Extracting Private Training Data in Federated Learning from Clients

Extracting Private Training Data in Federated Learning from Clients

Jiaheng Wei, Yanjun Zhang, Leo Yu Zhang, *Member, IEEE,* Chao Chen, *Member, IEEE,* Shirui Pan, *Senior Member, IEEE,* Kok-Leong Ong, Jun Zhang, *Senior Member, IEEE,* Yang Xiang, *Fellow, IEEE.*

***Abstract*—The utilization of machine learning algorithms in distributed web applications is experiencing significant growth. One notable approach is Federated Learning (FL) Recent research has brought attention to the vulnerability of FL to gradient inversion attacks, which seek to reconstruct the original training samples, posing a substantial threat to client privacy. Most existing gradient inversion attacks, however, require control over the central server and rely on substantial prior knowledge, including information about batch normalization and data distribution. In this study, we introduce Poisoning Gradient Leakage from Client (PGLC), a novel attack method that operates from the clients' side. For the first time, we demonstrate the feasibility of a client-side adversary with limited knowledge successfully recovering training samples from the aggregated global model. Our approach enables the adversary to employ a malicious model that increases the loss of a specific targeted class of interest. When honest clients employ the poisoned global model, the gradients of samples become distinct in the aggregated update. This allows the adversary to effectively reconstruct private inputs from other clients using the aggregated update. Furthermore, our PGLC attack exhibits stealthiness against Byzantine-robust aggregation rules (AGRs). Through the optimization of malicious updates and the blending of benign updates with a malicious replacement vector, our method remains undetected by these defense mechanisms. We conducted experiments across various benchmark datasets, considering representative Byzantine-robust AGRs and exploring different FL settings with varying levels of adversary knowledge about the data. Our results consistently demonstrate the ability of PGLC to extract training data in all tested scenarios.**

***Index Terms*—Federated learning, privacy attack, inversion attack.**

I. Introduction

With the ongoing development of web applications, machine learning technologies have found widespread application in various web-based platforms, particularly in the context of distributed web applications. Federated Learning (FL) [1], [2], [3] is an emerging distributed learning framework that represents a stride towards preserving user privacy and reducing dependency on centralized data storage. It allows multiple clients or participants (e.g., private mobile devices or IoT devices) to join the federated training collaboratively and improve model generalization without sharing their private data. Specifically, the clients can train the distributed model on their private datasets locally and submit the local model or gradient to the server, then the server will aggregate these updates using an aggregation algorithm (AGR).

Recent studies have highlighted the vulnerability of federated learning (FL) to gradient inversion adversaries, who aim to reconstruct private local training data by exploiting the gradients shared during the FL process. One common strategy employed by gradient inversion attacks is to optimize random noise in order to approximate the true gradient and subsequently reconstruct the original input data [4], [5], [6], [7], [8], [9], [10], [11], [12], [13]. However, existing gradient inversion attacks (GIAs) against FL exhibit the following limitations:

- Many existing GIAs necessitate control over the server to launch the attack [6], [8], [10], [11], [12]. However, in real-world scenarios, gaining control over the server is often challenging as it is typically heavily secured to prevent unauthorized access. Moreover, if the AGR operates in the encrypted domain [14], all server-side existing GIAs would fail. Therefore, client-side GIAs gain greater significance. However, the efficacy of such attacks [6], [5], [8] remains limited to unrealistically small batch sizes.
- Existing GIAs also rely on strong prior knowledge or specific information to achieve their objectives. For instance, GradInversion [8] requires access to batch normalization information, while generative-based methods [13], [15], [16] utilize the distribution of the training data as prior knowledge for the adversary. These requirements for strong prior knowledge or specific information in GIAs restrict their applicability and effectiveness in real-world settings.
- Some GIAs necessitate altering the model structure or parameters, potentially leading to unintended consequences. For instance, some methods require the adversary to manipulate the training process in a way that traps clients into a predetermined status preferred by the ad-

Manuscript received May 16, 2024.

Jiaheng Wei, Chao Chen, and Kok-Leong Ong are with the School of Accounting, Information System and Supply Chain, RMIT University, Melbourne, VIC 3001, Australia (e-mail: s3986349@student.rmit.edu.au; chao.chen@rmit.edu.au; kok-leong.ong2@rmit.edu.au).

Yanjun Zhang is with the School of Computer Science, University of Technology Sydney, Sydney, NSW 2007, Australia (e-mail: Yanjun.Zhang@uts.edu.au).

Leo Yu Zhang and Shirui Pan are with the School of Information and Communication Technology, Griffith University, Brisbane, QLD 4215, Australia (e-mail: leo.zhang@griffith.edu.au; s.pan@griffith.edu.au).

Jun Zhang and Yang Xiang are with the School of Science, Computing and Engineering Technologies, Swinburne University of Technology, Melbourne, VIC 3122, Australia (e-mail: junzhang@swin.edu.au; yxiang@swin.edu.au).

versary [12], [10]. However, this alteration can negatively impact the performance or even disrupt the entire training framework [11].

In this study, we introduce the first **P**oisoning **G**radient **L**eakage from **C**lient attack (PGLC). The PGLC involves an adversary developing a malicious model that increases the loss for a specific targeted class locally. This malicious model can then poison the global model from the client side. When honest clients use this poisoned global model, the gradients of samples from the targeted class are enhanced. Consequently, the targeted gradient significantly influences the aggregated update, allowing the adversary to successfully rebuild the private input of other clients through the aggregated update.

As such, our PGLC is capable of successfully inverting gradients with larger batch sizes. Additionally, PGLC is designed to function without any prior knowledge. Even if the adversary lacks information about the data distribution, they can still recover private data. Furthermore, our PGLC aligns with the general training framework of FL and maintains the performance of the untargeted classes during attacking.

Another highlight of PGLC is its ability to remain stealthy even when confronted with server-side defense measures, specifically Byzantine-robust AGRs, which aim to filter out malicious updates by analyzing the distribution of updates [17], [18] or their performance on the validation dataset [19], [20]. We developed an valid method for optimizing malicious updates by disguising them as benign updates. This involves blending the benign update with a malicious replacement vector. The attacker can fine-tune the mixed hyper-parameters to facilitate the poisoning process or enhance the update's stealthiness against Byzantine-robust AGRs.

We also consider our PGLC in a range of diverse FL settings where the adversary's knowledge about data can vary significantly. We consider three distinct levels of knowledge based on the data distribution information: i) *Full Knowledge*: In this setting, the adversary possesses the knowledge of the data distribution across all classes. ii) *Semi Knowledge*: The adversary only has access to the data distribution information specifically for the targeted class. iii) *No Knowledge*: In this scenario, the adversary lacks any prior knowledge regarding the data distribution of any class.

Our results demonstrate that PGLC poses a high risk against FL. We conducted a comprehensive evaluation of PGLC using three benchmark datasets and tested it against four representative Byzantine-robust AGRs across various levels of knowledge. Our results show that PGLC consistently and successfully extracts training input in all scenarios, surpassing the performance of existing GIAs. Notably, even in a scenario where the adversary possesses no prior knowledge, PGLC can successfully restore the training samples from batch sizes of 32 while the baselines only yield noise without any meaningful semantic features. Once given extra knowledge, a boost in performance is observed.

The contributions of our work are summarized as follows:

- We propose a new GIA, which is the first client-side gradient inversion attack capable of reconstructing another client's training sample from a large batch. It eliminates the strong assumption on prior knowledge. Additionally, PGLC adheres to the conventional training framework without requiring modifications to the model's architecture or parameters, and it preserves standard performance on untargeted classes during the attack.
- Our PGLC demonstrates the ability to evade detection by the Byzantine-robust AGRs. We reduce the attack into an optimization problem that maximizes the adversarial impact on the victim training samples while ensuring that the malicious updates remain disguised. PGLC can effectively generate malicious local updates while remaining disguised within the normal training data.
- We evaluate our PGLC under a comprehensive threat model including real-world attack scenarios considering different knowledge that the adversary can obtain through FL. The results demonstrate PGLC's ability to succeed under different knowledge scenarios and practical settings.
- We release the source code and the artifact at https://github.com/clientSideGIA/PGLC, which creates a new tool for the GIA.

II. Background and Related Work

A. Federated Learning

Federated learning (FL) [1], [2], [3] is a distributed learning framework. Generally, the common FL algorithms are FedAvg [1] and FedSGD [1]. The standard FL structure contains a server model Φ_G and N clients. Each client i has a private dataset D_i drawn from a data distribution $\mathcal{D}$.

