore, LDP has been shown to be an effective defense against these attacks [Rahman et al., 2018, Bernau et al., 2021, Gu et al., 2022].

Our work proposes a new AMI attack in FL where the dishonest server can maliciously modify the model weights to its advantage. We have proposed a strategy that results in minimal modifications to the model and can be executed in only one training iteration. More importantly, our attack can maintain a strong success rate even when the clients' data are protected by an LDP mechanism.

7 CONCLUSION AND DISCUSSION

In this paper, we have introduced a formal threat model for our AMI attack with dishonest FL servers, showing a more realistic privacy threat. Accordingly, we have proposed a new active membership inference (AMI) attack, exploiting the correlation among data features through a non-linear decision boundary. AMI attacks can achieve high success rates even under LDP protection, confirmed by both theoretical analysis and experimental evaluations. From this attack, our research has demonstrated that current implementations of FL provide virtually no privacy protection for clients.

With such a strong AMI attack, our future work would focus on the defenses. We discuss some challenges in devising such a solution as follows.

Noisy gradients with DPSGD. A potential defense against our attack is to let clients add DP noise to their gradients using DPSGD [Abadi et al., 2016] before sending them to the server, hindering the attacker from knowing the true value of the chosen neuron's gradient. However, recent work [Boenisch et al., 2021, Tramèr and Boneh, 2021] suggests that using DPSGD makes it impossible to train a good model for datasets like CIFAR-10 or ImageNet. Further-

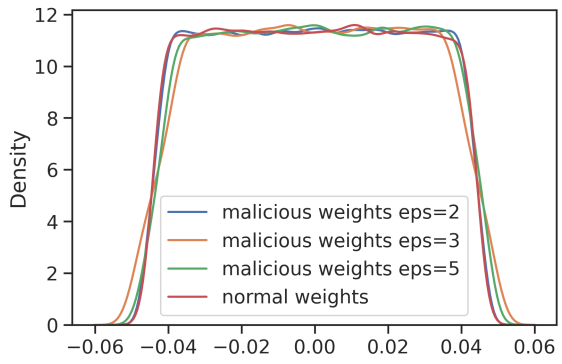


Figure 8: The kernel density estimation (KDE) of malicious and normal weights.

more, even if DPSGD is used, the attacker can still circumvent it by aggregating the noisy gradients over multiple FL iterations and potentially cancelling out the added DP noise. A detailed analysis on this can be found in Appendix E.

Detecting Malicious Weights is Challenging. The hardness in detecting malicious weights can be evaluated by examining the difference between malicious weights and normal weights (i.e., the weights that could be obtained from an honest server). Fig. 14 (Appendix) shows the distribution of the normal weights together with the distribution of the malicious weights when attacking under LDP protection at $\varepsilon = 2, 3$, and 5, and they largely resemble one another. Fig. 8 shows the 4 distributions using their respective kernel density estimation (KDE), which is used to visualize the shape of a data distribution, and represent the data using a continuous probability density curve. We can see that the malicious weights do not result in any abnormal distribution, making it indistinguishable from normal weights. This implies that, by observing the distribution of model weights, it is infeasible to determine whether the model weights have been modified maliciously by our attack.

Acknowledgements

This material is based upon work supported by the National Science Foundation under grants CNS-1935928, CNS-1935923, and CNS-2140477.

References

[Abadi et al., 2016] Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. (2016). Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318.

[Arachchige et al., 2019] Arachchige, P. C. M., Bertok, P., Khalil, I., Liu, D., Camtepe, S., and Atiquzzaman, M. (2019). Local differential privacy for deep learning. *IEEE Internet of Things Journal*, 7(7):5827–5842.

[Backes et al., 2016] Backes, M., Berrang, P., Humbert, M., and Manoharan, P. (2016). Membership privacy in microrna-based studies. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 319–330.

[Bernau et al., 2021] Bernau, D., Robl, J., Grassal, P. W., Schneider, S., and Kerschbaum, F. (2021). Comparing local and central differential privacy using membership inference attacks. In Barker, K. and Ghazinour, K., editors, *Data and Applications Security and Privacy XXXV*, pages 22–42, Cham. Springer International Publishing.

[Boenisch et al., 2021] Boenisch, F., Dziedzic, A., Schuster, R., Shamsabadi, A. S., Shumailov, I., and Papernot, N. (2021). When the curious abandon honesty: Federated learning is not private. *arXiv preprint arXiv:2112.02918*.

