More Than Enough Is Too Much: Adaptive Defenses Against Gradient Leakage in Production Federated Learning

Fei Wang, *Graduate Student Member, IEEE*, Ethan Hugh, and Baochun Li, *Fellow, IEEE*

***Abstract*— With increasing concerns on privacy leakage from gradients, various attack mechanisms emerged to recover private data from gradients, which challenged the primary advantage of privacy protection in federated learning. However, we cast doubt upon the real impact of these gradient leakage attacks on production federated learning systems. By taking away several impractical assumptions that the literature has made, we find that these attacks pose a limited degree of threat to the privacy of raw data. In this paper, through a comprehensive evaluation of existing gradient leakage attacks in a federated learning system with practical assumptions, we have systematically analyzed their effectiveness under a wide range of configurations. We first present key priors required to make the attack possible or stronger, such as a narrow distribution of initial model weights, as well as inversion at early stages of training. We then propose a new lightweight defense mechanism that provides *sufficient* and *self-adaptive* protection against time-varying levels of the privacy leakage risk throughout the federated learning process. Our proposed defense, called OUTPOST, selectively adds Gaussian noise to gradients at each update iteration according to the Fisher information matrix, where the level of noise is determined by the privacy leakage risk quantified by the spread of model weights at each layer. To limit the computation overhead and training performance degradation, OUTPOST only performs perturbation with iteration-based decay. Our experimental results demonstrate that OUTPOST can achieve a much better tradeoff than the state-of-the-art with respect to convergence performance, computational overhead, and protection against gradient leakage attacks.**

***Index Terms*— Federated learning, gradient leakage attack, data reconstruction, gradient perturbation.**

I. INTRODUCTION

AS AN EMERGING distributed machine learning paradigm, federated learning (FL) allows clients to train machine learning models collaboratively with private data, without transmitting them to the server. Though federated learning is celebrated as a privacy-preserving paradigm of training machine learning models, it was pointed out in the recent literature [1] that sharing gradients with an honest-but-curious server may lead to the potential reconstruction of raw private data, such as images and texts, used in the training process. The discovery of this new attack, known as *Deep Leakage from Gradients (DLG)*, has stimulated a new line of research to improve the attack efficiency [2], [3], [4] and to provide stronger defenses against known DLG-family attacks [5], [6] as well.

As existing studies have made significant efforts to indicate that federated learning is vulnerable to gradient leakage attacks from a malicious participant or eavesdropper, a lightweight defense that provides adequate privacy protection with guaranteed training accuracy is sought by recent work to prevent this attack [6]. However, before designing for even more efficient and effective defense mechanisms, we begin to have second thoughts on how severe the threat is in practice, even without any defense mechanisms in place. Existing works focused on reconstructing raw data from known gradients or model weights in ideal settings, rather than considering practical settings in production federated learning.

As its name suggests, DLG proved that sharing gradients has the potential of leaking private data. However, when this attack was first proposed and later improved, most works in the literature considered sharing model updates as equivalent to sharing gradients. In production FL, however, multiple epochs are used routinely, and gradients are only accessible locally in a single step of gradient descent. No gradient — in its strict, original connotation — is transmitted to the server at all. Instead, only model updates — the *delta* between local models and the server's global model in the preceding round — are transmitted from clients to the server. Yet, to the best of our knowledge, very little is known on the effectiveness of gradient leakage attacks in practical contexts in production federated learning. It was shown [7] that gradients can be calculated from model updates with a known learning rate; but with multiple epochs, we find that this calculation is far from accurate. Only [4] and [6] considered the possibility of reconstructing raw data with model updates directly, without estimating the gradients.

Even with the assumption of direct gradient sharing, existing works have mainly validated the efficiency when reconstructing one or multiple images (using a larger batch size) in full gradient descent, i.e., merely one local step of Stochastic Gradient Descent (SGD). None of them has shown convincing evidence that reconstructed images are recognizable by humans under the standard settings of production federated learning, where clients perform more local computation and

Manuscript received 15 December 2023; accepted 29 February 2024; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor Y. Zhang. *(Corresponding author: Fei Wang.)*

The authors are with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 1A1, Canada (e-mail: silviafey.wang@utoronto.ca).

Digital Object Identifier 10.1109/TNET.2024.3377655

less communication (i.e., multiple update steps on a local model) [8]. Moreover, existing attacks neglect the change of model status through FL training and tend to use untrained neural networks that are explicitly initialized with weights of a wide distribution for deriving gradients, which we have found makes the model and shared gradients fundamentally more vulnerable. Our empirical results disclose very limited privacy leakage even when gradients are shared, not to mention the privacy leakage with model updates, *delta*, only.

Existing defenses in the literature were designed explicitly for gradient leakage attacks [5], [6], and evaluated their performance with respect to privacy metrics (such as the peak signal-to-noise ratio) and utility metrics (such as the validation accuracy of the global model). However, they failed to show the feasibility of data reconstruction by the DLG attack in the first place before any defense is even applied, especially in the context of production FL [9] with federated averaging (FedAvg). Given that gradient leakage attacks are much weaker as we will discover in this paper, we argue that a new defense mechanism against such attacks can be much more lightweight, without introducing extra overhead or incurring the risk of sacrificing the utility of the global model after training completes.

Inspired by our empirical observations that models with weights from a narrower distribution and more local SGD update steps will effectively make potential attacks weaker, we propose a new defense mechanism, called OUTPOST, that provides *sufficient* and *self-adaptive* protection throughout the federated learning process against time-varying levels of privacy leakage risks. As its highlight, OUTPOST is designed to apply *biased perturbation* to gradients based on how spread out and how informative model weights are at different local update steps. Specifically, OUTPOST uses a probability threshold to decide whether we perform a perturbation to the gradients at the current step or not; and to limit the computation overhead, such a threshold decays as local update steps progress over time. When performing the perturbation, OUTPOST first evaluates privacy leakage risks of the current local model by the variance statistics of the model weights at each layer, and then adds Gaussian noise to each layer of the gradients of the current step based on the Fisher information matrix, whose range is decided by the quantified privacy leakage risks.

We have evaluated OUTPOST in comparison to four state-of-the-art defense mechanisms in the literature, against two gradient leakage attacks under both production FL settings and a simplistic, yet unrealistic, FL scenario where the attack is the most effective.[1] We seek to evaluate both *utility metrics* — regarding both the wall-clock time needed to converge and the converged accuracy — and *privacy metrics*, which shows the effectiveness of the defenses against gradient leakage attacks in the worst case. With two image classification datasets and the same `LeNet` neural network architecture used in the literature [1], our experimental results have demonstrated convincing evidence that OUTPOST can achieve a much better accuracy compared with the state-of-the-art, incurs a much smaller amount of computational overhead, while effectively providing a sufficient level of protection against DLG attacks when evaluated using common privacy metrics in the literature.

Furthermore, we have extended our investigation into the effectiveness of state-of-the-art gradient leakage attacks and defenses in broader contexts. *First*, we analyze deeper neural network models, particularly those in the ResNet family. Our findings reveal that the success of attacks, such as those proposed in [2] and [4], heavily relies on explicitly configuring the client's model into the evaluation state during implementation. Such an explicit configuration of the model state does not reflect reality, and may be in place in order to bypass the batch normalization layers commonly present in neural network models for vision tasks. Even in the simplistic federated learning settings we experimented with, which favor the attacker, OUTPOST can still provide substantial protection on model updates, with minimal impact on model training and computational overhead, achieving the best trade-off compared to other defenses. *Second*, we examine gradient leakage attacks [10], [11] under the assumption of a dishonest and malicious server who disregards the federated learning protocol and modifies the global model to tempt the client into disclosing its private data in gradients. We find that, similar to honest-but-curious attacks, malicious attacks are only effective in highly unrealistic scenarios that do not occur in practice. Even in such an adverse situation that exposes clients to much greater risks, OUTPOST proves to be highly effective.

The remainder of this paper is organized as follows. In Section II, we explain the underlying mechanism of optimization-based gradient leakage attacks. In Section III, we review different constructions of gradient leakage attacks and re-evaluate the validity of assumptions made by the literature. We then describe the details of our proposed defense, OUTPOST, including the design and convergence analysis, in Section IV. In Section V, we provide a thorough empirical evaluation of our proposed defense against gradient leakage attacks, comparing it with state-of-the-art defense mechanisms. In Section VI, we present our surprising findings on gradient leakage attacks particularly [2] on ResNet models, after correcting the implementation in related works to align with real-world federated learning. In Section VII, we evaluate gradient leakage attacks with a malicious server, particularly the Fishing attack proposed by [11], and demonstrate the robustness and reliability of OUTPOST against these more advanced attacks. In Section VIII, we discuss prior research on gradient leakage attacks in federated learning, considering both the assumption of an honest-but-curious server and a malicious server, as well as defenses that are specifically designed to mitigate gradient leakage attacks. Finally, we summarize and conclude the paper in Section IX.

II. PRELIMINARIES

A. Gradient Leakage Attacks

Recent research has discovered that, by simply exchanging gradients of neural network models rather than raw data among multiple clients, privacy leakage still cannot be prevented in

[1]Our implementation is available as an open-source git repository at https://github.com/TL-System/plato/tree/main/examples/gradient_leakage_attacks.

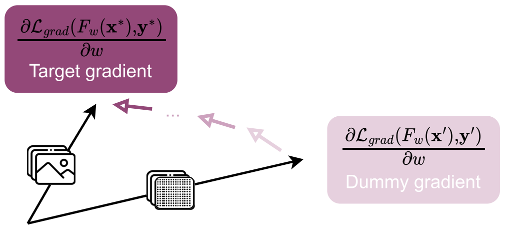

$\frac{\partial \mathcal{L}_{grad}(F_w(\mathbf{x}^*),\mathbf{y}^*)}{\partial w}$
Target gradient
$\frac{\partial \mathcal{L}_{grad}(F_w(\mathbf{x}'),\mathbf{y}')}{\partial w}$
Dummy gradient

Fig. 1. Matching the dummy gradient with the target gradient.

distributed machine learning. An honest-but-curious server can recover the private data from the obtained gradients through an optimization process [1], [2].