For FedSGD, in every round t, each client i first receives the global model Φ_G^{t-1} broadcasted by the server, then commits the gradient $\delta_i^t = \nabla_{\Phi_G^{t-1}} \mathcal{L}(D_i)$ to the server. The server will aggregate selected client gradients and use the averaged gradient $\overline{\delta}^t = \frac{1}{N}\sum_{i=1}^{N} \delta_i^t$ to update the global model and obtain Φ_G^t. Then, the server distributes Φ_G^t to each client and begins the next training round.

For FedAvg, it is similar to FedSGD, but each client i can initialize a local model $\dot{\Phi}_i^t$ with Φ_G^{t-1} and run more local epochs with dataset D_i to optimize the local model $\dot{\Phi}_i^t$. The server will collect all trained local models and aggregate them with weight averaging, then the server gets a new global model $\Phi_G^t = FedAvg(\{\dot{\Phi}_i^t | i \in [N]\})$. Note that the model update in FedAvg equals the gradient update in FedSGD when FedAvg only runs one local epoch. Then the server will distribute the current global model to every client and begin the next round. The key feature of FL is that no private data will be disclosed to other clients or the server.

B. Byzantine-robust Aggregation Rules

Due to the distributed nature of FL, multiple Byzantine-robust AGRs [17], [18], [19], [20], [21], [22] are proposed to defend against malicious updates from clients. *Krum*, *Multi-krum* [17] and *Bulyan* [18] aim to identify and remove malicious gradients by calculating the distance between different updates. The idea is to cluster similar updates together and treat the remaining gradients as potentially malicious. *Trimmed-mean* and *Median* [21], [22] use either the median or trimmed mean (excluding extreme values) to aggregate

gradients, aiming to mitigate the influence of extreme gradients. *AFA* [19] introduces a trust bootstrapping mechanism to identify and eliminate potentially harmful local updates that deviate significantly from a trusted clean dataset used as a reference. *Fang defense* [20] uses a validation set on the server side to filter gradients. It evaluates the performance of gradients on the validation set and discards those that do not align with the overall model's behavior.

C. Local Model Poisoning Attack against FL

Model poisoning attacks against FL systems can have two objectives: breaking model performance to cause Denial-of-Service (DoS) or introducing a backdoor into the model.

When aiming to disrupt the performance of a model, recent studies have focused on developing strategies to create local poisoned models or model updates that can successfully penetrate Byzantine-robust AGRs. An example of such an attack is the LIE attack [23], which requires minimal knowledge of Byzantine-robust AGRs and introduces small perturbations to benign gradients to evade detection. Fang [20] introduced a general optimization problem where adversaries optimize local poisoned gradients to make the aggregated global model deviate in the opposite direction from the previous round. Recently, Shejwalkar et al. [24] proposed a comprehensive framework for model poisoning under robust AGRs, considering different levels of knowledge and developing AGR-tailored attacks and AGR-agnostic attacks.

In the case of backdooring FL systems, Bagdasaryan et al. [25] proposed a method where an adversary trains a backdoor model locally and subsequently replaces the global model by employing adversarial model replacement methods. Wang et al. [26] demonstrated the feasibility of utilizing edge-case samples to introduce a backdoor into the global model. Bhagoji et al. [27] provided an effective optimization process for executing a backdoor attack under Byzantine-robust AGRs in FL. This approach incorporates a stealthy element that optimizes malicious gradients to align with the average of benign gradients.

Different from traditional poisoning attacks which aim to corrupt the model's integrity, PGLC's objective is to reconstruct the original private training samples through poisoning.

D. Inversion Attack

Inversion attacks typically consist of two types: model inversion attack (MIA) and gradient inversion attack (GIA). Almost all GIAs are based on server-side. In these attacks, the adversary aims to reconstruct original training data by utilizing either gradient information or a publicly available model.

Typical MIA [4] attempts to fit real confidence score distribution to extract training data from DNN based on black-box access. More advanced model inversion attacks can only use label [28] or exploit GAN [29] to launch the attack.

GIA usually poses a greater threat compared to MIA due to its ability to reconstruct users' private input with higher precision. The adversary's goal in a GIA attack is to find an input (x, y) such that it can produce the same gradient as a certain local gradient update (or equivalently model update), which can be observed by the server during FL training. It is generally achieved by optimizing the l_2 distance [5] or cosine similarity [6] between the gradient associated with (x, y) and server's observation. Inferring the true value of y (i.e., label inference) is essential in the effectiveness of GIAs, including reducing the search space and stabilizing the optimization processing, which finally improves inversion performance. In particular, iDLG [9] and GradInversion [8] developed successful zero-shot label inference methods based on the feature of cross-entropy loss and the final layer gradient. By assuming knowledge of batch normalization layers as a prior, GradInversion [8] makes an amazing effect on image gradient inversion. Some researchers also demonstrated other methods to invert gradients besides optimization methods. One of the successful methods is utilizing mathematical analysis to infer input [30]. In such attacks, the adversary generally plays as server side and sends malicious parameters to clients [12], [10] or change framework structure [11] to capture some features. Another successful method is based on generative models. These methods [13], [15], [16] can produce realistic data, but the data is different from the inputs.

III. Threat Model

A. Adversary's Capabilities

We consider a typical poisoning client-side adversary's capability in a common federated learning system [1] following the prior research [25], [27], [19], [24], [31]. The adversary can download the global model of each round and take the full control of m out of N clients, including their local updates and local training data. The adversary can also decide whether to inject malicious updates for the current round or not. We assume the adversary does not compromise the central aggregator. The server can equip the defense methods, i.e., *Byzantine-robust AGRs*.

B. Adversary's Goal

The adversary aims at continuously rebuilding the private training samples from benign clients. We call the victim sample to be restored as the *targeted sample*, and the class that the targeted sample belongs to as the *targeted class*. We remark that the adversary does not select a particular target sample from honest clients as they know nothing about others' training data. Instead, they focus on a targeted class and continuously inverse the aggregated gradients during the FL training. If the aggregated gradients include samples from the targeted class, the adversary can proceed to restore them. For the other samples that are not under attack, we refer to them as the *untargeted samples* and their corresponding classes as the *untargeted classes*.

C. Adversary's Knowledge

We place PGLC under diverse FL settings with varying levels of adversary knowledge about the data, so as to conduct

a comprehensive evaluation of PGLC. We consider the adversary's knowledge from two aspects: knowledge of training data distribution and knowledge of the AGR algorithm of the server.

For the knowledge of data distribution (denote as $\mathcal{K}$), we consider three levels:

Full Knowledge *($\mathcal{K}_{full}$)*. The adversary has the full knowledge of the data distribution of all classes. As such, he/she can generate synthetic training records of each class from their marginal distributions. Such an assumption applies to the FL with IID data in which the adversary is able to obtain the statistical information of all classes from its own training dataset.

Semi Knowledge *($\mathcal{K}_{semi}$)*. The adversary has the knowledge of the data distribution of the targeted class, and the knowledge of the data distribution of some other (not all) untargeted classes. This assumption applies to the FL with Non-IID data in which the adversary only has the training samples from some certain classes including the targeted class.

No Knowledge *($\mathcal{K}_{no}$)*. The adversary knows nothing about the underlying distribution of any classes. However, the adversary has the potential to acquire information from the distributed global model. This assumption also applies to the Non-IID FL, while in this case, the training samples held by the adversary are insufficient for them to derive the underlying distribution. In addition, the adversary does not hold any training samples from the targeted class.

For the knowledge of AGR, we assume the adversary is AGR-agnostic, which is in favor of the defense mechanism as it makes the attack harder.

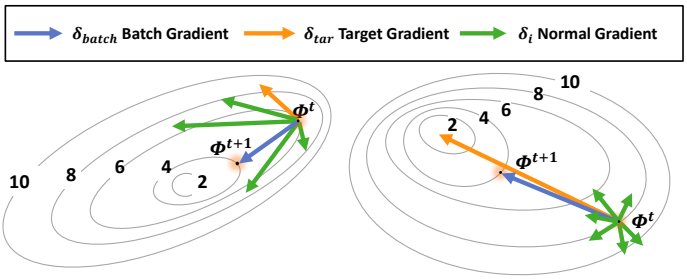


Fig. 1: **Left**: gradient aggregation on normal local updates; **Right**: gradient aggregation with poisoned model.

IV. PGLC

A. Intuition

The particular challenges for a client-side adversary are two-fold: (1) How to extract the targeted gradient from a batch without the strong assumption of prior knowledge, such as batch normalization (BN) information and data distribution? (2) How to retain the attack's effectiveness in the presence of AGRs? In the following, we present the insight we use in PGLC to tackle these challenges.

Obtain the targeted gradient from the batch. We design a poisoning client-side adversary that can actively influence the global model to extract the gradient of the targeted sample from batches.