[Carlini et al., 2022] Carlini, N., Chien, S., Nasr, M., Song, S., Terzis, A., and Tramer, F. (2022). Membership inference attacks from first principles. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1897–1914. IEEE.

[Carlini et al., 2019] Carlini, N., Liu, C., Erlingsson, Ú., Kos, J., and Song, D. (2019). The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 267–284.

[Deng et al., 2009] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

[Dwork et al., 2006] Dwork, C., McSherry, F., Nissim, K., and Smith, A. (2006). Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284.

[Dwork et al., 2014] Dwork, C., Roth, A., et al. (2014). The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407.

[Erlingsson et al., 2014] Erlingsson, U., Pihur, V., and Korolova, A. (2014). Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC CCS*, pages 1054–1067.

[Fowl et al., 2021] Fowl, L. H., Geiping, J., Czaja, W., Goldblum, M., and Goldstein, T. (2021). Robbing the fed: Directly obtaining private data in federated learning with modified models. In *International Conference on Learning Representations*.

[Geiping et al., 2020] Geiping, J., Bauermeister, H., Dröge, H., and Moeller, M. (2020). Inverting gradients-how easy is it to break privacy in federated learning? *Advances in Neural Information Processing Systems*, 33:16937–16947.

[Gu et al., 2022] Gu, Y., Bai, Y., and Xu, S. (2022). Cs-mia: Membership inference attack based on prediction confidence series in federated learning. *Journal of Information Security and Applications*, 67:103201.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

[Hoeffding, 1963] Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*.

[Krizhevsky et al., 2009] Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

[Lai et al., 2021] Lai, P., Phan, H., Xiong, L., Tran, K. P., Thai, M., Sun, T., Dernoncourt, F., Gu, J., Barmpalios, N., and Jain, R. (2021). Bit-aware randomized response for local differential privacy in federated learning.

[Lecuyer et al., 2019] Lecuyer, M., Atlidakis, V., Geambasu, R., Hsu, D., and Jana, S. (2019). Certified robustness to adversarial examples with differential privacy. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 656–672. IEEE.

[Liu et al., 2015] Liu, Z., Luo, P., Wang, X., and Tang, X. (2015). Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*.

[Lyu et al., 2020] Lyu, L., Li, Y., He, X., and Xiao, T. (2020). Towards differentially private text representations. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1813–1816.

[Melis et al., 2019] Melis, L., Song, C., De Cristofaro, E., and Shmatikov, V. (2019). Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE symposium on security and privacy (SP)*, pages 691–706. IEEE.

[Nasr et al., 2019] Nasr, M., Shokri, R., and Houmansadr, A. (2019). Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*, pages 739–753. IEEE.

[Nguyen and Thai, 2022] Nguyen, T. and Thai, M. T. (2022). Preserving privacy and security in federated learning. *arXiv preprint arXiv:2202.03402*.

[Nguyen et al., 2022] Nguyen, T., Thai, P., Tre'R, J., Dinh, T. N., and Thai, M. T. (2022). Blockchain-based secure client selection in federated learning. In *2022 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–9. IEEE.

[Pyrgelis et al., 2018] Pyrgelis, A., Troncoso, C., and Cristofaro, E. D. (2018). Knock knock, who's there? membership inference on aggregate location data. In *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*. The Internet Society.

[Rahman et al., 2018] Rahman, M. A., Rahman, T., Laganière, R., Mohammed, N., and Wang, Y. (2018). Membership inference attack against differentially private deep learning model. *Trans. Data Priv.*, 11(1):61–79.

[Salem et al., 2019] Salem, A., Zhang, Y., Humbert, M., Berrang, P., Fritz, M., and Backes, M. (2019). Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*. The Internet Society.

[Shokri et al., 2017] Shokri, R., Stronati, M., Song, C., and Shmatikov, V. (2017). Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE.

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

[Sun et al., 2021] Sun, L., Qian, J., and Chen, X. (2021). LDP-FL: practical private aggregation in federated learning with local differential privacy. In Zhou, Z., editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pages 1571–1578. ijcai.org.

[Tramèr and Boneh, 2021] Tramèr, F. and Boneh, D. (2021). Differentially private learning needs better features (or much more data). In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

[Van der Maaten and Hinton, 2008] Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

[Wagh et al., 2021] Wagh, S., He, X., Machanavajjhala, A., and Mittal, P. (2021). Dp-cryptography: marrying differential privacy and cryptography in emerging applications. *Communications of the ACM*, 64(2):84–93.

[Wang et al., 2019] Wang, N., Xiao, X., Yang, Y., Zhao, J., Hui, S. C., Shin, H., Shin, J., and Yu, G. (2019). Collecting and analyzing multidimensional data with local differential privacy. In *IEEE ICDE*, pages 638–649.