Essentially, DLG and other existing gradient inversion attacks tried to solve an optimization problem. With the given target gradient $\nabla^* : \nabla_t^k = \frac{\partial \mathcal{L}_{\text{grad}}(F_w(\mathbf{x}^*),\mathbf{y}^*)}{\partial w}$ received from a participant k at a certain step t, the attacker can steal the inputs with labels $(\mathbf{x}_t^k, \mathbf{y}_t^k)$ such as an image in pixels or a sentence in tokens in this training step. To do so, the attacker first generates random (dummy) inputs with labels $(\mathbf{x}'_0, \mathbf{y}'_0)$ of the same size as the target data. The attacker then (1) derives dummy gradient $\nabla'_0 = \frac{\partial \mathcal{L}_{\text{grad}}(F_w(\mathbf{x}'_0),\mathbf{y}'_0)}{\partial w}$ *w.r.t.* the model weights from the dummy data by feeding the dummy data into the machine learning model F_w shared between the participants; (2) updates the dummy data $(\mathbf{x}'_1, \mathbf{y}'_1)$ by gradient-based methods, in which the loss function is the distance between the dummy gradient and the given gradient ∇^*. By iterating these steps, the dummy data $(\mathbf{x}'_i, \mathbf{y}'_i)$ will move closer to the target training data as the dummy gradient matches the given gradient based on the following objective:

$$\mathbf{x}'^*, \mathbf{y}'^* = \underset{\mathbf{x}',\mathbf{y}'}{\arg\min}\, \mathbb{D}(\nabla', \nabla^*). \tag{1}$$

The distance $\mathbb{D}(\nabla', \nabla^*)$ is a function differentiable *w.r.t.* the dummy data, which can be the L2 distance $\|\nabla' - \nabla^*\|^2$ [1] or the cosine distance $1 - \frac{\langle \nabla', \nabla^* \rangle}{\|\nabla'\|\|\nabla^*\|}$ [2]. We illustrate the process that DLG attacks use in Fig. 1. More comprehensive objective functions incorporating regularization terms have been designed by some existing works to make the reconstructed images less noisy [2].

Existing gradient inversion attacks, including [1], [2], [3], [4], [7], and [12], explored the space of this attack in a wider set of circumstances, such as reconstructing multiple images and allowing multiple training steps. However, none of them considered these settings in the context of production FL. We now proceed to systematically evaluate the effectiveness of these attacks, considering both the required assumptions and applicable coverage, in the context of production FL.

B. Adversary Models

Objectives. Gradient leakage attacks in federated learning aim to reconstruct private training data from participating clients by analyzing the model updates exchanged during the training process.

Knowledge Assumptions. The adversary is assumed to possess knowledge of the global model at each communication round, including its structures and parameters, as well as

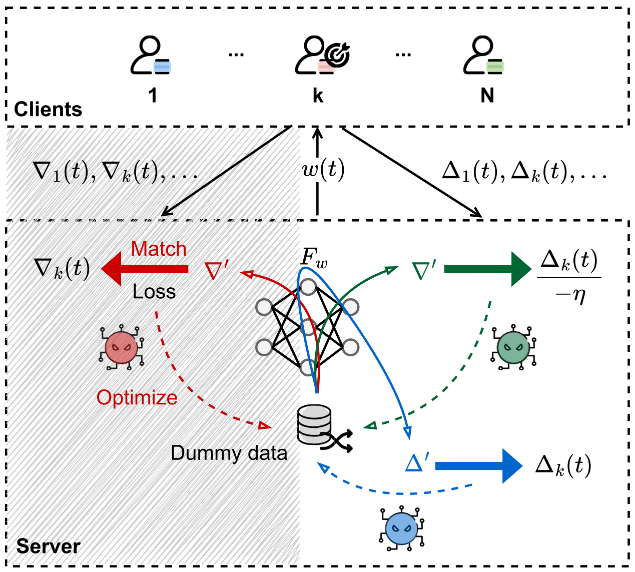


Fig. 2. Three different ways of performing data reconstruction from deep leakage in federated learning: [Red] matching the dummy gradient with the target gradient; [Green] matching the dummy gradient with the approximated gradient converted from the model update; [Blue] matching the dummy model update with the model update directly. The shadow area indicates that matching dummy gradients directly from gradients is not possible in production FL, as local gradients will not be accessible by the server.

the model updates transmitted from the selected client upon completing local training. Typically, the federated learning server is considered the ideal adversary for gradient leakage attacks.

Capabilities. The capabilities of the server acting as the adversary may vary depending on two different assumptions about the server: the honest-but-curious server and the dishonest server. An honest-but-curious server adheres to the federated learning protocol but may attempt to learn information about the clients' private data by analyzing the model updates. Conversely, a dishonest server manipulates the global model distributed to clients in an attempt to coax them into revealing their private data through gradients, disregarding the federated learning protocol entirely. Therefore, while the honest-but-curious server has limited capability compared to the dishonest server, its activities are less suspicious from the clients' perspective.

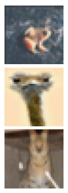
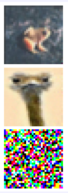
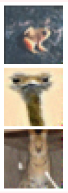
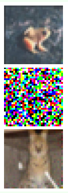
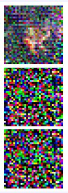
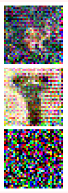
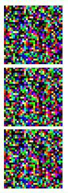
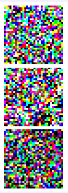
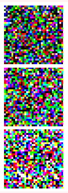
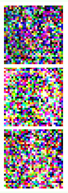
III. Re-Evaluating Gradient Leakage Attacks in Production Federated Learning

In this section, we first show that existing gradient inversion attacks do not work effectively in production FL, as some of their implicit assumptions fail to hold.

A. Assumptions: Re-Evaluating Their Validity in Production FL

Gradients are not shared directly with the server. Existing gradient inversion attacks in the literature (*e.g.*, [1], [3], [7], [12]) contained a technical misconception that model updates (delta Δ_t^k) are equivalent to gradients ∇_t^k. They assumed that a client will send either a gradient computed from a single local training step or an average gradient over multiple local training

TABLE I
RECONSTRUCTING A SINGLE IMAGE USING THE DLG ATTACK, WITH DIFFERENT MODEL INITIALIZATION METHODS OR TRAINING STAGES

Ground truth	**Untrained network with explicit initialization**			**Untrained network with default PyTorch initialization**			**Trained network pre-trained with the same data**		
	RS1	RS2	RS3	RS1	RS2	RS3	RS1	RS2	RS3
									

steps (such as multiple epochs or multiple mini-batches) to the server. However, in production FL using FedAvg [9], computation over multiple local training steps are typically used, and gradients in each step as intermediate outputs are not visible to the other clients or to the server.

Neural network models are not initialized explicitly before training. Existing gradient inversion attacks in the literature assumed that the neural network model used by the attack was untrained. Yet, we discovered that initial weights in the neural network model substantially affect the difficulty of performing the gradient inversion attack. As a rule of thumb, to launch a successful attack, gradients need to have large magnitudes and contain the most information to recover the data. In fact, we discovered an explicit weight initialization step in the source code of both the DLG algorithm [1] and other alternatives derived from DLG [2], [3], [12]. Such an initialization step uses initial random weights and biases with a broader range of values using an uniform distribution, compared to the range of values that is initialized by `PyTorch` [13]. This initialization step produces a neural network model that is more naive, which conveys more informative gradients to the attacker.

In production FL, training a pre-trained global model is more common than training from scratch. Even if we intend to train a model from scratch, the model will evolve into different states over multiple communication rounds. The gradient will be closer to zero over an appropriate training process as the loss is approaching the minimum, leading to a more challenging attack on a trained model. Unless an attacker performs gradient inversion at the beginning of a federated learning session with a naïve model, there is very limited information about the raw data carried by the gradients.

To compare the capability of image reconstruction under different scenarios of model initialization and training stages, we ran the DLG attack on arbitrary images in the `CIFAR-10` dataset and show the converged results in Table I, where different random seeds (RS1, RS2, and RS3) were used to generate dummy data in multiple trials. By comparing the results with the same random seeds, we can observe that there is a significant amount of noise in the recovered image when the shared gradients come from an untrained network initialized by `PyTorch`. When a pre-trained network model is used, we can hardly recognize the recovered images. Even for the best cases where shared gradients come from an untrained model with random weights explicitly initialized with a wider distribution, the attack is challenging to realize without using a particularly delicate choice of random seeds for generating the dummy data.

There are multiple update steps (across batches and epoches) in local training. With FedAvg [9], a client selected by the server performs $\tau = E \cdot n/B$ steps of gradient descent in each communication round, where n is the number of data samples at the client, E denotes the number of local training epochs in each communication round over the local dataset, and B is the local mini-batch size in each epoch. However, existing attacks are only capable of reconstructing one or multiple images when they are used as a full batch for training in a single gradient descent step (i.e., $E = 1$ and $B = n \geq 1$). In production FL, however, multiple images are used across multiple update steps (i.e., $E \geq 1, n \gg 1, B < n$) in training, with each of these steps corresponding to a mini-batch in an epoch. The attacker's ability to reconstruct images using gradient matching may be substantially affected as the local model evolves throughout multiple update steps, since the attacker only has access to the global model at the beginning of each communication round, and the assumption of having only a single gradient descent step is no longer valid.

Label estimation is more challenging with non-i.i.d. data distributions. It was shown that labels of a single image [3] or even of a batch of images [4] can be estimated from the averaged gradients. However, the accuracy of such label estimation will be reduced if multiple images belong to one label [14]. In production FL, the distribution of local data across different clients is non-i.i.d. (independent and identically distributed), and data samples on the same client will correspond to fewer labels as a result of such a non-i.i.d. distribution. Under these circumstances, the accuracy of label estimation may be significantly affected.