In PGLC, we let the adversary push updates towards the malicious direction of the targeted sample's gradients. When the data owner runs its local gradient descent, the malicious updates that are injected into the global model will cause an abrupt magnification of the gradient on the targeted sample while the gradients of other untargeted samples are kept unchanged in the batch. The adversary can thus exploit such differences to obtain the targeted sample's gradients. Particularly, when the untargeted samples' gradients are small, the batch gradients will be dominated by the targeted sample's gradients. This further eases the attack as it allows the adversary to conduct the gradient inversion on the batch gradients directly to restore the targeted sample.

To illustrate this, let Φ_G denote the global parameters of a DNN model and $\mathcal{L}$ denote the loss function. Let (X, Y) be a batch B of data samples with $X = \{x_i\}_{i=1}^{|B|}$ and $Y = \{y_i\}_{i=1}^{|B|}$. The batch gradient δ_{batch} is given by:

$$\delta_{batch} = \frac{1}{|B|} \sum_{i=1}^{|B|} \nabla_{\Phi_G} \mathcal{L}((x_i, y_i), \Phi_G).$$

Let x_{tar} denote the targeted sample belonging to the targeted class tar. We can express the gradients as follows:

$$\begin{aligned}
\delta_{tar} &= \frac{1}{|B|} \nabla_{\Phi_G} \mathcal{L}((x_{tar}, y_{tar}), \Phi_G), \\
\delta_{main} &= \frac{1}{|B|} \sum_{y_j \neq y_{tar}} \nabla_{\Phi_G} \mathcal{L}((x_i, y_i), \Phi_G), \\
\delta_{batch} &= \delta_{tar} + \delta_{main},
\end{aligned}$$

where $main$ means other classes that are not targeted.

If the gradient δ_{tar} corresponding to the targeted sample is significantly larger than δ_{main}, i.e.,

$$\delta_{tar} \gg \delta_{main},$$

then we can approximate the batch gradient as:

$$\delta_{batch} \approx \delta_{tar},$$

indicating that δ_{tar} dominates the aggregated gradient.

To achieve this dominance goal, it is possible to find a global model Φ_G to effectively enlarge δ_{tar}. Therefore, the optimization target is

$$\arg\max_{\Phi_G} \left(\nabla_{\Phi_G} \mathcal{L}((x_{tar}, y_{tar}), \Phi_G) - \sum_{y \neq y_{tar}} \nabla_{\Phi_G} \mathcal{L}((x_i, x_j), \Phi_G) \right).$$

Figure 1 demonstrates differences between gradients on a normal model and gradients on a poisoned model, where t is the training round. In Figure 1 Left, δ_{tar} is similar to any δ_i, and has no obvious relationship with δ_{batch}. However, in Figure 1 Right, δ_{tar} can dominate δ_{batch} on a targeted poisoning model because the δ_{tar} is far larger than $\sum_{j=1, y_j \neq y_{tar}}^{|B|} \delta_j$ so that the adversary can directly use δ_{batch} as δ_{tar}.

Evade Byzantine-robust AGRs. To evade Byzantine-robust AGRs, we need the malicious updates crafted by the adversary to be close to the updates of the honest clients. To this end, we reduce our PGLC into an optimization problem that minimizes the distance between the to-be-crafted malicious updates and

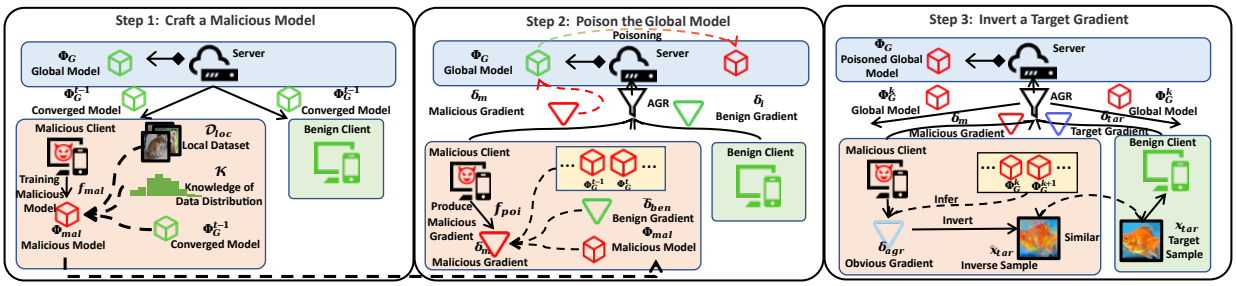


Fig. 2: Overview of PGLC.

the honest clients' updates by covering the maliciousness with an add-on benign gradient while retaining the adversarial impact with respect to the targeted sample. Once the attack successfully evades detection, the malicious impact on the targeted samples will take effect and lead to successful client-side gradient inversion.

B. Overview of PGLC

Figure 2 shows the overview of PGLC, which consists of three steps.

Step 1. Craft a malicious model locally.

In this step, the adversary aims to craft a local malicious model that maximizes the difference between the targeted gradient and the others. Let

$$\Omega = \frac{\nabla_{\Phi_G}\mathcal{L}((x_{tar}, y_{tar}), \Phi_G)}{\nabla_{\Phi_G}\mathcal{L}((x_{tar}, y_{tar}), \Phi_G) + \sum_{y_j \neq y_{tar}} \nabla_{\Phi_G}\mathcal{L}((x_i, y_i), \Phi_G)}.$$

Since

$$\nabla_{\Phi_G}\mathcal{L}((x, y), \Phi_G) \approx \frac{\Delta\mathcal{L}((x, y), \Phi_G)}{\Delta\Phi_G}, \textit{(Taylor series)}$$

we have

$$\Omega \approx \mathcal{L}_{gap} = \frac{\Delta\mathcal{L}((x_{tar}, y_{tar}), \Phi_G)}{\Delta\mathcal{L}((x_{tar}, y_{tar}), \Phi_G) + \Delta \sum\limits_{y_j \neq y_{tar}} \mathcal{L}((x_i, y_j), \Phi_G)}. \tag{1}$$

To satisfy the first-order Taylor expansion, we presume that the loss function is adequately smooth and differentiable, allowing for local linear approximation.

This leads to an optimization problem where the goal is to find a Φ_G that maximizes $\Delta\mathcal{L}((x_{tar}, y_{tar}), \Phi_G)$ while keeping $\Delta \sum_{y_j \neq y_{tar}} \mathcal{L}((x_i, y_j), \Phi_G)$ small:

$$\arg\max_{\Phi_G} \left(\Delta\mathcal{L}((x_{tar}, y_{tar}), \Phi_G) - \Delta \sum_{y_j \neq y_{tar}} \mathcal{L}((x_i, y_j), \Phi_G) \right). \tag{2}$$

To achieve this, we require the model is nearly converged, that is, $\Delta \sum_{y_j \neq y_{tar}} \mathcal{L}((x_i, y_j), \Phi_G)$ is approaching 0. This indicates that the poisoning attack should be launched during the later stages of training.

Therefore, the adversarial optimization objective becomes:

$$\arg\max_{\Phi_G} \left(\mathcal{L}((x_{tar}, y_{tar}), \Phi_G) - \sum_{y_j \neq y_{tar}} \mathcal{L}((x_i, y_j), \Phi_G) \right). \tag{3}$$

We denote a local malicious model that maximizes the loss of the targeted sample as:

$$\Phi_{mal} = f_{mal}(\Phi_G^{t-1}, \mathcal{K}, D_{loc}), \tag{4}$$

where Φ_G^{t-1} is the global model at round $t-1$ (before poisoning), $\mathcal{K}$ represents the knowledge of training data distribution (aforementioned in Section III-C) and D_{loc} denotes a local training dataset held by the adversary. The function f_{mal} outputs Φ_{mal} which performs normally on untargeted samples but fails on the targeted samples. We propose different f_{mal} for various $\mathcal{K}$ accordingly for *full knowledge*, *semi knowledge* and *no knowledge* of data distribution, respectively, detailed later in Section IV-C.

Step 2. Poison the global model. In this step, the adversary aims to submit a malicious update δ_m that contains the maliciousness of Φ_{mal} while breaking the AGRs. This can be characterized by

$$\delta_m = f_{poi}(\Phi_G^{t-1}, \Phi_G^t, \Phi_{mal}, \overline{\delta}_{ben}), \tag{5}$$

where $\overline{\delta}_{ben}$ represents the average of benign gradients from the malicious clients, and Φ_G^{t-1} and Φ_G^t are used to approximate the aggregated gradients of the server in the absence of adversary. f_{poi} denotes an optimization process that minimizes the difference of δ_m and other clients' updates while retaining the effect of Φ_{mal}. The adversary can produce and submit the malicious update in multiple training rounds until the global model has been successfully poisoned. Detailed construction of δ_m and the optimization process are given in Section IV-D.