[Warner, 1965] Warner, S. L. (1965). Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69.

[Yeom et al., 2018] Yeom, S., Giacomelli, I., Fredrikson, M., and Jha, S. (2018). Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st computer security foundations symposium (CSF)*, pages 268–282. IEEE.

[Yin et al., 2021] Yin, H., Mallya, A., Vahdat, A., Alvarez, J. M., Kautz, J., and Molchanov, P. (2021). See through gradients: Image batch recovery via gradinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16337–16346.

[Zhao et al., 2020] Zhao, Y., Zhao, J., Yang, M., Wang, T., Wang, N., Lyu, L., Niyato, D., and Lam, K. Y. (2020). Local differential privacy based federated learning for internet of things. *IEEE Internet of Things Journal*.

[Zhu et al., 2019] Zhu, L., Liu, Z., and Han, S. (2019). Deep leakage from gradients. *Advances in neural information processing systems*, 32.

A INFEASIBILITY OF LINEARITY FOR AMI

Suppose that, given the target data sample $t \in \mathbb{R}^d$, there exists a W that can satisfy Eq. (4) for all $x \neq t$. We choose $x_1 = t_1 + c$ and $x_i = t_i$ for $i > 1$ and $c > 0$. Denoting $w \equiv W_i$, from the second condition in Eq. (4), we have that:

$$w_1 c + \sum_{j=1}^{d} w_j t_j \leq 0 \Longrightarrow -w_1 c \geq \sum_{j=1}^{d} w_j t_j \tag{13}$$

Likewise, choosing $x'_1 = t_1 - c$ and $x'_i = t_i$ for $i > 1$, from the second condition in Eq. (4), we have:

$$-w_1 c + \sum_{j=1}^{c} w_j t_j \leq 0 \Longrightarrow w_1 c \geq \sum_{j=1}^{d} w_j t_j \tag{14}$$

As $\sum_{j=1}^{d} w_j t_j > 0$ by the first condition in Eq. (4), the two equations (13) and (14) contradicts one another. Therefore, there exists no W that can satisfy Eq. (4) for all $x \neq t$.

B PROOF OF THEOREM 1

Given the target sample t and any data samples $x \neq \mathcal{M}(t, \varepsilon)$, the AMI attack $\mathcal{A}_{LDP}^{\mathbb{D},\mathcal{M}}$ is successful in determining the membership of t if it can ensure that the chosen neuron is activated only by the LDP-preserving $\mathcal{M}(t, \varepsilon)$. Following the expected output stability property in DP [Lecuyer et al., 2019], in which the expected value of an ε-LDP algorithm with bounded output is not sensitive to small changes in the input, the trained attack $\mathcal{A}_{LDP}^{\mathbb{D},\mathcal{M}}$ is certifiably robust to $\mathcal{M}(\cdot, \varepsilon)$ if the following condition holds:

$$\begin{cases} \mathbb{E}\big[v(t)\big] > 0 \\ \mathbb{E}\big[v(x)\big] \leq 0, \quad x \neq \mathcal{M}(t, \varepsilon) \end{cases} \tag{15}$$

where $v(t) = h \cdot \mathrm{ReLU}(W\mathcal{M}(t, \varepsilon))$ and $v(x) = h \cdot \mathrm{ReLU}(Wx)$ are the values of the chosen neuron, given the randomized target sample $\mathcal{M}(t, \varepsilon)$ and any other data samples $x \neq \mathcal{M}(t, \varepsilon)$, respectively.

However, due to the potentially complex nature of the post-noise computation, we cannot precisely compute the expectations in Eq. 8. We therefore resort to Monte Carlo sampling to estimate the expectations $\hat{\mathbb{E}}(\cdot)$. This estimation is obtained by invoking $\mathcal{M}(\cdot)$ multiple times with independent draws of the noise over the input. We denote $v_p(t)$ as the p draws of $\mathcal{M}(t, \varepsilon)$ from the target sample t and $v_q(x)$ as the q draws of $\mathcal{M}(x, \varepsilon)$ from the sample x.