B. More Sophisticated DLG Attacks

Approximating gradients from updates. As we have argued, instead of gradients, *model updates* are transmitted from clients to the server in production FL. When training a neural network in federated learning, each client tries to optimize the network parameters θ using a loss function $\mathcal{L}_\theta$ on local training data. With a model $F_{w_t^k}$, the gradient ∇_t^k at a local training step can be evaluated by $\nabla \mathcal{L}_\theta(F_{w_t^k})$ and the

TABLE II
ASSUMPTIONS AND SETTINGS: PRODUCTION FL VS. WHAT WAS USED IN EXISTING GRADIENT LEAKAGE ATTACKS

Parameters/Settings	**What is practical in production FL**	**Used in previous attacks**						
Network model state	Model in an arbitrary communication round	Untrained model with explicit initialization						
The number of epochs E, the number of data samples n, and batch size B	Multiple local steps with $E \geq 1, n \gg 1, B < n$	$\xrightarrow{E\text{:stronger}}$		$\xrightarrow{n\text{:stronger}}$		$\xrightarrow{B\text{:stronger}}$		Examples
		1	≥ 1	1	≥ 1	$= n$	$\leq n$	
		✓		✓		✓		DLG [1]
		✓		✓		✓		iDLG [3]
		✓			✓		✓	csDLG [2]
			✓		✓	✓		
		✓			✓	✓		GradInversion [4]
Data heterogeneity	non-i.i.d. distribution	Not fully investigated						
Gradient descent optimizer	Incorporating learning rate, momentum, weight decay, learning rate schedule	Fixed learning rate only						
Other prior knowledge	Not accessible for the server	Batch normalization statistics [2], [4], private labels [2], etc.						

new model weights can be updated with a learning rate η as $w^k_{t+1} = w^k_t - \eta\nabla^k_t$. With τ^k steps of local training completed, the model weights at the end of this communication round can be expressed as $w^k(t+1) : w^k_{t+\tau^k}$. After multiple local steps of training in a communication, the server updates the global model by the local model updates (delta) as $w(t+1) = w(t) + \sum_k \frac{n^k}{n}\Delta^k(t, t+1)$, where $\Delta^k(t, t+1) = w^k(t+1) - w^k(t)$.

Existing work in the literature [7] and [15] considered gradients and model updates as mathematically equivalent, and used Eq. 2 to convert the delta $\Delta^k(t, t+1)$ to gradients before performing gradient matching. Note that, for such an approximation to be accurate, the fixed learning rate used at the victim client (or shared between all the clients) must be known by the attacker.

$$\nabla^k(t, t+1) = -\frac{\Delta^k(t, t+1)}{\eta} \tag{2}$$

Even with a known learning rate, such an approximation only works effectively in the simplest case where only the learning rate is used in the local gradient descent steps. In production FL, however, there are a number of other factors, such as momentum, weight decay, and learning rate schedules, that are used routinely as best practices when training neural networks. Gradient approximation from model updates will fail in these more realistic settings as the computation is not fully reversible for the attacker [16].

Matching deltas from updates. As an alternative to approximating gradients from updates, Geiping et al. [2] directly conducted the matching process over model updates (or updated weights), with a similar matching mechanism to conventional gradient matching. The attacker directly matches its dummy weights or weight delta with the absolute weights or model updates.

For delta matching to be effective, the server requires a series of prior knowledge to realize the same gradient descent process using the dummy data instead, including the number of images to be reconstructed n, the batch size B, the learning rate η, and other gradient descent factors such as weight decay and momentum. What's more, the effectiveness of the delta matching process relies heavily on the assumption that attackers know the data labels [2]. Without known labels as prior, data reconstruction will become harder. But as we have pointed out earlier in this section, label recovery often fails in production FL, where client data distributions are usually non-i.i.d.

Summary. Based on our analysis so far, the most feasible way to perform this kind of attack in federated learning is by matching deltas from updates. Therefore, when we devise our new defense and conduct experiments, we only consider matching deltas from updates, which does not imply that our defense is ineffective against attacks that approximate gradients from updates. We show in Table II the stark differences between assumptions in the gradient leakage literature and practical settings in production FL.

IV. OUTPOST: OUR LIGHTWEIGHT DEFENSE

Thanks to the inherent resilience against gradient leakage attacks in production FL, such as multiple local iterations and more complex gradient descent optimizers, it becomes feasible to design a simple, lightweight, yet effective gradient protection mechanism as a proactive defense, without sacrificing the training performance in FL sessions. In this section, we propose a new defense mechanism, called OUTPOST, to achieve these objectives.

A. OUTPOST: *Mechanism Design*

Privacy leakage risk. The scale of the initial distribution has a significant effect on both the outcome of the optimization procedure and on the ability of the neural network model to generalize [17]. To elaborate what we have found in Section III, the default model initialization in `PyTorch` uses the Kaiming Uniform method [18] for both linear and convolutional layers. The weights (and biases) of each layer are values drawn randomly from a uniform distribution of $\mathcal{U}(-\sqrt{\frac{1}{\text{in\_features}}}, \sqrt{\frac{1}{\text{in\_features}}})$, where in_features is the size of the previous layer. However, the explicit model weight initialization in those existing attacks for generating the gradients uses a distribution of $\mathcal{U}(-0.5, 0.5)$, which is of a much larger scale. This phenomenon gives us insights that the magnitude of neural network weights can intuitively reveal the privacy leakage risks of the corresponding gradients. Therefore, we employ the variance statistics of the neural network weights as a way to quantify the privacy leakage risks per layer. We denote the scalar variance of all weights at layer d as $\operatorname{Var}\left[w_d\right]$. Then, the privacy leakage risk at the d-th layer of the client k's local model at communication round t iteration i can be expressed as $r_{t+i_d}^k=\operatorname{Var}\left[w_{t+i_d}^k\right]$.

Selective perturbation. Existing defenses based on gradient compression or pruning techniques are designed for pruning gradients with small magnitudes to zeros. As these gradients have insignificant effects when weights are updated, the accuracy and convergence speed are both preserved with these techniques in place. However, we doubt the effectiveness of this method in the context of gradient leakage attacks, as the remaining gradients still carry the primary information for data reconstruction. With this insight, we suspect that perturbing those insignificant gradients cannot sufficiently protect privacy information from recovering. Therefore, we tend to also perturb gradients of which the model parameters contain more important information.

We choose the diagonal of the Fisher information matrix (FIM) to estimate how important a certain parameter is. FIM [19] is strongly related to the Hessian Matrix, indicating the curvature of the loss function for efficient optimization. The Fisher information of the model can be expressed as

$$I_\theta=E_{p(x \mid \theta)}\left[\nabla_\theta \log p(x \mid \theta) \nabla_\theta \log p(x \mid \theta)^T\right], \quad (3)$$

where $\log p(x \mid \theta)$ is a the log-likelihood function given by the model with parameter θ.

However, it is infeasible to directly use FIM or Hessian information in the context of deep learning as the likelihood is intractable. We use the empirical Fisher information matrix instead, which crudely approximates the FIM, and is often used to make computation easier [20]. We express the empirical Fisher as

$$\widehat{\mathbf{F}}=\frac{1}{n/B} \sum_{i=1}^{n/B} \nabla \mathcal{L}_{\theta i} \cdot \nabla \mathcal{L}_{\theta i}^{\top}, \quad (4)$$

where $\nabla \mathcal{L}_{\theta i}$ denotes the gradient w.r.t. the i-th samples at the given point, and $\cdot$ is the outer product of individual gradients.

In the given iteration, we can compute the empirical Fisher for each model parameter based on average gradients over this batch of data. When performing the perturbation, we only choose gradients with the $\varphi\%$ highest values of empirical Fisher in each layer to add noise, at a level determined by the privacy leakage risk.

Perturbation with iteration-based decay. As we have found that model updates shared after multiple local training steps have a decreased risk of leaking information about the raw training data, we propose to devise a schedule in OUTPOST to adaptively perturb gradients in different local training steps. Unlike existing defenses that treat gradients in each local step equally, we introduce a bias on the probability of perturbating gradients according to the number of local iterations. Specifically, in each local training iteration i, the probability of a client applying perturbation on its gradients averaged over a mini-batch at the current iteration is determined by $\operatorname{Pr}=1/(1+\beta \cdot i)$, where β is the hyperparameter to control how fast the probability decay is as the number of iterations grows.

The layer-based perturbation consists of two steps: (1) compressing gradients by their magnitudes: $\rho\%$ of the smallest gradients in each layer l are pruned to zeros; (2) adding noise to gradients by their privacy leakage risks: noise is added to each layer l of the gradients following the Gaussian distribution $\mathcal{N}(0,\left(\lambda r_{t+i_d}^k\right)^2)$, where λ controls the range of variance. With such a design, OUTPOST is a simple and *self-adaptive* defense mechanism against time-varying levels of privacy leakage risks, yet without introducing a substantial amount of computation overhead. The overall random perturbation mechanism in OUTPOST is shown as Algorithm 1.

Algorithm 1 FedAvg Local Training at Client k With OUTPOST

```
Input: Broadcast the global model $F_w t$ with weights $w(t)$ of the current communication round $t$; learning rate $\eta$; the number of local data samples $n$; the number of epochs $E$; batch size $B$; Initial iteration number as 0
Output: Local model update $\Delta^k(t,t+1)$
 1: Set local model weights same as the global one $w^k(t)=w(t)$
 2: for each epoch 1 to $E$ do
 3:     for each batch 1 to $n/B$ do
 4:         Iteration $i=i+1$
 5:         Compute loss and derive gradient at the current iteration $\nabla_{t+i}^k=\nabla \mathcal{L}_\theta(F_{w_{t+i}^k})$
 6:         if a random value $(\in[0,1)) \leq \operatorname{Pr}=1/(1+\beta \cdot i)$ or $i=1$ then
 7:             Evaluate privacy leakage risks based on current model per layer by $r_{t+i_d}^k=\operatorname{Var}\left[w_{t+i_d}^k\right]$
 8:             Perform pruning with threshold $\rho\%$ and update perturbed gradient to $\tilde{\nabla}_{t+i}^k$
 9:             Add noise to the selected $\varphi\%$ gradient at each layer and update the perturbed gradient as $\tilde{\nabla}_{t+i}^k=\tilde{\nabla}_{t+i}^k+m$ where $m \in \mathcal{N}(0,\left(\lambda r_{t+i_d}^k\right)^2)$
10:         else
11:             Update perturbed gradient as $\tilde{\nabla}_{t+i}^k=\nabla_{t+i}^k$
12:         end if
13:         Update local model weights with the perturbed gradient and learning rate as $w^{t+i}=w^{t+i-1}-\eta \tilde{\nabla}_{t+i}^k$
14:     end for
15: end for
16: Send the model update $\Delta^k(t,t+1)=w^k(t+1)-w^k(t)$ to the server
```

B. Convergence Guarantee With the FedAvg Algorithm

Our convergence analysis of FedAvg on non-i.i.d. data with the OUTPOST defense mechanism is based on the following assumptions. We use the same assumptions (Assumptions 1 to 5) as [8] on local objective functions $F_1,\cdots,F_N$ and partial device participation.