Step 3. Invert the targeted gradient. The final step is to invert the gradients of the targeted sample from its owner. The adversary cannot control the input of other benign clients and cannot select targeted samples (i.e. the adversary does not know if the current round exists any targeted sample), therefore, he/she has to continuously invert each round aggregated gradient passively. The adversary uses a gradient inversion function to restore the targeted sample as:

$$\hat{x}_{tar} = f_{inv}(\Phi_G^k, \Phi_G^{k+1}, \cdots), \tag{6}$$

where $(\Phi_G^k, \Phi_G^{k+1}, \cdots)$ denote the poisoned global models from round k ($k \geq t$) onwards. If a certain training round contains x_{tar}, its gradients will dominate the aggregated updates (i.e., the difference between the global model of the current and the previous round). This then enables the adversary to restore the targeted sample from the aggregated updates. Detailed explanation of f_{inv} are given in Section IV-E. We also provide notation tables in Appendix A.

C. Craft the Malicious Model

In this section, we discuss the methods of crafting the malicious model based on the adversary's different levels of knowledge.

1) Full Knowledge ($\mathcal{K}_{full}$): In this setting, the adversary does not require the global model from the server and all operations can run locally. He/She can directly synthesize an auxiliary dataset D_{aux} that has the same distribution as $\mathcal{K}_{full}$. Then D_{aux} can represent the entire data distribution, thus D_{loc} can be ignored.

The adversary first removes the samples of the targeted class from D_{aux}. We denote this dataset without the targeted class as D_{main}. Training on D_{main} can ensure the local model to achieve the normal performance on the untargeted classes while learning nothing about the targeted class.

To further enlarge the performance gap between the targeted class and the untargeted classes, the adversary additionally prepares a dataset called D_{trap} which consists of the samples from D_{main} but falsely labeled by the targeted class y_{tar}. We conduct gradient ascent when training on D_{trap}. As such, the loss with respect to the targeted class is significantly increased as the local model is updated towards the opposite direction of predicting an arbitrary sample to the targeted class (that is, the local model will never classify any samples to the targeted class). In addition, such training on D_{trap} in the gradient ascent manner will not affect the performance of the untargeted classes as well.

To craft the malicious model, the adversary iteratively trains on D_{main} and D_{trap} for r_{end} rounds that the adversary can set, detailed in Algorithm 1.

Algorithm 1 Crafting the Mal. Model with Full Knowledge.

Input:
$\mathcal{K}_{full}$: the full knowledge of the data distribution of all classes; Φ^0: the initialized model; r_{end}: the number of local training rounds; α_1: the normal training learning rate; α_2: the trap training learning rate; y_{tar}: the targeted class;
Output:
Φ_{mal}: the targeted poisoning model

Construct $D_{aux} \sim \mathcal{K}_{full}$
Construct $D_{main} = \{(x_i, y_i)\}$ from D_{aux}
Construct $D_{trap} = \{(x_i, y_{tar})\}$ from D_{aux}
Initialize Φ^0
for $r \leftarrow 0$ to r_{end} **do**
  ▷ Minimize untargeted classes loss.
  $\tilde{\Phi}^{r+1} = \Phi^r - \alpha_1 \nabla_{\Phi^r} \mathcal{L}(\Phi^r, D_{main})$
  ▷ Maximize targeted class loss.
  $\Phi^{r+1} = \tilde{\Phi}^{r+1} + \alpha_2 \nabla_{\tilde{\Phi}^{r+1}} \mathcal{L}(\tilde{\Phi}^{r+1}, D_{trap})$
end for
$\Phi_{mal} \leftarrow \Phi^{r_{end}}$
return Φ_{mal}

2) Semi Knowledge ($\mathcal{K}_{semi}$): In this setting, the adversary first pulls the nearly converged global model Φ_G^{t-1} from the server (the adversary can wait for sufficient training rounds or estimate global model convergence based on the local validation dataset collected), then he/she is able to generate a D_{tar} that matches the distribution of the targeted class only based on $\mathcal{K}_{semi}$. To craft the malicious model, the adversary first calculates the gradients with respect to the targeted class based on D_{tar}. Then he/she pushes the local updates towards the ascending direction of the loss of targeted samples (also known as the targeted model poisoning [32]).

However, such loss on the targeted class will damage model generalization on the untargeted classes to some extent. Therefore, we let the adversary construct a D_{ben} from its own local datasets D_{loc}. The D_{ben} only contains the adversary's untargeted samples for the main classification task, i.e., the samples of the targeted class are removed. He/she then uses it to repair the lost cross-class features. Finally the entire optimization objective is The optimization objectives are following:

$$\begin{aligned} &\arg\max_{\Phi_{mal}} \mathcal{L}(\Phi, D_{tar}), \\ &\arg\min_{\Phi_{mal}} \mathcal{L}(\Phi, D_{ben}). \end{aligned} \tag{7}$$

The detailed algorithm is given in Algorithm 2.

Algorithm 2 Crafting the Mal. Model with Semi Knowledge.

Input:
$\mathcal{K}_{semi}$: the semi-knowledge of the data distribution of the targeted class; D_{loc}: the local dataset; Φ_G^{t-1}: the converged global model; r_{end}: the number of local training rounds; α_1: the poisoning training learning rate; α_2: the repair training learning rate; y_{tar}: the targeted class;
Output:
Φ_{mal}: the targeted poisoning model

Construct $D_{tar} = \{(x_{tar}, y_{tar})\} \sim \mathcal{K}_{semi}$
Construct $D_{ben} = \{(x_i, y_i)\}$ from D_{loc}
Initialize Φ^0
for $r \leftarrow 0$ to r_{end} **do**
  ▷ Poisoning targeted class.
  $\tilde{\Phi}^{r+1} = \Phi^r + \alpha_1 \nabla_{\Phi^r} \mathcal{L}(\Phi^r, D_{tar})$
  ▷ Repair untargeted classes performance.
  $\Phi^{r+1} = \tilde{\Phi}^{r+1} - \alpha_2 \nabla_{\tilde{\Phi}^{r+1}} \mathcal{L}(\tilde{\Phi}^{r+1}, D_{ben})$
end for
$\Phi_{mal} \leftarrow \Phi^{r_{end}}$
return Φ_{mal}

3) No Knowledge ($\mathcal{K}_{no}$): In this setting, the adversary does not have any samples of the targeted class. In order to craft the desired malicious model against the targeted class, the adversary must first obtain some knowledge about it. To this end, we let the adversary conduct model inversion [4], [33], [7] against the converged global model Φ_G^{t-1}, aiming to reconstruct some samples from the targeted class by solving

$$\begin{aligned} \hat{X}_{tar} = \arg\min_{\hat{X}} \; & \lambda_0 \mathcal{L}(\Phi_G^{t-1}, (\hat{X}, Y_{tar})) \\ & + \lambda_1 \mathcal{R}_{TV}(\hat{X}) + \lambda_2 \mathcal{R}_{l_2}(\hat{X}) + \lambda_3 \mathcal{R}_f(\hat{X}), \end{aligned} \tag{8}$$

where $\hat{X}_{tar}$ represents reconstructed samples of targeted class, Y_{tar} is the targeted class label array, and λ_i ($i \in \{0, 1, 2, 3\}$) is the coefficient. Here, $\mathcal{R}_{TV}$ and $\mathcal{R}_{l_2}$ are loss functions that regularize the total variation and l_2 norm of $\hat{X}$ and $\mathcal{R}_f$ is to ensure feature similarities between the mean and variance of

$\hat{X}$ and the global model's batch normalization layer, which is formalized as:

$$\begin{aligned}\mathcal{R}_f(\hat{X}) = &\sum_l ||\mu_l(\hat{X}) - BN_l(mean)||_2 \\ &+ \sum_l ||\sigma_l^2(\hat{X}) - BN_l(variance)||_2.\end{aligned} \tag{9}$$

Here we point out that the required batch normalization layer information is different from those required by GradInverion [8]. GradInverion [8] requires batch-wise mean and variance of local models but our method only needs the batch normalization layer information from the global model and such information is insensitive to specific batch input.

We note that due to the limited knowledge owned by the client-side adversary, the reconstructed samples $\hat{X}_{tar}$ in this step can be far away from the real training samples: they can be vague, distorted and contain many noises. However, the information about the common features contained in $\hat{X}_{tar}$ can be utilized by the adversary to perturb the local model toward the malicious direction against the targeted class.

Next, following the method in Section IV-C2 (i.e., Semi knowledge), the adversary constructs the malicious model by ascending the loss of targeted samples based on $\hat{D}_{tar}$ constructed by $\hat{X}_{tar}$ and repairs the performance of the untargeted classes using his/her local datasets D_{loc} (no targeted samples because of $\mathcal{K}_{no}$). The optimization objectives are following:

$$\begin{aligned}&\arg\max_{\Phi_{mal}} \mathcal{L}(\Phi, (\hat{X}_{tar}, Y_{tar})), \\ &\arg\min_{\Phi_{mal}} \mathcal{L}(\Phi, D_{loc}).\end{aligned} \tag{10}$$

The training algorithm is detailed in Algorithm 3.