Denoting $\mathrm{Range}\big(v(\cdot)\big)$ as the range of $v(\cdot)$, $v(\cdot) \in \mathrm{Range}\big(v(\cdot)\big)$. In other words, $v(\cdot)$ is bounded in $\mathrm{Range}\big(v(\cdot)\big)$. Given a broken probability δ, using Hoeffding's inequality, with $t \geq 0$ we have:

$$\begin{aligned} P\Big((\frac{1}{p}\sum_{i=1}^{p} v(t)) - \mathbb{E}[v(t)] \geq t\Big) &= P\Big(\frac{1}{p}\sum_{i=1}^{p}\big(v(t) - \mathbb{E}[v(t)]\big) \geq t\Big) \\ &\leq \exp\Big(-\frac{2pt^2}{\frac{1}{p}\sum_{i=1}^{p}\mathrm{Range}\big(v(t)\big)^2}\Big) = \exp\Big(-\frac{2pt^2}{\mathrm{Range}\big(v(t)\big)^2}\Big) \end{aligned} \tag{16}$$

As mentioned in Section 4, we replace $\mathbb{E}[v(t)]$ in Eq. 16 with $\hat{\mathbb{E}}[v(t)]$. Given a broken probability δ, we have:

$$\exp\Big(-\frac{2pt^2}{\mathrm{Range}(v(t))^2}\Big) = \delta \Leftrightarrow t = \mathrm{Range}\big(v(t)\big)\sqrt{-\frac{\ln(\delta)}{2p}} \tag{17}$$

Similarly, with the non-target samples, we have:

$$t = \mathrm{Range}\big(v(x)\big)\sqrt{-\frac{\ln(\delta)}{2q}} \tag{18}$$

By leveraging the Monte Carlo sampling for the expectation estimation, we can replace $\mathbb{E}\big[v(t)\big]$ with $\hat{\mathbb{E}}\big[v(t)\big] = \frac{1}{p}\sum_p v_p(t)$ and replace $\mathbb{E}\big[v(x)\big]$ with $\hat{\mathbb{E}}\big[v(x)\big] = \frac{1}{q}\sum_q v_q(x)$, where p an q are the number of invocations of $\mathcal{M}(\cdot)$ for t and x, respectively.

The key idea is to simultaneously ensure that the lower bound $\hat{\mathbb{E}}^{lb}\big[v(t)\big]$ is larger than 0 and the upper bound $\hat{\mathbb{E}}^{ub}\big[v(x)\big]$ is smaller than or equal to 0 with a broken probability δ. That provides a certified guarantee for the Eq. (8) to hold. From Eqs. 17 and 18, we can compute $(1-\delta)$-confidence the lower bound $\hat{\mathbb{E}}^{lb}\big[v(t)\big]$ and the upper bound $\hat{\mathbb{E}}^{ub}\big[v(x)\big]$, as follows:

$$\hat{\mathbb{E}}^{lb}\big[v(t)\big] \triangleq \hat{\mathbb{E}}\big[v(t)\big] - \text{Range}\big(v(t)\big)\sqrt{-\frac{\ln(\delta)}{2p}} \tag{19}$$

$$\hat{\mathbb{E}}^{ub}\big[v(x)\big] \triangleq \hat{\mathbb{E}}\big[v(x)\big] + \text{Range}\big(v(x)\big)\sqrt{-\frac{\ln(\delta)}{2q}} \tag{20}$$

By replacing the bounds in Eqs. 19 and 20 to the expectations in Eq. 15, we we derive the certified guarantee so that Eq. (15) holds. In other words, The AMI attack $\mathcal{A}^{\mathbb{D},\mathcal{M}}_{LDP}$ is successful in inferring the membership of the target sample t in $\mathbb{D}$ if the following condition is satisfied:

$$\begin{cases} \hat{\mathbb{E}}^{lb}\big[v(t)\big] > 0 \\ \hat{\mathbb{E}}^{ub}\big[v(x)\big] \leq 0, \quad x \neq \mathcal{M}(t,\varepsilon) \end{cases} \tag{21}$$

Consequently, Theorem 1 holds.

C OME [Lyu et al., 2020] AND BITRAND [Lai et al., 2021]

Apart from applying LDP-preserving mechanisms in real values of inputs or gradients [Warner, 1965, Zhao et al., 2020, Wang et al., 2019], there is a line of work introducing LDP-preserving mechanisms to inputs or embedding features [Lai et al., 2021, Lyu et al., 2020, Arachchige et al., 2019]. In these mechanisms, they encode the original data or embedding features into binary vectors, then apply the LDP mechanisms on top of the binary vectors, before training the local models.