Assumption 1: $F_1,\cdots,F_N$ are all L-smooth: for all $\mathbf{v}$ and $\mathbf{w}$, $F_k(\mathbf{v}) \leq F_k(\mathbf{w})+(\mathbf{v}-\mathbf{w})^T \nabla F_k(\mathbf{w})+\frac{L}{2}\|\mathbf{v}-\mathbf{w}\|_2^2$.

Assumption 2: $F_1, \cdots, F_N$ are all μ-strongly convex: for all $\mathbf{v}$ and $\mathbf{w}$, $F_k(\mathbf{v}) \geq F_k(\mathbf{w}) + (\mathbf{v} - \mathbf{w})^T \nabla F_k(\mathbf{w}) + \frac{\mu}{2}\|\mathbf{v} - \mathbf{w}\|_2^2$

Assumption 3: Let ξ_t^k be sampled from the k-th device's local data uniformly at random. The variance of stochastic gradients in each device is bounded: $\mathbb{E}\left\|\nabla F_k\left(\mathbf{w}_t^k, \xi_t^k\right) - \nabla F_k\left(\mathbf{w}_t^k\right)\right\|^2 \leq \sigma_k^2$ for $k = 1, \cdots, N$.

Assumption 4: The expected squared norm of stochastic gradients is uniformly bounded, i.e., $\mathbb{E}\left\|\nabla F_k\left(\mathbf{w}_t^k, \xi_t^k\right)\right\|^2 \leq G^2$ for all $k = 1, \cdots, N$ and $t = 1, \cdots, T-1$

Assumption 5: Assume $\mathcal{S}_t$ contains a subset of K indices uniformly sampled from $[N]$ without replacement. Assume the data is balanced in the sense that $p_1 = \cdots = p_N = \frac{1}{N}$. The aggregation step of FedAvg performs $\mathbf{w}_t \longleftarrow \frac{N}{K}\sum_{k \in \mathcal{S}_t} p_k \mathbf{w}_t^k$.

We now derive new bounds for Assumption 3 with our defense. The expected norm of the distance between the perturbed gradients $\nabla F_k'\left(\boldsymbol{W}_t^k, \xi_t^k\right)$ and the original gradients $\nabla F_k\left(\boldsymbol{W}_t^k, \xi_t^k\right)$ of the whole model is the sum of that of every layer. Thus,

$$\begin{aligned}&\mathbb{E}\left\|\nabla F_k'\left(\boldsymbol{W}_t^k, \xi_t^k\right) - \nabla F_k\left(\boldsymbol{W}_t^k, \xi_t^k\right)\right\|^2 \\ =&\sum_{d=1}^{D}\mathbb{E}\left\|\nabla F_k'\left(\boldsymbol{w}_{dt}^k, \xi_t^k\right) - \nabla F_k\left(\boldsymbol{w}_{dt}^k, \xi_t^k\right)\right\|^2 \\ \leq&\left(\lambda r_d^k\right)^2 \cdot D, \end{aligned} \tag{5}$$

according to our perturbation noise distribution $\mathcal{N}(0, \left(\lambda r_d^k\right)^2)$, where $r_d^k \in [0,1]$ is the privacy risk of the d-th layer, and D is the total number of layers of the stochastic gradients.

By using the norm triangle inequality, we can bound the variance of stochastic gradients after perturbation in each device as

$$\begin{aligned}&\mathbb{E}\left\|\nabla F_k'\left(\boldsymbol{W}_t^k, \xi_t^k\right) - \nabla F_k\left(\boldsymbol{W}_t^k\right)\right\|^2 \\ \leq&\mathbb{E}\left\|\nabla F_k'\left(\boldsymbol{W}_t^k, \xi_t^k\right) - \nabla F_k\left(\boldsymbol{W}_t^k, \xi_t^k\right)\right\|^2 \\ &+\mathbb{E}\left\|\nabla F_k\left(\boldsymbol{W}_t^k, \xi_t^k\right) - \nabla F_k\left(\boldsymbol{W}_t^k\right)\right\|^2 \\ \leq&\left(\lambda r_d^k\right)^2 \cdot D + \sigma_k^2, \end{aligned} \tag{6}$$

in which we use Assumption 3 and Eq. 5.

Similarly, we can derive new bounds for Assumption 4 of the expected squared norm of stochastic gradients after pertuerbation in each device as $\mathbb{E}\left\|\nabla F_k'\left(\mathbf{w}_t^k, \xi_t^k\right)\right\|^2 \leq \left(\lambda r_d^k\right)^2 \cdot D + G^2$ for all $k = 1, \cdots, N$ and $t = 1, \cdots, T-1$.

Similar to [8], let F^* and F_k^* be the minimum values of F and F_k, respectively. The term $\Gamma = F^* - \sum_{k=1}^N p_k F_k^* > 0$ can be used for quantifying the degree of non-i.i.d. We then have the following convergence guarantee with FedAvg on non-i.i.d. data, and with the OUTPOST defense mechanism.

Theorem 1: Let Assumptions 1 to 5 hold and L, μ, σ_k, G be defined therein. Choose $\kappa = \frac{L}{\mu}$, $\gamma = \max\{8\kappa, E\}$ and the learning rate $\eta_t = \frac{2}{\mu(\gamma + t)}$. Then

$$\begin{aligned}&\mathbb{E}\left[F\left(\mathbf{w}_T\right)\right] - F^* \\ &\leq \frac{\kappa}{\gamma + T - 1}\left(\frac{2(B+C)}{\mu} + \frac{\mu\gamma}{2}\mathbb{E}\left\|\mathbf{w}_1 - \mathbf{w}^*\right\|^2\right),\end{aligned}$$

where

$$\begin{aligned}B =& \sum_{k=1}^{N} p_k^2(\sigma_k^2 + \left(\lambda r_d^k\right)^2 \cdot D) + 6L\Gamma + 8(E-1)^2(\left(\lambda r_d^k\right)^2 \\ &\cdot D + G^2), \\ C =& \frac{N-K}{N-1}\frac{4}{K}E^2(\left(\lambda r_d^k\right)^2 \cdot D + G^2).\end{aligned}$$

V. EXPERIMENTAL RESULTS

We are now ready to compare OUTPOST with state-of-the-art defense mechanisms in the literature, against different gradient leakage attacks and under a variety of experimental settings. All our experiments are conducted on a server with two Intel Xeon Silver 4210R CPUs and one NVIDIA RTX A4500 GPU with 20 GB CUDA memory.

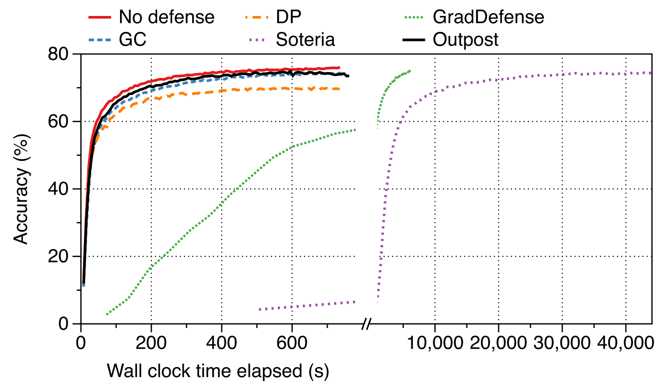
Defense and attack baselines. We compare OUTPOST with two state-of-the-art defense mechanisms: Soteria [5] and GradDefense (GD) [6], along with two commonly used defenses for general attacks in federated learning — gradient compression (GC) [1], which prunes small values in gradients; and differential privacy (DP) [21], which adds noise to gradients. We evaluate these defense mechanisms against two gradient leakage attacks: DLG [1] and csDLG [2]. When applying these attacks, we use the mechanism of matching deltas from updates, which is the most practical choice in production FL.

Datasets and models. We evaluate both gradient leakage attacks and defenses in PLATO,[2] an open-source research framework for federated learning. We show data reconstruction results over two image classification datasets: `EMNIST` and `CIFAR-10`. We use the same `LeNet` model evaluated in [1] and [2], which consists of 4 convolutional layers and 1 fully-connected layer.

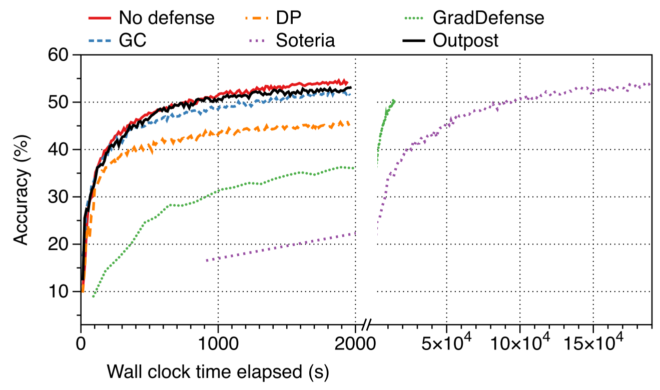
Evaluation metrics. To evaluate the effectiveness of the defense mechanisms, we use mean-square-error (MSE), structural similarity index measure (SSIM), and learned perceptual image patch similarity (LPIPS) as our metrics to measure the distance between the reconstructed image and raw image. To evaluate the impact of the defenses on the convergence performance of FL training, we show the accuracy of the global model on the validation dataset over multiple communication rounds using the FedAvg algorithm. To evaluate the computation overhead introduced by the defenses, we measure the wall-clock time averaged across multiple communication rounds.

Hyperparameter configurations. We evaluate OUTPOST with respect to two aspects: (1) *training performance*, where we examine how our defense affects the FL training time and the validation accuracy of the converged model in production FL settings; and (2) *defense effectiveness*, where we evaluate how effective OUTPOST is under settings where the gradient leakage attacks are as threatening as possible. For alternative defense mechanisms, we have the following configurations. For GC, we set the pruning rate of gradients to 80%; for DP, we use Laplacian noise and set the noise variance to 0.1;

[2]Available online at https://github.com/TL-System/plato.



(a) EMNIST



(b) CIFAR-10

Fig. 3. The impact of defenses on FL training.

for Soteria, we set the pruning rate of the fully connected layer's gradients to 50%; for GradDefense, we turn on the local clipping operation and use the same settings in their source code with 0.01 as the Gaussian noise variance; and for OUTPOST, we set our hyperparameters as $\lambda = 0.8, \varphi = 40, \beta = 0.1, \rho = 80$.