Algorithm 3 Crafting the Mal. Model with No Knowledge.

Input:

D_{loc}: the local dataset based on no knowledge; Φ_G^{t-1}: the converged global model; p_{end}: the number of model inversion rounds; r_{end}: the number of local training rounds; λ_i $(i \in \{0,1,2,3\})$: the coefficients in model inversion; α_1: the poisoning training learning rate; α_2: the repair training learning rate; y_{tar}: the targeted class;

Output:

Φ_{mal}: the targeted poisoning model

Initialize $\hat{X}_{tar}$

Define $\mathcal{L}_{MI}(\hat{X}) = \lambda_0\mathcal{L}(\Phi_G, (\hat{X}, Y_{tar})) + \lambda_1\mathcal{R}_{TV}(\hat{X}) + \lambda_2\mathcal{R}_{l_2}(\hat{X}) + \lambda_3\mathcal{R}_f(\hat{X})$

▷ Inverting global model to obtain target samples.

for $p \leftarrow 0$ to p_{end} **do**

  $\hat{X}_{tar}^{p+1} = \hat{X}_{tar}^{p} - \nabla_{\hat{X}_{tar}^{p}} \mathcal{L}_{MI}(\hat{X}_{tar}^{p})$

end for

$\hat{X}_{tar} \leftarrow \hat{X}_{tar}^{p_{end}}$

Construct $\hat{D}_{tar} = \{(\hat{x}_{tar}, y_{tar})\}$

Initialize Φ^0

for $r \leftarrow 0$ to r_{end} **do**

  ▷ Poisoning target class.

  $\tilde{\Phi}^{r+1} = \Phi^r + \alpha_1 \nabla_{\Phi^r} \mathcal{L}(\Phi^r, \hat{D}_{tar})$

  ▷ Repairing untargeted classes performance.

  $\Phi^{r+1} = \tilde{\Phi}^{r+1} - \alpha_2 \nabla_{\tilde{\Phi}^{r+1}} \mathcal{L}(\tilde{\Phi}^{r+1}, D_{loc})$

end for

$\Phi_{mal} \leftarrow \Phi^{r_{end}}$

return Φ_{mal}

D. Poison the Global Model

In this section, we show the process of replacing the global model with the crafted malicious model while defeating the AGRs.

We define the replacement problem below. Let

$$\delta_{poi} = \Phi_G^t - \Phi_{mal}$$

be the updates that the adversary aims to inject into the global model. Clearly, if there is no AGR equipped, the adversary can directly commit δ_{poi} as his malicious update δ_m to boost attack efficiency.

In order to evade the AGRs, the adversary needs to minimize the distance between δ_m and the current round aggregated gradient δ_{agr}. Thus, the adversary's objective becomes

$$\arg\min_{\delta_m} f_{dis}(\delta_m, \delta_{agr}), \tag{11}$$

where f_{dis} is a distance metric function. However, the above formula is hard to solve due to the variety of AGRs and the adversary cannot obtain the current round aggregated gradient δ_{agr}. Inspired by the literature of optimizing a traditional poisoning attack [20], [24], [27], we express it in a different form that is better suited for optimization and universal for all AGRs.

The adversary first calculates the aggregated updates of the previous round as

$$\delta_{agr} = \frac{1}{\alpha}(\Phi_G^{t-1} - \Phi_G^t).$$

Then, he/she constructs δ_m as

$$\delta_m = \gamma \frac{\delta_{poi}}{||\delta_{poi}||} + \beta\overline{\delta}_{ben},$$

where $\overline{\delta}_{ben}$ represents the average of benign gradients from malicious clients (i.e., calculated by malicious clients on their local datasets). It can be calculated from the local dataset D_{loc} of every malicious client. The parameters γ, β are the scaling coefficients for the unitary malicious gradient and reference gradient, which is initialized as γ_0 and β_0 (and they are tunable). Therefore, Equation (11) can be transformed as:

$$\arg\min_{\gamma,\ \beta} ||\delta_m - \delta_{agr}||_2, \tag{12}$$

by which the adversary minimizes the distance between malicious update δ_m and aggregated gradients δ_{agr} to break AGRs. Then the adversary still uses SGD to optimize γ, β. The optimization of the malicious update is described in Algorithm 4.

E. Invert the Targeted Gradient

In this section, we describe the method of restoring the targeted sample from the poisoned global models.

Algorithm 4 Optimizing Malicious Update

Input:

Φ_G^t: the global model at t round; Φ_G^{t-1}: the global model at $t-1$ round; Φ_{mal}: the malicious model; $\overline{\delta}_{ben}$: the benign gradient; γ_0: the initialized coefficient of malicious gradient; β_0: the initialized coefficient of benign gradient; r_{end}: the number of optimization rounds; α: the learning rate of the server; α_1: the learning rate for γ; α_2: the learning rate for β;

Output:

δ_m: the malicious update

▷ Calculating the aggregation gradient of the last round.

$\delta_{agr} = \frac{1}{\alpha}(\Phi_G^{t-1} - \Phi_G^t)$

▷ Optimizing the malicious update δ_m.

Initialize γ^0, β^0

Initialize $\delta_m^0 \leftarrow \gamma^0 \frac{\delta_{poi}}{||\delta_{poi}||} + \beta^0 \overline{\delta}_{ben}$

Define $\mathcal{L}_{dis}(\delta_m) = ||\delta_m - \delta_{agr}||_2$

for $r \leftarrow 0$ to r_{end} **do**

  $\gamma^{r+1} = \gamma^r - \alpha_1 \nabla_{\gamma^r} \mathcal{L}_{dis}(\delta_m^r)$

  $\beta^{r+1} = \beta^r - \alpha_2 \nabla_{\beta^r} \mathcal{L}_{dis}(\delta_m^r)$

  $\delta_m^{r+1} = \gamma^{r+1} \frac{\delta_{poi}}{||\delta_{poi}||} + \beta^{r+1} \overline{\delta}_{ben}$

end for

$\delta_m \leftarrow \delta_m^{r_{end}}$

return δ_m

The adversary first pulls the poisoned global models $(\Phi_G^k, \Phi_G^{k+1}, \cdots)$ from round k ($k \geq t$) onwards. He/she then calculates the aggregated updates of each round as:

$$\delta_{agr}^k = \frac{1}{\alpha}(\Phi_G^{k+1} - \Phi_G^k).$$

Next, the adversary conducts gradient inversion [6], [5] against each of δ_{agr}^k as follows. First, the adversary initializes the targeted sample $\hat{x}_{tar}$ with random noise and constructs dummy gradients by

$$\hat{\delta}_{tar} = \nabla_{\Phi_G^k} \mathcal{L}(\hat{x}_{tar}, y_{tar}). \tag{13}$$

Then the adversary aims to restore $\hat{x}_{tar}$ by solving:

$$\arg\min_{\hat{x}_{tar}} \lambda_0 (1 - \frac{< \hat{\delta}_{tar}, \delta_{agr}^k >}{||\hat{\delta}_{tar}|| ||\delta_{agr}^k||}) + \lambda_1 R_{TV}(\hat{x}_{tar}) + \lambda_2 R_{l_2}(\hat{x}_{tar}) + \lambda_3 R_{clip}(\hat{x}_{tar}), \tag{14}$$

by which the adversary finds an optimal $\hat{x}_{tar}$ that maximizes the cosine similarity of $\hat{\delta}_{tar}$ and δ_{agr}^k. The parameter λ_i ($i \in \{0, 1, 2, 3\}$) is the coefficients, and R_{TV}, R_{l_2} and R_{clip} are the regularization terms of the total variation, the l_2 norm and the clip item, respectively [6]. These regularization techniques are frequently employed in image generation to constrain image vectors within their original feature space [9], [5], [6], [8].

Since the adversary is unaware of whether the aggregated gradient of the current round encompasses the targeted gradient, they must consistently apply inversion attacks to every aggregated gradient to restore the targeted samples.

V. Experimental Settings

A. Datasets and Model Architecture

We employ Cifar100 [34], TinyImageNet [35], and CalTech256 [36] as benchmark datasets, utilizing Resnet18 [37] for the model structure. In our experiment, we adjust the input data to align with the original dataset dimensions (e.g., 32×32 for Cifar100, 64×46 for TinyImageNet, 112×112 for CalTech256). Further details on the datasets are provided in Supplemental Document B.1.