In OME, each bit i is randomized differently depending on whether it is the odd or even bit or it is bit 0 or 1, as follows:

$$\forall i \in [0, rl-1] : P(v'_x(i)=1) = \begin{cases} p_{1X} = \dfrac{\alpha}{1+\alpha}, \text{ if } i \in 2j, v_x(i)=1 \\ p_{2X} = \dfrac{1}{1+\alpha^3}, \text{ if } i \in 2j+1, v_x(i)=1 \\ q_X = \dfrac{1}{1+\alpha\exp(\frac{\varepsilon}{rl})}, \text{ if } v_x(i)=0 \end{cases} \tag{22}$$

where ε the total privacy budget.

This mechanism is similar to the Utility enhancing randomization (UER) mechanism (Theorem III.4 [Arachchige et al., 2019]). As shown in [Arachchige et al., 2019, Lyu et al., 2020], model accuracy is almost constant although ε is changed.

However, existing LDP mechanisms suffer from the curse of privacy composition in which excessive privacy budgets are consumed proportionally to the large dimensions of input or embedded features [Arachchige et al., 2019], gradients [Zhao et al., 2020, Wang et al., 2019], and training rounds [Zhao et al., 2020, Wang et al., 2019], causing loose privacy protection or inferior model accuracy [Wagh et al., 2021].

To mitigate the curse of privacy composition and to optimize the trade-off among privacy and model utility, [Lai et al., 2021] introduce bit-aware term $\frac{i\%l}{l}$ and a temperature α for better control of the randomization probabilities. In BITRAND, the randomization probabilities are adaptively randomized such that *"bits with a more substantial impact"* on model utility will

have *"smaller randomization probabilities (less noisy)"* and vice-versa under the same privacy budget, as follows:

$$\forall i \in [0, rl-1] : P(v'_x(i)=1) = \begin{cases} p_X = \dfrac{1}{1+\alpha \exp(\frac{i\%l}{l}\varepsilon)}, \text{ if } v_x(i)=1 \\ q_X = \dfrac{\alpha \exp(\frac{i\%l}{l}\varepsilon)}{1+\alpha \exp(\frac{i\%l}{l}\varepsilon)}, \text{ if } v_x(i)=0 \end{cases} \tag{23}$$

where $v_x(i) \in \{0,1\}$ is the value of v_x at the bit i, v'_x is the perturbed vector created by randomizing all the bits in v_x, ε is a privacy budget, and α is a parameter bounded with $0 < \alpha \leq \sqrt{\frac{\varepsilon+rl}{2r\sum_{i=0}^{l-1}\exp(2\frac{\varepsilon}{l}i\%l)}}$. The bit-aware term $\frac{i\%l}{l}$ to indicate the location of bit i, which is associated with the sensitivity of the bit at that location, in its l-bit binary encoded vector among rl concatenated binary bits.

D ADDITIONAL EXPERIMENTS

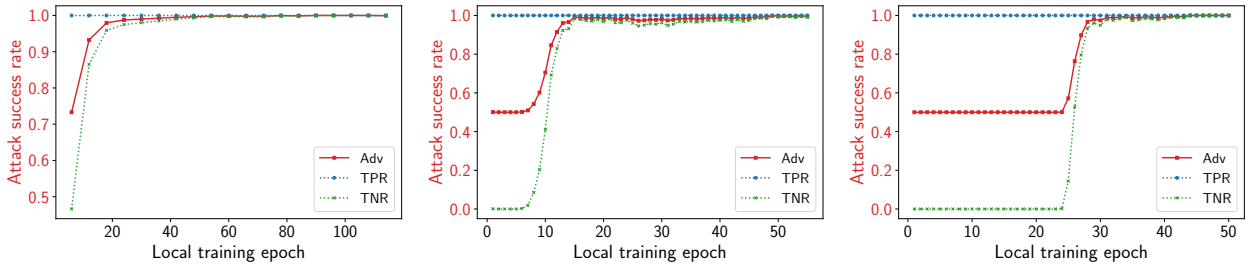
This section provides more details on the experimental settings, and presents additional experiments.

Settings. Our experiments in this paper are implemented using Python 3.8 and conducted on a single GPU-assisted compute node that is installed with a Linux 64-bit operating system. The allocated resources include 8 CPU cores (AMD EPYC 7742 model) with 2 threads per core, and 60GB of RAM. The node is also equipped with 8 GPUs (NVIDIA DGX A100 SuperPod model), with 80GB of memory per GPU.

The model accuracies in Figs. 5 and 11 are measured by evaluating the model on legitimate classification tasks. For the CelebA dataset [Liu et al., 2015], the task is to classify whether a person is smiling or not based on face images. With regard to the CIFAR-10 dataset [Krizhevsky et al., 2009], we use its original classification task with 10 classes. For the ImageNet dataset [Deng et al., 2009], we extract a subset of 10 classes: tench, English springer, cassette player, chain saw, church, French horn, garbage truck, gas pump, golf ball, and parachute[1]. Then, we evaluate the model performance on classifying those 10 classes.