A. Evaluating the Training Performance

With respect to the training performance, for both datasets, data is non-i.i.d. distributed across 100 clients, each holding 1% of the total training samples (i.e., 1128 for EMNIST and 500 for CIFAR-10). Regarding the local epoch E and batch size B, we set them as $E = 5, B = 32$ for both datasets. We apply the SGD optimizer for local training and set the learning rate η to 0.01. In each communication round, the server randomly selects 10 clients out of 100 and aggregates their model updates. We terminate FedAvg training sessions with various defenses when the number of communication rounds reaches 100 on EMNIST and 150 CIFAR-10, where the global models converge.

With respect to the validation accuracy of the converged model and the elapsed wall-clock time, our results over EMNIST and CIFAR-10 have been shown in Fig. 3. If we examine the ultimate global model accuracy at the end of each training session, it can be observed that all the defenses incurred a certain amount of losses. DP affected the convergence performance the most, and only reached 91.5% and 83.1% of the validation accuracy without any defense, over EMNIST and CIFAR-10 datasets, respectively. This observation is consistent with our expectations based on these defense mechanisms. GC and Soteria both prune gradients with small magnitudes to zeros, which have insignificant effects on weights to be updated. Though GradDefense and OUTPOST also add noise to gradients, the level of noise is controlled with model sensitivity and model status, respectively. Overall, OUTPOST induces only 3.28% and 2.19% accuracy loss with 3.54% and 1.47% delay, over EMNIST and CIFAR-10 datasets, respectively.

When we examine the wall-clock time elapsed in the same number of communication rounds, it is apparent from Fig. 3 that GC, DP and OUTPOST as defenses did not introduce any significant delays due to the computation overhead, while GradDefense and Soteria extended the training session an order of magnitude longer than FedAvg without any defenses. This is because Soteria has to learn the perturbed data representation in each local SGD iteration based on data representation in the FC layer of the model trained after that iteration. Similarly, GradDefense needs to compute model sensitivity in each iteration based on which it adds Gaussian noise to selected slices of gradients. In contrast, OUTPOST is an order of magnitude less time-consuming than these state-of-the-art alternatives, thanks to its fast evaluation of the privacy leakage risk based on model status and its perturbation with iteration-based decay.

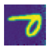
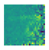
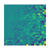
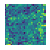
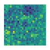
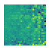
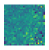
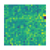
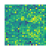
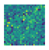
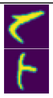
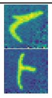
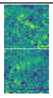
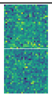
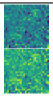
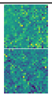
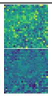
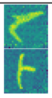
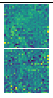
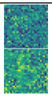
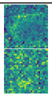
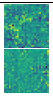
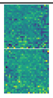
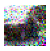
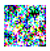
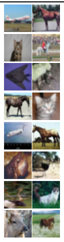
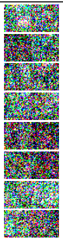
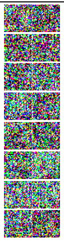
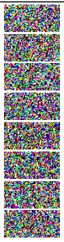
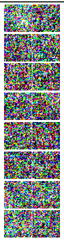
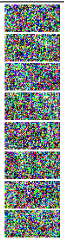
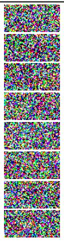
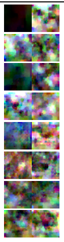
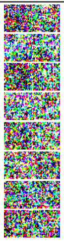
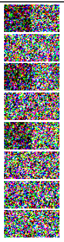
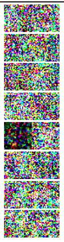
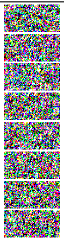
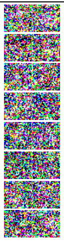
One may wonder why Soteria and GradDefense are not applied to model updates at the end of each communication round instead, since the notions of gradients and model updates are not clearly differentiated in their papers. This is due to the fact that, to compute the mask for pruning, Soteria needs detailed information such as the gradient of the loss with respect to the input batch data, which is only accessible in each SGD iteration. The slicing and perturbation methods in GradDefense are also gradient specific.

B. How Effective Are the Defenses?

For our defense evaluation, we make the attack as strong as possible by having only one client participate in the training, with the attack performed in the first round of training. The model is explicitly initialized with a wider distribution, and no momentum, weight decay, or learning rate scheduler is applied to the SGD optimizer. Although we've shown in Section III that this setting is completely unrealistic, we only use it to demonstrate the effectiveness of the defenses, and exaggerate the difference between various defense mechanisms in this extreme case. For both DLG and csDLG attacks, we applied the L-BFGS optimizer with 3000 iterations of reconstruction to guarantee convergence, and presented the worst defense result in 10 trials with different dummy data.

Table III shows the results from our experiments, evaluating a variety of metrics for each defense. The up and down arrows indicate the direction of better defense performance. We also put the reconstructed images along with ground truth images in the last row of each experiment scenario to show if they are recognizable by human eyes. Note that the order of reconstructed images may be inconsistent with the

TABLE III
THE EFFECTIVENESS OF VARIOUS DEFENSE MECHANISMS AGAINST DIFFERENT ATTACK BASELINES, DATASETS, HYPERPARAMETER CONFIGURATIONS

	DLG						**csDLG**					
	[Scenario 1] `EMNIST`: $E = 1, n = 1, B = 1$											
	No defense	GC	DP	Soteria	GD	OUTPOST	No defense	GC	DP	Soteria	GD	OUTPOST
MSE ↑	6.6e-3	13.96	113.63	95.19	32.57	77.05	2.6e-7	199.08	297.84	296.76	360.98	294.678
LPIPS ↑	7.1e-2	0.55	0.60	0.63	0.64	0.58	5.8e-7	0.60	0.66	0.63	0.64	0.68
SSIM ↓	0.99	0.30	0.19	3.3e-2	1.0e-2	0.13	1.00	0.32	6.5e-2	4.1e-2	1.7e-2	1.6e-2
												
	[Scenario 2] `EMNIST`: $E = 1, n = 2, B = 1$											
	No defense	GC	DP	Soteria	GD	OUTPOST	No defense	GC	DP	Soteria	GD	OUTPOST
MSE ↑	5.8e-2	69.87	319.30	16.07	25.50	75.92	0.14	336.05	1231.82	782.93	668.79	277.73
LPIPS ↑	0.31	0.61	0.66	0.70	0.65	0.67	0.40	0.60	0.65	0.59	0.63	0.66
SSIM ↓	0.77	3.7e-2	2.4e-2	4.7e-2	2.7e-2	6.4e-2	0.70	3.3e-2	1.4e-2	1.6e-2	3.4e-2	3.8e-2
												
	[Scenario 3] `CIFAR-10`: $E = 1, n = 1, B = 1$											
	No defense	GC	DP	Soteria	GD	OUTPOST	No defense	GC	DP	Soteria	GD	OUTPOST
MSE ↑	5.1e-2	7.83	34.08	25.91	11.46	13.10	5.9e-5	27.50	34.51	25.91	56.66	35.24
LPIPS ↑	0.53	0.77	0.77	0.76	0.74	0.77	1.8e-3	0.76	0.78	0.76	0.77	0.77
SSIM ↓	0.57	4.4e-2	3.6e-2	5.5e-2	2.2e-2	2.1e-2	0.99	2.6e-2	2.9e-2	5.5e-2	1.7e-2	3.4e-2
												
	[Scenario 4] `CIFAR-10`: $E = 1, n = 16, B = 16$											
	No defense	GC	DP	Soteria	GD	OUTPOST	No defense	GC	DP	Soteria	GD	OUTPOST
MSE ↑	1.37	4.82	19.92	2.85	4.04	15.85	6.2e-2	6.85	13.68	14.87	2.1e10	13.03
LPIPS ↑	0.67	0.69	0.71	0.70	0.69	0.70	0.51	0.69	0.71	0.69	0.71	0.70
SSIM ↓	2.9e-2	1.9e-2	1.5e-2	2.2e-2	2.2e-2	1.7e-2	0.30	4.5e-2	1.6e-2	2.7e-2	3.8e-2	3.1e-2
												

raw ones in Scenario 4, which is an inherent issue of the attacks.

Our previous argument — that gradient leakage attacks are not practical when there are multiple update steps in local training — can be supported by our results in the columns of attacks without defense in Table III, even though csDLG demonstrates a better data reconstruction capability in general. As shown in Scenario 2, the performance of attacks is heavily affected with merely two update steps each having a batch of one data sample as the color difference in reconstructed images is less clear compared to Scenario 1. Performing the attack on a larger batch size not only makes the attacks significantly slower, but also worse, with only one image out of 16 being reconstructed using csDLG without defense. Realistic FL settings use batch sizes that are many magnitudes greater than 16, meaning that attacks are unlikely to ever converge; and even if they were to converge, a successful reconstruction would be nearly impossible.

We notice that MSE is the most inconsistent metric and a higher MSE does not correlate to a reconstructed image containing fewer key features of ground truth. Most related works use this metric as a basis for their conclusions, which means we may need to revisit some of their claims. Compared to `CIFAR-10` images with 32×32 pixels and three channels, images in `EMNIST` have 28×28 pixels in resolution with only one channel, which makes it easier, not only to leak privacy information in gradients but also to recognize visually from the reconstructed images, even with defenses such as GC. Our OUTPOST does not achieve the highest LPIPS and lowest SSIM in every scenario with both attack methods; however, it can still provide solid and sufficient protection resulting in highly noisy images.

VI. INHERENT ROBUSTNESS OF DEEP CONVOLUTIONAL NEURAL NETWORKS TO GRADIENT LEAKAGE ATTACKS

Similar to previous studies [3], [7], our evaluations so far have primarily focused on Zhu's `LeNet` model [1]. However, there has been related work, such as GradInversion [4] and csDLG [2], that explored various alternative neural network models, particularly those in the ResNet [22] family. In their work, Yin et al. [4] discovered that more powerful feature extraction, as exhibited by ResNet-50, yields better data reconstruction performance compared to shallower network architectures like `ResNet-18`. To assess the robustness of various models against gradient leakage attacks, we intend to expand our analysis in this section and include deeper neural network architectures in our investigation.

Upon integrating ResNet into our implementation, we made an unexpected discovery: there were significant inconsistencies in the reconstruction results of csDLG between our adapted version and the original implementation. We have found that the image reconstruction in our implementation is significantly worse than that of the original version. Through careful examination and comparison, we identified the root cause of the discrepancies between our implementation and the original version. In the original implementation, the target gradient is computed when the model is explicitly switched to the evaluation state. However, the gradient is supposed to be derived during the client's local training in federated learning and thus the model should be in the training state. This technical oversight is also present in the implementation of GradInversion. In contrast, our implementation ensured that the model states were correctly set at the client side while the existing attack mechanisms were accurately implemented at the server side.