B. Implementation Settings

FL settings. We set the number of clients N varying from 10 to 50 in our experiments. The proportion of malicious clients is set to around 20% following the common setting in prior research [24], [20], [23]. We use m to denote the number of malicious clients. The batch size $|B|$ is set to range from 8 to 32. More detailed settings can be found in Supplemental Document B.2.

Evaluated AGRs. In our experiments, we evaluate the following four representative Byzantine-robust AGRs: **Multi-krum** [17], **Bulyan** [18], **Adaptive federated average (AFA)** [19], **Fang defenses** [20]. The details about these AGRs are given in Supplemental Document B.3.

C. Baselines and Evaluation Metrics

Baselines. We compare the PGLC with the following baselines: **DLG** [5], **iDLG** [9], **IG** [6]. They are the state-of-the-art model/gradient inversion attacks that can be launched from the client side in FL. Details of these baseline methods are given in Supplemental Document B.4.

Evaluation metrics. We use three metrics: **PSNR**, **LPIPS** [38], **SSIM** [39]. More details have been presented in Supplemental Document B.5.

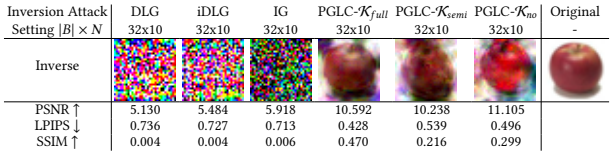
VI. Evaluation Results

A. PGLC *Baseline Performance*

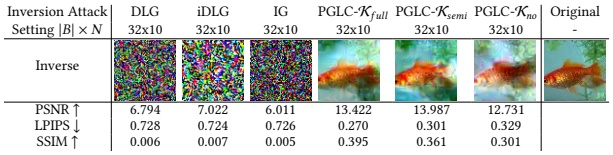
Figure 3 shows the comparison of our PGLC with DLG [5], iDLG [9] and IG [6]. We do not equip any Byzantine-robust AGRs in this set of experiments for a fair comparison with the baselines. The results demonstrate that only PGLC can invert target samples successfully as a client-side adversary when the batch size is set to 32 on CIFAR100 and TinyImageNet and 8 on CalTech256.

B. PGLC *against Byzantine-robust AGRs*

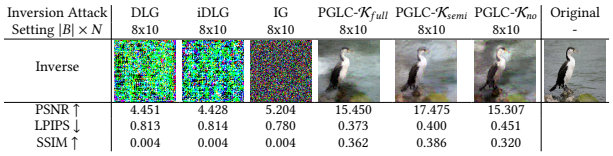
The PGLC's performance against AGRs is illustrated in Figure 4. In general, PGLC is capable of defeating all evaluated robust AGRs across various datasets. While some restored images may contain noise, the essential semantic features of the targeted images can still be revealed through our attack. The reconstructed image quality is not highly sensitive to the choice of robust AGRs as we consistently observe PSNR values exceeding 10.00, LPIPS scores below 0.60, and SSIM values above 0.2 across all settings and datasets. For example, our PGLC achieves LIPIS(↓) values of 0.377, 0.358,



(a) Cifar100.



(b) TinyImageNet.



(c) CalTech256.

Fig. 3: Comparison of PGLC with Baseline Methods.

0.350, 0.462, and 0.387 against *NoAGR*, *Multi-krum*, *Bulyan*, *AFA*, and *Fang*, respectively, across all knowledge levels and datasets in average. We observe that the LPIPS value on AFA is slightly higher, while the LPIPS values on other robust AGRs are similar. This can be explained by the success of our attack relying on the targeted gradient's ability to dominate the aggregated gradient. Since AFA uses a weighted average aggregation method, the influence of the targeted gradient is slightly impaired during aggregation, resulting in slightly worse performance compared to other AGRs. In this set of experiments, we only consider the cases in which one targeted sample appears in a single aggregation. The study of multiple targeted samples co-occurring in a single aggregation can be found in Section VII.

We also investigate the global model's training accuracy with/without our PGLC, as demonstrated in Table I. In general, our PGLC does not impact on the model's overall test performance, demonstrating the stealthiness of our approach. Considering different levels of adversary knowledge, the $\mathcal{K}_{full}$ adversary possesses all data distributions which enables it to achieve higher test accuracy.

TABLE I: Accuracy Performance of the Global Model with/without PGLC

AGR	Dataset											
	Cifar100				TinyImageNet				CalTech256			
	No Attack	$\mathcal{K}_{full}$	$\mathcal{K}_{semi}$	$\mathcal{K}_{no}$	No Attack	$\mathcal{K}_{full}$	$\mathcal{K}_{semi}$	$\mathcal{K}_{no}$	No Attack	$\mathcal{K}_{full}$	$\mathcal{K}_{semi}$	$\mathcal{K}_{no}$
Training Acc.(%)												
No Robust AGR	99.982	95.358	96.522	98.978	99.978	97.188	99.335	99.260	99.941	96.072	99.477	99.444
Multi-Krum	99.982	95.354	95.534	98.976	99.976	97.201	99.285	99.151	99.924	95.831	99.461	99.187
Bulyan	99.982	95.342	93.816	98.980	99.976	97.212	99.333	99.330	99.928	96.050	99.452	99.351
AFA	99.982	95.920	97.680	98.978	99.978	96.883	99.323	99.459	99.941	63.851	98.672	98.849
Fang	99.982	95.382	94.318	98.978	99.977	97.223	99.310	99.367	99.928	91.178	99.469	99.541
Test Acc.(%)												
No Robust AGR	69.760	70.840	62.030	64.870	51.230	53.110	47.330	47.570	68.412	68.791	66.314	66.447
Multi-Krum	69.800	71.050	61.220	64.570	51.280	53.140	47.260	47.210	68.395	66.611	66.380	65.935
Bulyan	69.700	71.070	60.080	66.640	51.170	53.060	47.150	47.750	68.263	68.742	66.083	66.199
AFA	69.520	70.550	64.620	64.760	51.370	49.130	47.180	49.130	68.147	46.598	60.992	64.746
Fang	69.730	70.970	61.100	64.860	51.300	53.260	47.290	48.020	68.610	58.966	66.380	67.834

C. Ablation Studies

In the ablation study, we will present how the malicious model crafting stage and the poisoning stage influence PGLC.

Global Model Poisoning Performance. We first investigate the performance of model poisoning. During poisoning, the adversary's objective is to replace the global model with a malicious model. To quantify the extent of poisoning, we employ the model distance as the poisoning measurement. We regard the distances between the global models and malicious models before poisoning as the initial distance and after poisoning as the final distance, which is calculated as $\sqrt{\sum_{l_0=1}^{l}(\phi_G - \phi_{mal})^2/d_\phi}$ where l represents the number of layers and d_ϕ is the parameter dimension. When the final replacement distance is approaching zero, we can regard the poisoning processing as successful.

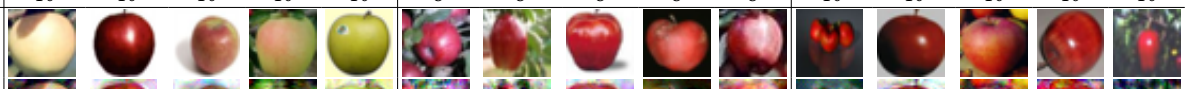
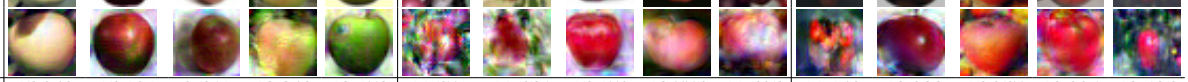
We present the results in Table II. Our model poisoning strategy succeeds in all scenarios. Among the three knowledge levels, the adversary with semi-knowledge achieves the smallest initialized model distance (61.046×10^{-3}, 89.778×10^{-3} and 40.019×10^{-3} on Cifar100, TinyImageNet and CalTech256 respectively), followed by the adversary with no-knowledge (1087.533×10^{-3}, 128.167×10^{-3} and 217.541×10^{-3} on Cifar100, TinyImageNet and CalTech256 respectively). The full-knowledge adversary has the largest initialized model distance (2157.039×10^{-3}, 5082.850×10^{-3} and 288.970×10^{-3} on Cifar100, TinyImageNet and CalTech256 respectively), meaning that it requires more poisoning iterations than the other two to make the final poisoned model close to the malicious model.

Comparing different AGRs, we observe that Fang and AFA are more robust as the final distances of AFA and Fang are also generally more distinct than other AGRs. This is due to the use of the validation dataset. Particularly, in the extreme case that the validation dataset only consists of samples from the target class, our poisoning will fail.