To obtain the feature embeddings of data samples, we use the pre-trained Resnet-18 model from Img2Vec[2].

Attack performance without LDP. In this setting, the attack strategy in Fig. 2 requires the attacker to train the malicious parameters h, W, which takes multiple local training epochs. Fig. 9 shows the attack success rate per local epoch with 2,000 neurons in the first layer ($r = 2,000$). As can be seen, over time, the attack success rate reaches 100% across all three datasets. Table 1 shows the average number of local training epochs needed to train those parameters to achieve the 100% success rate, as we vary r. We observe that increasing r helps the attacker find the optimal parameters h, W faster. For CIFAR-10, with $r = 2,000$ neurons in the first layer, the attacker can easily train h, W within 50 epochs.



(a) CelebA (b) ImageNet (c) CIFAR-10

Figure 9: Attack success rate of AMI during local epochs of training h, W. The success rate is represented via the advantage (Adv), true positive rate (TPR), and true negative rate (TNR) according to Eq. 3. The baseline of random guessing is 0.5.

To understand the reason behind this behavior, we note that training h, W in this attack strategy is equivalent to finding a non-linear decision boundary that overfits to the target sample t (Eq. 5), thereby distinguishing the target sample t from

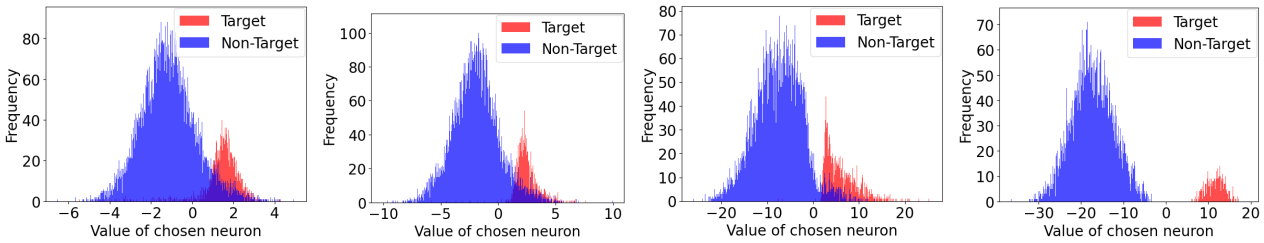
[1] https://github.com/fastai/imagenette
[2] https://github.com/christiansafka/img2vec

any other samples. Hence, increasing r raises the chance of over-fitting which, in turn, shortens the time to find h, W. Furthermore, Table 1 indicates that as few as $r = 5$ neurons are needed to attain the 100% success rate, albeit the longer training time.

Table 1: Number of local epochs needed to train h, W by the adversary $\mathcal{A}$ (Fig. 2) to obtain a 100% success rate. We vary the number of neurons in the first layer (r) and get the average number of local epochs over multiple runs.

Dataset	r	**No. of local epochs**
CelebA	5	3585
	500	763
	1000	497
	2000	297
ImageNet	5	1610
	500	131
	1000	88
	2000	63
CIFAR-10	5	309
	500	91
	1000	54
	2000	44

Attack performance under LDP. To shed light into how the privacy budget ε in the LDP mechanism, i.e., BitRand, affects the AMI success rate, we visualize the distribution of the values of the chosen neuron associated with the target and non-target samples. In Fig. 10a, with rigorous privacy budget (e.g., $\varepsilon = 1$), the mean values of the chosen neuron is positive given the target samples and negative given the non-target samples. However, there is a notable overlap in the two distributions of the target samples (i.e., red distribution) and the non-target samples (i.e., blue distribution). This makes the attack difficult in distinguishing the target and non-target samples, based on the value of the chosen neuron. Consequently, the attack success rate is moderate. On the other hand, when ε increases (Figs. 10b-10d), the distribution of the target samples shift to the right, meanwhile the distribution of the non-target samples shift to the left. The shifts result in a less overlap between the two distributions. As a result, when the privacy budget ε increases, the attack success rate of AMI increases. We observe this phenomenon in all three datasets.