Such a discrepancy in the model state leads to very different behaviors of certain modules or layers within the model. When it comes to ResNets, the layers affected by the model states are the batch normalization layers. Batch normalization is a common technique incorporated into many convolutional neural networks for improving training and generalization by normalizing the activations within each batch [23]. In `PyTorch`, batch normalization is implemented by the `BatchNorm2d` [24] module, and normalization is conducted by the following calculation using the mean and standard deviation per dimension over the mini-batch of input data:

$$y = \frac{x - \mathrm{E}[x]}{\sqrt{\mathrm{Var}[x] + \epsilon}} \cdot \phi + \psi, \tag{7}$$

where ϕ and ψ are learnable parameter vectors with default values of 1 and 0, respectively. During model training, batch normalization layers keep running estimates of its computed mean and standard deviation over mini-batches. During model evaluation, the tracked running statistics are used for normalization. In addition to ResNets, batch normalization layers are also commonly found in other vision neural network models such as DenseNet [25] and VGG [26].

A. Revisiting Gradient Leakage Attacks on ResNets With Accurate Implementation

We present the reconstructed images along with the corresponding reconstruction loss and peak signal-to-noise ratio (PSNR) values obtained through csDLG by matching model updates under various scenarios in Table IV. These scenarios involving a very small number of local data samples or batches are typically unrealistic in production federated learning, but they are useful for highlighting the limited effectiveness of existing gradient leakage attacks. We compared the attack performance on both untrained and pretrained models of `ResNet-18` and `ResNet-152` on the `CIFAR-100` dataset. The pretrained models we employed have a test accuracy over 60%. The original implementation indicates that the model is set to the evaluation state at the client side, while our implementation indicates that the model is in the training state as it should be.

We are only interested in csDLG as opposed to DLG and iDLG because csDLG generally demonstrates enhanced image recovery capabilities. The cosine similarity reconstruction loss in csDLG proves to be highly beneficial, especially when the model updates become sparse during training. In such cases, csDLG can effectively measure the distance between the target model update and the dummy model update with greater accuracy, compared to the L2-Norm distance in DLG and iDLG, even when a significant number of near-zero values are present. Additionally, csDLG assumes that the attacker has knowledge of the mean and variation statistics of the entire centralized dataset for normalizing its dummy data, which facilitates image reconstruction.

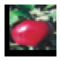
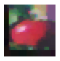
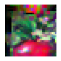
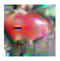
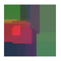
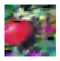
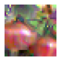
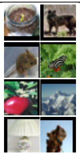
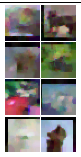
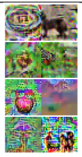
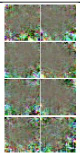
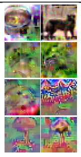
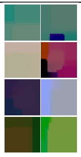
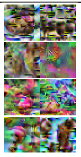
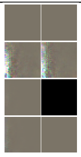
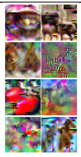
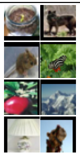
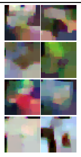
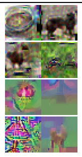
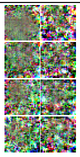
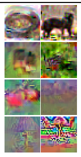
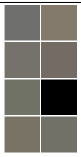
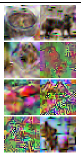
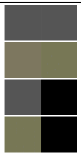
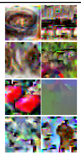
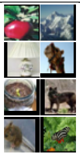
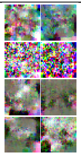
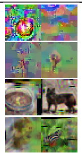
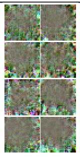
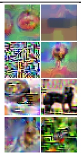
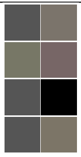
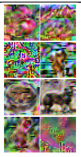
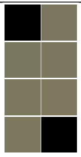
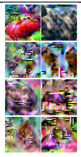
Performance discrepancy in the model training and evaluation state. By comparing the attack performance on untrained models in two different model states, we have observed a significant improvement when the model is mistakenly set to the evaluation state as in the original implementation. The reason is that, during the initial stages of training when the model has not yet encountered any data, the parameters of batch normalization layers are simply zeros in the evaluation state, thereby leaving the output unaffected. On the contrary, in the training state, batch normalization layers introduce noise as normalization uses the mean and variance statistics estimated on the very first batch. Based on the results obtained from all FedAvg settings on both `ResNet-18` and `ResNet-152`, it is evident that csDLG has completely failed, as almost no information about the raw images can be found in the reconstruction, provided that the model state is correctly set as demonstrated in our implementation.

However, as the target model progresses in its training, the performance gap between the two model states will gradually diminish. In the training state of a pretrained model, the impact of batch normalization layers is reduced because the mean and variance statistics of the current batch align more closely with the overall statistics of the training data that the model has encountered during its training process. On the other hand, in the evaluation state of a pretrained model, the running statistics (*i.e.*, moving averages of the mean and variance) derived from all the encountered data are even more representative of the entire training dataset. This explains the similar reconstruction results observed between the original implementation and our implementation on a pretrained model.

Performance on pretrained models in elementary federated learning settings. Our focus is now solely on attacks targeting pretrained models when the model is correctly set

TABLE IV

THE PERFORMANCE OF CSDLG ON BOTH UNTRAINED AND PRETRAINED MODELS OF `ResNet-18` AND `ResNet-152` ON THE `CIFAR-100` DATASET IN VARIOUS FEDERATED LEARNING SETTINGS. A LOWER RECONSTRUCTION LOSS OR A HIGHER PSNR INDICATES BETTER IMAGE RECONSTRUCTION QUALITY. ALL THE RESULTS WERE OBTAINED AFTER A SUFFICIENT NUMBER OF ITERATIONS, ENSURING THAT THE ATTACK HAS CONVERGED. THE GROUND TRUTH IMAGES WHICH ARE THE CLIENT'S TRAINING DATA MAY BE RANDOMLY CROPPED OR HORIZONTALLY FLIPPED

Settings	**Ground truth**	**ResNet-18 & CIFAR-100**				**ResNet-152 & CIFAR-100**			
		Original implementation		Our implementation		Original implementation		Our implementation	
		Untrained	Pretrained	Untrained	Pretrained	Untrained	Pretrained	Untrained	Pretrained
$E=1, n=1, B=1$ Loss PSNR (dB)		 0.022 18.565	 0.204 10.289	 0.598 10.568	 0.730 9.570	 0.012 11.898	 0.090 10.382	 0.992 10.755	 0.807 9.699
$E=1, n=8, B=8$ Loss PSNR (dB)		 0.022 19.222	 0.173 9.301	 0.681 10.630	 0.316 10.906	 0.004 9.152	 0.244 8.841	 0.983 10.646	 0.538 9.470
$E=5, n=8, B=8$ Loss PSNR (dB)		 0.013 12.585	 0.119 10.598	 0.565 8.380	 0.321 10.581	 0.023 12.589	 0.092 11.768	 0.761 12.466	 0.549 11.350
$E=1, n=8, B=4$ Loss PSNR (dB)		 0.096 9.563	 0.031 11.821	 0.695 10.160	 0.389 10.678	 0.883 12.535	 0.119 10.582	 0.739 12.460	 0.617 9.995

to the training state. According to the four settings shown in Table IV, it is possible for model updates from a pretrained model to contain a small amount of information about the private training data. For instance, in the reconstructed images, it is possible to observe a red round object, although the number of objects may not align with the ground truth. Additionally, in the case of 8 data points, a small portion of the reconstructed images may contain recognizable objects, such as a four-legged animal and a cup or jar.

When the total number of local data samples (n) is equal to the batch size (B), meaning that the same data samples are used in every batch, increasing the number of epochs appears to amplify the amount of private information carried in the model updates, particularly in the case of `ResNet-152`. However, it is important to note that the settings evaluated in Table IV are overly simplistic and do not accurately represent real-world federated learning scenarios. Despite these simplistic settings, the observed information leakage from model updates remains limited.

The reconstruction loss and PSNR may not accurately reflect the amount of private information contained in the reconstructed images still. For instance, in some cases, images reconstructed from untrained models may exhibit lower reconstruction losses and higher PSNR values compared to those from pretrained models, while the former images consist of completely random noise. Exploring additional evaluation methods and metrics specifically designed for gradient leakage attacks can be an interesting research topic.

In addition to the `ResNet-18` and `ResNet-152` architectures, we also explored a wider range of deep neural networks, such as Vision Transformer (ViT) models [27]. The experimental results on `ViT` imported from HuggingFace [28], along with datasets such as `CIFAR-100` and `Tiny ImageNet`, are shown in Table V. Even in the simplest scenario where $E = 1, n = 1, B = 1$, the reconstruction quality remains very low when using our correct implementation regarding the model state. `ViT` exhibits significantly higher resilience against the csDLG attack compared to `LeNet` and `ResNets`. Additionally, the reconstruction of higher resolution image data in `Tiny ImageNet` presents increased difficulty within the same experimental settings.

B. Assessing the Effectiveness of Our Defense on ResNets

Despite the fact that the scenarios analyzed in Table IV may not be realistic in real-world federated learning due to the extremely small number of local data samples or batches, we have validated the effectiveness of our defense mechanism, OUTPOST, in preventing data leakage in these worst-case

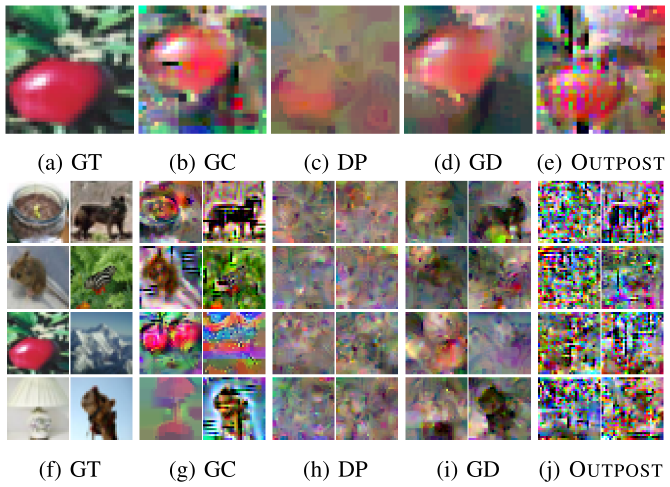
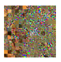
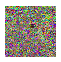
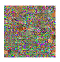
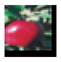
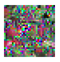


Fig. 4. The effectiveness of various defense mechanisms against csDLG on pretrained `ResNet-18`. The first row (a)-(e) presents reconstructed images in the case of $E = 1, n = 1, B = 1$; and the second row (f)-(j) presents reconstructed images in the case of $E = 1, n = 8, B = 8$.