The effectiveness of gradient inversion in PGLC is influenced by both the performance of malicious model crafting and global model poisoning in practical environments. The malicious models crafted in step 1 serve as the foundation for PGLC, as the loss gap between the targeted class and the untargeted classes determines whether the targeted gradient can dominate the aggregated gradient. For poisoning the global model (step 2), different AGRs determine the difficulty of model replacement, which determines whether the malicious updates can be successfully injected into the global model.

Adversary Knowledge Evaluation. We investigate how different levels of the adversary's knowledge about the data distribution affect the attack's performance.

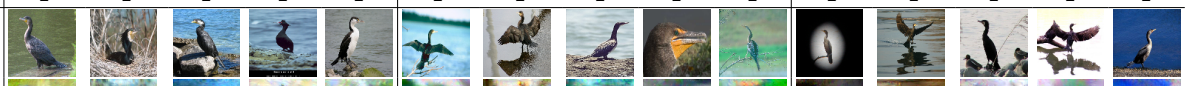
We fix the batch size in each set of comparisons among the three knowledge levels. As shown in Figure 5, the full knowledge adversary achieves the best performance, and its performance remains stable with the increasing number of clients. The adversary with semi-knowledge and no-knowledge's averaged performance are similar, while being unstable in some cases (e.g., Cifar100 inversion results under 20 and 30 clients). Basically, the full knowledge adversary has a larger loss gap which enables the targeted gradient on the malicious

Knowledge Level		Full Knowledge					Semi Knowledge					No Knowledge				
AGR		NoAGR	MKrum	Bulyan	AFA	Fang	NoAGR	MKrum	Bulyan	AFA	Fang	NoAGR	MKrum	Bulyan	AFA	Fang
Setting	$\|B\| \times N$	32x50	32x50	32x51	32x50	32x50	32x40	32x40	32x41	32x40	32x40	32x50	32x50	32x51	32x50	32x50
	m	10	10	10	10	10	8	8	8	8	8	10	10	10	10	10
Original																
Inverse																
PSNR ↑		15.040	15.101	9.561	13.342	13.178	11.571	11.556	15.179	13.775	11.505	11.142	13.983	15.957	12.906	13.635
LPIPS ↓		0.108	0.118	0.221	0.372	0.160	0.414	0.326	0.210	0.203	0.488	0.356	0.245	0.172	0.529	0.230
SSIM ↑		0.708	0.516	0.458	0.401	0.507	0.227	0.217	0.561	0.643	0.320	0.296	0.418	0.644	0.121	0.416

(a) PGLC on Cifar100

Knowledge Level		Full Knowledge					Semi Knowledge					No Knowledge				
AGR		NoAGR	MKrum	Bulyan	AFA	Fang	NoAGR	MKrum	Bulyan	AFA	Fang	NoAGR	MKrum	Bulyan	AFA	Fang
Setting	$\|B\| \times N$	32x40	32x40	32x41	32x40	32x40	32x30	32x30	32x31	32x30	32x30	32x40	32x40	32x41	32x40	32x40
	m	8	8	8	8	8	6	6	6	6	6	8	8	8	8	8
Original																
Inverse																
PSNR ↑		12.139	17.271	16.161	15.600	13.529	16.338	14.516	15.958	12.437	11.954	11.721	14.651	15.654	14.430	13.103
LPIPS ↓		0.314	0.270	0.325	0.352	0.354	0.333	0.299	0.300	0.452	0.312	0.445	0.319	0.427	0.455	0.446
SSIM ↑		0.367	0.506	0.494	0.529	0.361	0.315	0.258	0.408	0.238	0.298	0.312	0.340	0.372	0.332	0.242

(b) PGLC on TinyImageNet

Knowledge Level		Full Knowledge					Semi Knowledge					No Knowledge				
AGR		NoAGR	MKrum	Bulyan	AFA	Fang	NoAGR	MKrum	Bulyan	AFA	Fang	NoAGR	MKrum	Bulyan	AFA	Fang
Setting	$\|B\| \times N$	8x10	8x10	8x11	8x10	8x10	8x10	8x10	8x11	8x10	8x10	8x10	8x10	8x11	8x10	8x10
	m	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
Original																
Inverse																
PSNR ↑		12.377	12.525	13.326	9.161	13.018	11.676	10.994	12.112	12.065	14.78	10.771	11.422	13.741	9.513	10.999
LPIPS ↓		0.463	0.514	0.485	0.668	0.617	0.491	0.627	0.517	0.552	0.420	0.470	0.511	0.497	0.580	0.464
SSIM ↑		0.256	0.173	0.160	0.156	0.194	0.304	0.189	0.246	0.286	0.318	0.196	0.249	0.269	0.300	0.331

(c) PGLC on CalTech256

Fig. 4: PGLC's Performance against AGRs.

model trained with full knowledge difficult to be diluted during aggregation. Therefore, the full knowledge malicious model can work on larger FL settings. In comparison to a full-knowledge adversary, semi-knowledge and no-knowledge methods utilize targeted class knowledge to poison models. However, the limited knowledge of the targeted class can pose challenges. This limitation can result in scenarios where the targeted class loss is not significant enough which may impair inversion performance, or the untargeted classes loss increases due to the potential damage to generalization features. Thus, it can be noted that malicious models created with semi or no knowledge may exhibit instability in some instances.

VII. Scalability Evaluation

We then conduct scalability evaluation to demonstrate the generality of PGLC in varied experimental settings.

Targeted Sample's Distribution Scalability. We also consider the case in which more than one targeted sample appears in one aggregation. To this end, we study the effect of the targeted sample distribution and investigate two typical settings: the first one is that all targeted samples belong to a single client (single-client distribution), and the second one is that the targeted samples are distributed uniformly at random among multiple clients (multi-client distribution). We fix the number of clients as 40, the batch size as 32, and the number of malicious clients as 8. The FL is equipped with Multi-krum [17] and we assume the adversary has full knowledge.

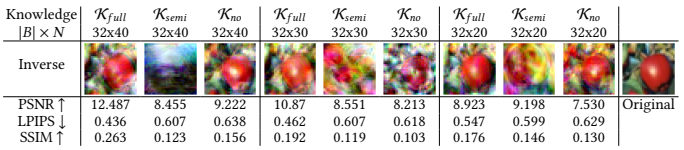
Figure 6 illustrates the results. In general, our PGLC succeeds in restoring multiple targeted samples from one aggregation. However, we observe some transfers of semantic features among the restored samples of which their corresponding original samples are similar because their gradients are averaged during the aggregation.

We also find that such transfers of semantic features are more likely to appear in single-client distribution. This is due to the fact that targeted samples are aggregated locally, which happens prior to the global aggregation. When all targeted samples are grouped together in a single batch and processed by a single client, the batch data will go through the

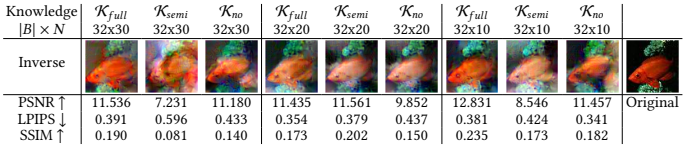
TABLE II: The Performance of the Global Model Poisoning.

AGR	Setting		Metric	
	$\|B\| \times N$	m	I.M. Dis.[1](10^{-3})	F.M. Dis.[2](10^{-3})
Cifar100 - $\mathcal{K}_{full}$				
No Robust AGR	32x50	10	2157.039	3.203
Multi-Krum	32x50	10	2157.039	0.446
Bulyan	32x51	10	2157.039	0.039
AFA	32x50	10	2157.039	318.045
Fang	32x50	10	2157.039	5.944
Cifar100 - $\mathcal{K}_{semi}$				
No Robust AGR	32x40	8	61.046	1.798
Multi-Krum	32x40	8	61.046	0.473
Bulyan	32x41	8	61.046	0.254
AFA	32x40	8	61.046	17.086
Fang	32x40	8	61.046	4.437
Cifar100 - $\mathcal{K}_{no}$				
No Robust AGR	32x50	10	1087.533	2.823
Multi-Krum	32x50	10	1087.533	0.018
Bulyan	32x51	10	1087.533	45.807
AFA	32x50	10	1087.533	2.427
Fang	32x50	10	1087.533	2.958
TinyImageNet - $\mathcal{K}_{full}$				
No Robust AGR	32x40	8	5082.850	1.800
Multi-Krum	32x40	8	5082.850	0.091
Bulyan	32x41	8	5082.850	23.094
AFA	32x40	8	5082.850	39.341
Fang	32x40	8	5082.850	14.497
TinyImageNet - $\mathcal{K}_{semi}$				
No Robust AGR	32x30	6	89.778	1.025
Multi-Krum	32x30	6	89.778	0.078
Bulyan	32x31	6	89.778	2.044
AFA	32x30	6	89.778	0.625
Fang	32x30	6	89.778	0.544
TinyImageNet - $\mathcal{K}_{no}$				
No Robust AGR	32x40	8	128.167	1.367
Multi-Krum	32x40	8	128.167	0.263
Bulyan	32x41	8	128.167	1.878
AFA	32x40	8	128.167	21.990
Fang	32x40	8	128.167	2.622
CalTech256 - $\mathcal{K}_{full}$				
No Robust AGR	8x10	2	288.970	2.448
Multi-Krum	8x10	2	288.970	43.910
Bulyan	8x11	2	288.970	9.523
AFA	8x10	2	288.970	195.884
Fang	8x10	2	288.970	174.439
CalTech256 - $\mathcal{K}_{semi}$				
No Robust AGR	8x10	2	40.019	4.474
Multi-Krum	8x10	2	40.019	0.603
Bulyan	8x11	2	40.019	0.857
AFA	8x10	2	40.019	11.968
Fang	8x10	2	40.019	1.603
CalTech256 - $\mathcal{K}_{no}$				
No Robust AGR	8x10	2	217.541	6.819
Multi-Krum	8x10	2	217.541	0.718
Bulyan	8x11	2	217.541	9.314
AFA	8x10	2	217.541	11.477
Fang	8x10	2	217.541	22.017