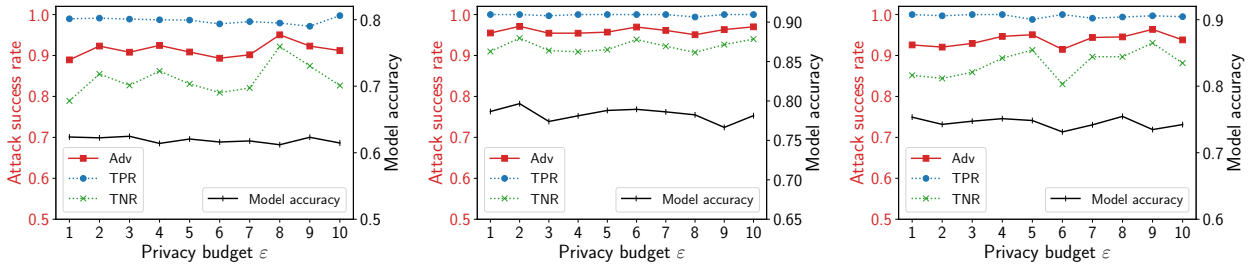


(a) $\varepsilon = 1$ (b) $\varepsilon = 3$ (c) $\varepsilon = 5$ (d) $\varepsilon = 10$

Figure 10: Histograms of the values of the chosen neuron when attacking against the BitRand mechanism with $\varepsilon = 1, 3, 5,$ and 10, respectively. The higher ε, the less overlapping distribution of the values of the chosen neuron, given the target samples and the non-target samples. This indicates the higher attack success rate of AMI. These data samples are obtained from the CelebA dataset.

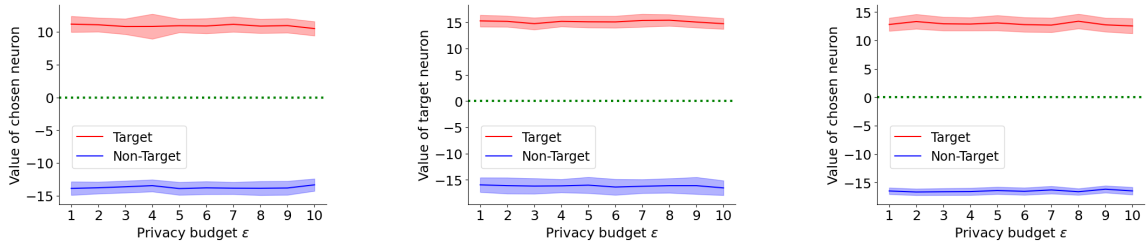
In addition to evaluating the attack under the BitRand mechanism in Section 5, Fig. 11 shows our attack performance under the OME mechanism [Lyu et al., 2020]. First, we observe the same phenomenon of OME as in [Arachchige et al., 2019, Lyu et al., 2020] in which the model accuracy does not change much for $\varepsilon \in [1, 10]$. Second, our attack maintains a severe success rate of about 90% for CelebA and CIFAR-10. More importantly, the attack success rate reaches more than 95% for ImageNet. These results demonstrate that our AMI attack remains very effective even with low privacy budget ε when the training set $\mathcal{D}$ is protected by the OME mechanism.

Fig. 12 shows the certified guarantee of success for the CelebA, ImageNet, and CIFAR-10 datasets when the LDP-preserving OME mechanism is used. We can derive a certified guarantee of success for our AMI attack given $\varepsilon \geq 1$ with a small broken



(a) CelebA (b) ImageNet (c) CIFAR-10

Figure 11: Attack success rate of AMI under the OME mechanism on CelebA, ImageNet, and CIFAR-10 datasets. The success rate is represented via the advantage (Adv), true positive rate (TPR), and true negative rate (TNR) according to Eq. 3. The baseline of random guessing is 0.5. The model accuracy illustrates the utility loss of the data when using LDP.



(a) CelebA (b) ImageNet (c) CIFAR-10

Figure 12: Certified guarantee of success for $\varepsilon \in [1, 10]$ in OME [Lyu et al., 2020]: Expectation (solid lines), upper-bound, and lower-bound (shaded areas surrounding the expectation) of the values of the chosen neuron.

probability 10^{-8}. This result is consistent with our attack success rate reported in Fig. 11. In addition to the unaffected model accuracy and attack success rate, the effect of the privacy budget ε is modest in the certified guarantee of success, given $\varepsilon \in [1, 10]$ used in OME.