TABLE V

THE PERFORMANCE OF CSDLG ON BOTH UNTRAINED AND PRETRAINED MODELS OF `ViT` ON THE `Tiny ImageNet` AND `CIFAR-100` DATASETS IN THE CASE OF $E = 1, n = 1, B = 1$

Ground truth	Original implementation		Our implementation	
	Untrained	Pretrained	Untrained	Pretrained
`Tiny ImageNet`				
Loss	0.106	0.429	0.116	0.208
PSNR (dB)	10.891	8.941	11.178	9.164
`CIFAR-100` 				
Loss	0.108	0.099	0.120	0.088
PSNR (dB)	9.979	9.321	8.764	9.822

situations for the client. In Fig. 4, we present the impact of different defenses on image reconstruction from csDLG on the pretrained `ResNet-18`, specifically when $E = 1, n = 1, B = 1$ and $E = 1, n = 8, B = 8$. Note that all the hyperparameter configurations for these defenses remain the same as initially specified in Section V.

As expected, differential privacy (DP) can provide the strongest protection for model updates, ensuring that not a single object can be recognized in the reconstructed images in both scenarios. However, it is well-known that DP also has the greatest impact on model accuracy. Gradient compression (GC) is the least effective defense compared to others, which suggests that the cosine similarity distance used in csDLG's reconstruction loss can effectively mitigate the impact of small value pruning in gradients by GC. The results of Soteria are unavailable since the original design is not compatible with networks other than `LeNet`. GradDefense (GD) and OUTPOST have been shown to effectively reduce the quality of reconstructed images. However, similar to the observations in Fig. 3, we have noticed that GradDefense also introduces significant delays to FedAvg on `ResNet-18` due to the computation overhead. These delays are more severe compared to the delays experienced with `LeNet`. Overall, our defense mechanism still offers the best tradeoff between data protection and training efficiency.

VII. DEFENDING AGAINST GRADIENT LEAKAGE ATTACKS BEYOND HONEST SERVERS

Thus far, our findings demonstrate minimal data leakage when faced with an honest-but-curious server in production federated learning, even in the simplistic scenarios. Moreover, our lightweight defense mechanism is efficient to safeguard against potential leakage without compromising training performance. Building upon these achievements, our research now shifts its focus from honest-but-curious server settings to more malicious ones. In this context, where federated learning clients face increased risks, we will further evaluate the effectiveness of the OUTPOST defense.

Both the Robbing the Fed attack [10] and the Fishing attack [11] fall into the category of malicious servers. The Robbing the Fed attack involves modifying the architecture of the global model by inserting an imprint module, which consists of large linear layers. This modification induces a structured pattern in the model update, allowing for the extraction of information pertaining only to a subset of data points, which can be precisely recovered subsequently. However, this attack is relatively easy for clients to detect, as they can observe differences in the model's architecture. In contrast, the Fishing attack only modifies the parameters of the classification layer of the global model, which can magnify the gradients specifically associated with a targeted class, enabling the attacker to recover single data points belonging to that class from an arbitrary large batch of client's local data. Since it is more challenging for clients to detect parameter modifications, we are particularly interested in the Fishing attack.

The Fishing attack requires querying the client iteratively in every communication round while modifying the parameters of the global model sent to the client. This process continues until the corresponding gradient is reduced to reflect the update of a single data point only. There are two strategies employed in the Fishing attack: the class fishing strategy and the feature fishing strategy. The class fishing strategy is designed for cases where there is only one data sample belonging to the target class. On the other hand, the feature fishing strategy is more advanced and addresses intra-batch collisions in the label space, preventing mixed updates on all images from the target class. The feature fishing strategy typically requires more communication rounds for queries, but the number is usually fewer than 10 times as shown in [11]. The attacker selects either attack strategy based on their observation of the client's label space.

After integrating the Fishing attack into our codebase, we have discovered, unsurprisingly, that it explicity sets the client's model to the evaluation state. When we put the model back to the training state, the favorable attack performance observed in the Fishing attack paper diminishes significantly. Additionally, the Fishing attack has several other limitations. It only works for FedSGD, where there is only one gradient descent step in a single communication round, *i.e.,* $E = 1$ and $n = B$. This setup is essentially simpler compared to typical

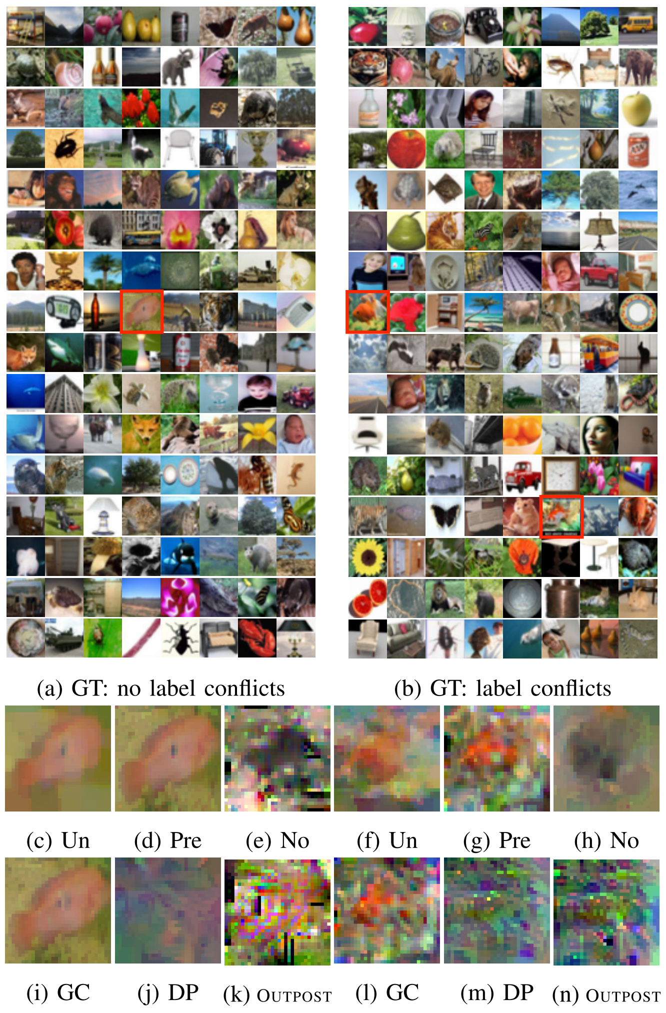


Fig. 5. Images reconstructed from a batch of 128 `CIFAR-100` data samples, obtained through the Fishing attack on `ResNet-18`. The left column corresponds to the scenario where only one data point belongs to the target class "1", while the right column represents the scenario where there are multiple data points belonging to the target class "1". The ground truth images of the 128 data samples are displayed in (a) and (b), with the target images highlighted by red circles. The attack results on the untrained model and pretrained model are displayed in (c) & (f), (d) & (g), respectively. Additionally, the ablation results on the pretrained model without parameter modification are displayed in (e) & (h). The final row displays the attack results on the pretrained model when the client employs various defense mechanisms.

real-world FedAvg settings, which involve multiple gradient descent steps within a single communication round, *i.e.,* $E \geq 1$ and $n \gg 1, B < n$. Additionally, the Fishing attack requires querying the client for labels initially in order to locate the target images within the batch.

We have conducted experiments on the Fishing attack with different defenses using the `ResNet-18` model and the `CIFAR-100` dataset within the setting $E = 1, n = B = 128$. The results in different scenarios are shown in Fig. 5. When there is only one data sample belonging to the target class, the class Fishing attack works exceptionally well on a pretrained model (with a test accuracy over 60%), even with such a large batch size. On the other hand, when there are two data samples belonging to the target class, which is quite common in real-world federated learning, the feature Fishing attack can only capture part of features of the target class. As a result, fish-like patterns can somehow be recognized in the reconstructed image. In contrast, for an untrained model, the recovery capability is slightly inferior in both cases, as the reconstructed pixels are less finely detailed. Additionally, we conducted an ablation study by removing the parameter modification procedure in the Fishing attack. In this case, the attack completely failed, indicating the crucial role of parameter modification in the success of the attack.

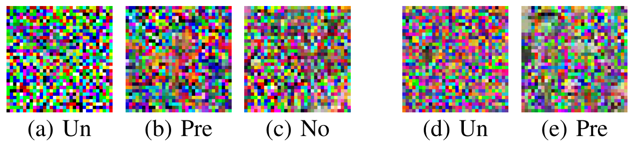


Fig. 6. Images reconstructed from a batch of 128 `CIFAR-100` data samples through the Fishing attack on `ViT` under the same settings as depicted in Fig. 5. Results without label conflicts are shown in (a)–(c), while results with label conflicts are shown in (d) & (e). When label conflicts exist within the batch of data, the ablation study of the Fishing attack on the pretrained model without parameter modification failed to converge.

The effectiveness of different defenses against the Fishing attack on the pretrained model is depicted in the final row of Fig. 5. It should be noted that the hyperparameter configurations of these defenses remained unchanged without explicit tuning. Unfortunately, the results of Soteria are not available as the original design did not consider ResNets. The results of GradDefense are also not shown due to the excessively long time it takes to calculate the sensitivity and apply it to the gradients. GC still demonstrates limited improvement compared to having no defense. DP, on the other hand, generally obscures the images to the extent that no information can be extracted from them. We also observed that when label conflicts exist, the Fishing attack requires multiple hundreds of queries to the client before the attack converges with the presence of DP. This significantly diminishes the attack effectiveness. Once again, OUTPOST demonstrates the ability to provide sufficient protection on the gradients without significantly affecting the model's convergence speed or introducing excessive computation overhead.