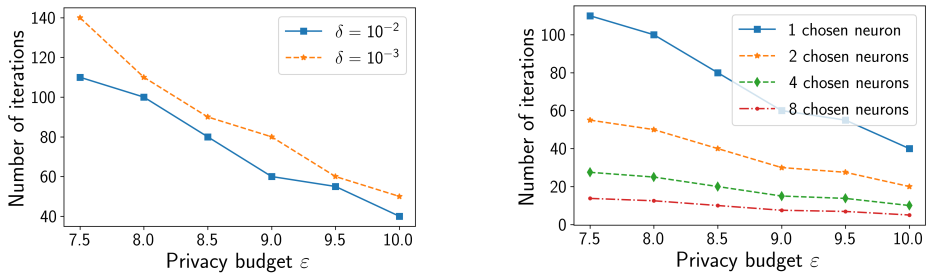
E NOISY GRADIENTS WITH DPSGD

Aside from LDP where clients perturb their own local training data $\mathcal{D}$ before computing the gradients, another method is to let clients add DP noise to their gradients using DPSGD [Abadi et al., 2016] before sending them to the server, hindering the attacker from knowing the true value of the chosen neuron's gradient g_t. Nevertheless, recent work suggests that using DPSGD makes it impossible to train models with reasonable accuracy for datasets like CIFAR-10 or ImageNet, even in a non-distributed setting [Boenisch et al., 2021, Tramèr and Boneh, 2021].

Moreover, even when DPSGD is used to add noise to the gradients, we can leverage the fact that the FL training is done in multiple iterations to circumvent this DP noise. In [Abadi et al., 2016], the DP noise is sampled from a zero-mean Gaussian distribution with a standard deviation of $\sigma = \frac{\sqrt{2\ln(1.25/\delta)}}{\varepsilon}$, where ε is the privacy budget and δ is a broken probability. However, as the noise is zero-mean, averaging the noise samples over multiple iterations will cancel out the noise and reveal the true value of the gradients. This is also referred to as the privacy composition problem in DP [Dwork et al., 2014].

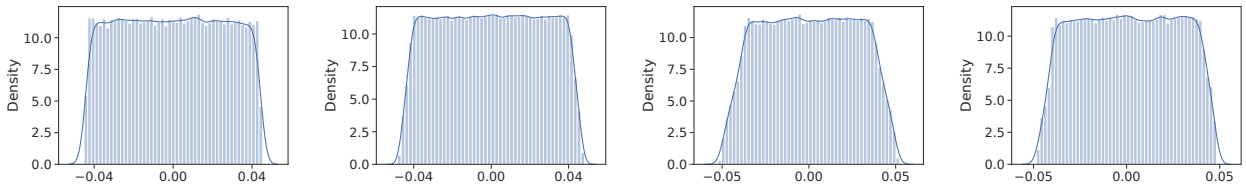
Specifically, denoting $g_t^{(i)'} = g_t + z_i$ (where $z_i \sim \mathcal{N}(0, \sigma^2\mathbf{I})$) as the noisy gradient of the chosen neuron at iteration i, the server can obtain the true gradient g_t by averaging over multiple iterations, i.e., $g_t = \frac{1}{P}\sum_{i=1}^{P} g_t^{(i)'}$. From this g_t, the attacker can determine whether the target sample was used in at least one of those iterations, following the same principle in Section 3. Fig. 13a shows the number of iterations needed to eliminate the DP noise. Previous work shows that training a neural network for CIFAR-10 up to a modest accuracy of 66.2% requires a privacy budget $\varepsilon \geq 7.53$ [Tramèr and Boneh, 2021], hence we evaluate with $\varepsilon \geq 7.5$. When $\varepsilon \geq 8$, we need less than 100 iterations.

To reduce the number of iterations, we can increase the number of chosen neurons in the second layer, and average the noisy gradients over all of the chosen neurons. Simply speaking, having K chosen neurons would reduce the number of iterations



(a) 1 chosen neuron (b) $\delta = 10^{-2}$

Figure 13: Number of iterations needed to eliminate DP noises. 13a shows the results for 1 chosen neuron, while 13b varies the number of chosen neurons and fixes $\delta = 10^{-2}$.



(a) Normal weights (b) $\varepsilon = 2$ (c) $\varepsilon = 3$ (d) $\varepsilon = 5$

Figure 14: Histograms of the distribution of normal weights and malicious weights. 14a shows the normal weights, while 14b, 14c, and 14d show the malicious weights when attacking under BitRand with $\varepsilon = 2, 3$, and 5, respectively. The solid blue line is the kernel density estimation (KDE).

by K-fold. Fig. 13b shows the number of iterations needed to eliminate the DP noise with multiple neurons. As can be seen, with only 4 chosen neurons, we only need less than 60 iterations to cancel out the DP noise at $\varepsilon = 7.5$. With 8 chosen neurons, the noise can be canceled out within 10 iterations.