We have also conducted experiments with the Fishing attack on the `ViT` model using the `CIFAR-100` dataset, as depicted in Fig. 6. Interestingly, regardless of whether the model is untrained or pretrained, and whether there are label conflicts or not, the Fishing attack completely failed on `ViT`. This observation suggests that the modification of model parameters in the Fishing attack may not be as effective for models other than ResNets. Consequently, it appears that no defense strategy is necessary in this scenario.

VIII. Related Work

Starting from the pioneering work in [1], researchers have made efforts to improve the capability and efficiency of gradient leakage attacks. One of the highlights of such efforts, iDLG [3], further discovered that ground-truth labels can be directly extracted from the given gradients, which simplified the gradient matching process since it only needed to recover the inputs $\mathbf{x}'$ in Eq. 1. Different from using the Euclidean distance as the objective function in DLG, Geiping et al. [2] utilized cosine similarity between the target gradient and the dummy gradient to optimize the reconstructed data during local updates. GradInversion [4] demonstrated even higher fidelity and better localization as compared with [1] and [2]. GIAS [12] revealed more severe privacy leakage of raw data from gradients with prior knowledge about the pre-trained generative model. These variants have made up the landscape of gradient leakage attacks to date.

Geiping's research group initiated an exploration of attacks in the settings of malicious servers rather than honest-but-curious servers, and proposed two specific attacks on vision tasks, namely the Robbing the Fed attack [10] and the Fishing attack [11]. The Robbing the Fed attack modifies the model architecture, while the Fishing attack modifies the model parameters, which help amplify the private information carried in the corresponding gradients. The Fishing attack enables the recovery of single data points of the target class from a large aggregate of a client's dataset.

To preserve the privacy from the gradients, researchers have applied several categories of general-purpose defenses to the gradient leakage attack, such as gradient compression and local differential privacy. But as [6] argued, these mechanisms were not custom tailored to this specific attack, and may incur either unacceptable computational overhead or significant degradation of performance, with respect to the converged accuracy after the FL training process completes. A particular defense, Soteria [5], was proposed specifically for the gradient leakage attack. In order to reduce the quality of the reconstructed data, and based on their finding that privacy leakage mainly comes from data representations embedded in the fully connected (FC) layer, it proposed a delicate design of perturbation on the data representation in the FC layer of shared gradients. Yet, GradDefense [6] showed that the raw data can still be recovered from the remaining gradients by muting the perturbed layer, and thus proposed a stronger defense by perturbing all the layers of the shared gradients by their measured sensitivity. Setting aside its statements contradictory to [5], GradDefense [6] demanded a substantial amount of computation for sensitivity measurements and perturbation by all the layers. Based on our own empirical observations and evaluations of the threat of gradient leakage attacks in the context of production FL, we are motivated to devise OUTPOST, a defense mechanism that is simple but good enough to defend against the small attack surface in production FL. OUTPOST is designed to optimize the trade-off between computation overhead, accuracy guarantee, and privacy preservation.

IX. CONCLUDING REMARKS

In this paper, we have thoroughly investigated gradient leakage attacks, a popular category of attacks in federated learning. Our original objective was to conduct an in-depth study of these attacks in the context of production federated learning systems, and to design a practical, simple, and lightweight defense mechanism that can be used to defend against real-world threats. Along our journey to achieve this goal, we discovered that the effectiveness and efficiency of existing gradient leakage attacks are weakened by a substantial margin in standard federated learning settings. In a nutshell, gradient leakage attacks are highly unlikely to succeed in federated learning, when clients send model updates rather than gradients to the server, perform multiple local training epochs — each containing multiple iterations — over local data with a non-i.i.d. distribution, initialize model weights normally, introduce more than one data sample per label, and last but not the least, correctly use the training mode during local training. This conclusion holds for both honest-but-curious and malicious servers, and the latter are capable of modifying the model at will.

It is worth noting that, among these assumptions to make attacks stronger, the most critical one is to use the evaluation mode, rather than training mode, for local training on each client. Short of blindly accepting a binary executable from the server, it would be highly unlikely for an autonomous client to train its local model in the evaluation mode.

Beyond using an extensive array of experiments to show specific and highly unrealistic cases where attacks are more likely to succeed, we also proposed OUTPOST, a new defense mechanism that can provide sufficient protection on shared model updates without sacrificing accuracy and convergence speed, and can adapt to time-varying levels of the privacy leakage risk throughout the federated learning process. In all vulnerable cases we identified that need protection, we showed convincing results that OUTPOST incurs much less computational overhead, achieves better accuracy, and converges much faster than its state-of-the-art alternatives in the literature. OUTPOST is shown to be not only effective when training baseline models such as `LeNet`, but also when deeper convolutional neural network models, such as ResNet, are trained with the presence of honest-but-curious and malicious servers.

With or without engaging OUTPOST, the implications that federated learning is fully capable of preserving data privacy in practice are profound. Several entire categories of defense mechanisms, including differential privacy, secure aggregation, and homomorphic encryption, are no longer necessary in federated learning. As we anticipate that this research will evolve over time and our conclusions may be remedied or reversed, we provide complete access to all source code used in this paper to the community. It is our hope that, with this paper, federated learning will be seen as trustworthy again in the not-so-distant future.

REFERENCES

[1] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2019, pp. 14774–14784. [Online]. Available: https://dl.acm.org/doi/10.5555/3454287.3455610

[2] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients–How easy is it to break privacy in federated learning?" in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 33, 2020, pp. 16937–16947.

[3] B. Zhao, K. R. Mopuri, and H. Bilen, "IDLG: Improved deep leakage from gradients," 2020, *arXiv:2001.02610*.

[4] H. Yin, A. Mallya, A. Vahdat, J. M. Alvarez, J. Kautz, and P. Molchanov, "See through gradients: Image batch recovery via gradinversion," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 16337–16346.

[5] J. Sun, A. Li, B. Wang, H. Yang, H. Li, and Y. Chen, "Soteria: Provable defense against privacy leakage in federated learning from representation perspective," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 9311–9319.

[6] J. Wang, S. Guo, X. Xie, and H. Qi, "Protect privacy from gradient leakage attack in federated learning," in *Proc. INFOCOM IEEE Conf. Comput. Commun.*, May 2022, pp. 580–589.

[7] W. Wei et al., "A framework for evaluating client privacy leakages in federated learning," in *Proc. Eur. Symp. Res. Comput. Secur.* Cham, Switzerland: Springer, 2020, pp. 545–566.

[8] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of FedAvg on non-IID data," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2020, pp. 1–26.

[9] B. McMahan et al., "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

[10] L. H. Fowl, J. Geiping, W. Czaja, M. Goldblum, and T. Goldstein, "Robbing the fed: Directly obtaining private data in federated learning with modified models," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2022.

[11] Y. Wen, J. Geiping, L. Fowl, M. Goldblum, and T. Goldstein, "Fishing for user data in large-batch federated learning via gradient magnification," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2022, pp. 1–17.

[12] J. Jeon, J. Kim, K. Lee, S. Oh, and J. Ok, "Gradient inversion with generative image prior," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 29898–29908.

[13] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 32, 2019, pp. 1–12.

[14] Y. Huang, S. Gupta, Z. Song, K. Li, and S. Arora, "Evaluating gradient inversion attacks and defenses in federated learning," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 34, 2021, pp. 1–10.

[15] H. Wu and P. Wang, "Fast-convergent federated learning with adaptive weighting," *IEEE Trans. Cognit. Commun. Netw.*, vol. 7, no. 4, pp. 1078–1088, Dec. 2021.

[16] D. Maclaurin, D. Duvenaud, and R. Adams, "Gradient-based hyperparameter optimization through reversible learning," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 2113–2122.

[17] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1026–1034.

[19] S.-I. Amari, "Natural gradient works efficiently in learning," *Neural Comput.*, vol. 10, no. 2, pp. 251–276, Feb. 1998.

[20] J. Martens, "Deep learning via hessian-free optimization," in *Proc. Int. Conf. Mach. Learn. (ICML)*, vol. 27, 2010, pp. 735–742.

[21] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018, pp. 1–14.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[23] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

[24] *Documentations on Torch.nn*. Accessed: Dec. 2023. [Online]. Available: https://pytorch.org/docs/stable/nn.html

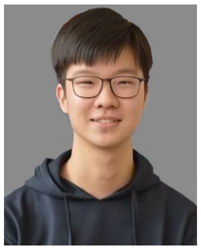
[25] G. Huang, Z. Liu, V. Laurens, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Aug. 2016, pp. 2261–2269.

[26] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2015, pp. 1–14. [Online]. Available: https://ora.ox.ac.uk/objects/uuid:60713f18-a6d1-4d97-8f45-b60ad8aebbce

[27] A. Dosovitskiy, "An image is worth 16×16 words: Transformers for image recognition at scale," in *Proc. ICLR*, 2021, pp. 1–12.

[28] *The Hugging Face Hub*. Accessed: Nov. 2, 2023. [Online]. Available: https://huggingface.co/models

Fei Wang (Graduate Student Member, IEEE) received the B.Eng. degree (Hons.) from the Hongyi Honor College, Wuhan University, China, in 2020. She is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, University of Toronto, Canada. Her research interests include efficiency improvement and privacy leakage in distributed machine learning. She was a recipient of the Best Paper Award from IEEE INFOCOM in 2023.



Ethan Hugh is currently pursuing the B.A.Sc. degree in computer engineering with the University of Toronto, Canada. His research interests include algorithms, computer systems programming, and machine learning. He was a recipient of the Best Paper Award from IEEE INFOCOM in 2023.

Baochun Li (Fellow, IEEE) received the B.Eng. degree from Tsinghua University in 1995 and the M.S. and Ph.D. degrees from the University of Illinois at Urbana–Champaign in 1997 and 2000, respectively. Since 2000, he has been with the Department of Electrical and Computer Engineering, University of Toronto, where he is currently a Professor. Since August 2005, he has been with the Bell Canada Endowed Chair in computer engineering. His current research interests include cloud computing, security and privacy, distributed machine learning, federated learning, and networking. He is a Fellow of the Canadian Academy of Engineering and the Engineering Institute of Canada. He was a recipient of the IEEE Communications Society Leonard G. Abraham Award in the Field of Communications Systems in 2000, the Multimedia Communications Best Paper Award from the IEEE Communications Society in 2009, the University of Toronto McLean Award in 2009, the Best Paper Award from IEEE INFOCOM in 2023, and the IEEE INFOCOM Achievement Award in 2024.