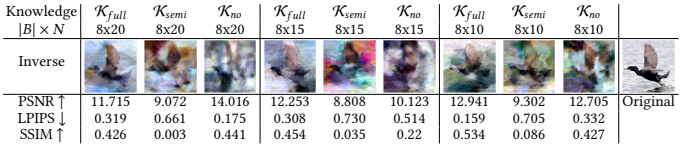
[1] "I.M.Dis." means "Initialized Model Distance".
[2] "F.M.Dis." means "Final Model Distance".



(a) Cifar100



(b) TinyImageNet



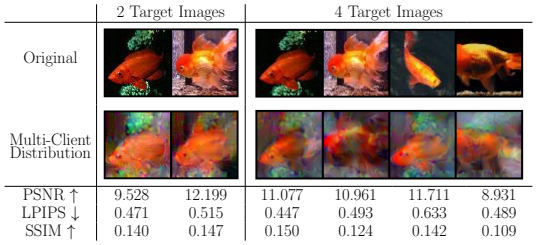
(c) CalTech256

Fig. 5: PGLC's Performance with Different Levels of Knowledge.

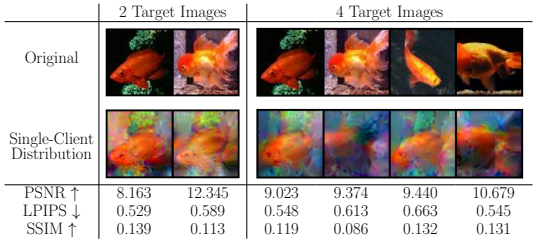
same set of batch normalization layers. As a result, the batch normalization layers facilitate the transfer of similar image features by reducing internal covariate shifts and stabilizing the feature distribution.

FL System Scalability. In PGLC, the effect of the injected maliciousness with respect to the targeted sample is affected by the batch size and the number of clients as the sum of the local updates will gradually dominate the aggregated gradient and dilute the targeted gradient when the number of training samples increases in one aggregation. We thus conduct the following experiments to explore the impact.

Figure 7 shows the PGLC's performance with the varying number of batch sizes in one aggregation of 50 clients. The experimental setting is fixed with the full-knowledge adversary

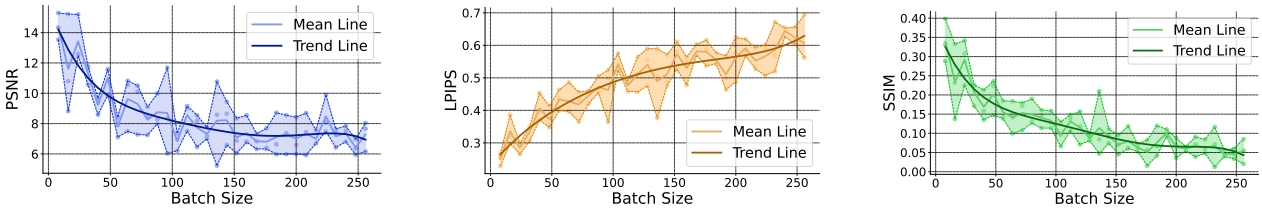


(a) Multiple-client setting



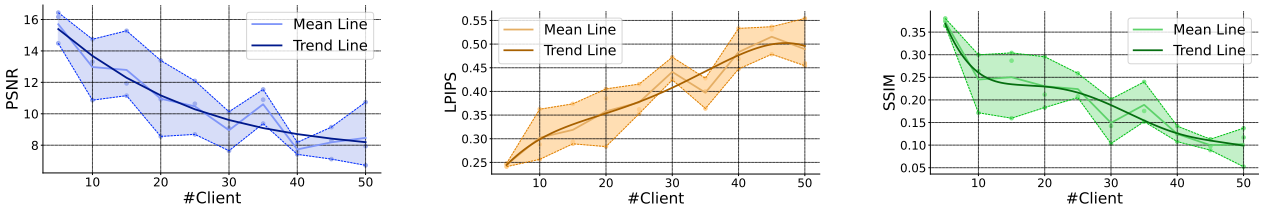
(b) Single-client setting

Fig. 6: PGLC's Performance of Restoring Multiple Targeted Samples on TinyImageNet.



(a) PSNR (b) LPIPS (c) SSIM

Fig. 7: PSNR, LPIPS, and SSIM of PGLC against the Batch Size in the Aggregation of 50 Clients on TinyImageNet.



(a) PSNR (b) LPIPS (c) SSIM

Fig. 8: PSNR, LPIPS, and SSIM of PGLC against the Client Number with Batch Size 128 on TinyImageNet.

without any robust AGRs equipped. We can find that the performance is going down as the batch size increases. It is worth noting that in this set of experiments, the level of maliciousness is kept fixed, resulting in a fixed loss gap. However, an adaptive attacker can actively increase the level of maliciousness in order to enlarge the loss gap, especially for larger batch sizes. By doing so, the attacker can maintain a high level of attack performance.

We also perform experiments by fixing the batch size and evaluating across different client numbers, as shown in Figure 8, with a similar performance trend.

Data Heterogeneity (Non-IID). We adhere to the Non-IID setting in the work [19]. We assume that there are M classes and N clients. We randomly allocate M classes into N clients with a probability q of assigning a training example with label l to a particular client and a probability $\frac{1-q}{N-1}$ of assigning it to other clients. When $q = \frac{1}{N}$, the clients' local training data are independent and identically distributed (IID), otherwise the clients' local training data are Non-IID. A larger q indicates a higher Non-IID degree.

We conduct experiments on TinyImageNet dataset with 200 classes and assume a $\mathcal{K}_{semi}$ adversary. We set $N = 30$, meaning that if $q = \frac{1}{30} \simeq 0.033$, the distribution is IID. We select $q = 0.033, 0.3, 0.6, 0.9$, and explore PGLC performance on these settings.

Figure 9 illustrates the restored images. In general, our PGLC can successfully restore images with meaningful semantics, even in the Non-IID setting with extremely high degree. However, we observe a performance degradation as the Non-IID degree increases. We further analyze the performance of model poisoning using the initial/final distance defined in Section VI-C. As shown in Table IV, the final distance slightly increases with a higher degree of Non-IID distribution. This is due to the increased probability of multiple targeted samples co-occurring within a single aggregation.

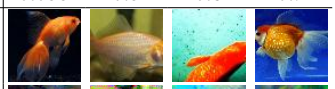
Knowledge Level	Semi Knowledge			
AGR	MultiKrum			
Setting $\lvert B\rvert \times N$ m	32x30 6			
q	0.033	0.3	0.6	0.9
Original				
Inverse				
PSNR ↑	14.516	13.750	11.743	9.884
LPIPS ↓	0.299	0.485	0.432	0.628
SSIM ↑	0.258	0.179	0.260	0.148

Fig. 9: PGLC's Performance with Non-IID Data.

TABLE IV: The Performance of the Global Model Poisoning with Non-IID Data.

AGR	Setting			Metric	
	$\lvert B\rvert \times N$	m	q	I.M. Dis.[1](10^{-3})	F.M. Dis.[2](10^{-3})
Multi-Krum	32x30	6	0.033(IID)	89.778	0.078
	32x30	6	0.3	89.778	0.140
	32x30	6	0.6	89.778	1.467
	32x30	6	0.9	89.778	4.110

[1] "I.M.Dis." means "Initialized Model Distance".
[2] "F.M.Dis." means "Final Model Distance".
[3] IID FL

Different Model Architecture Performance. To prove PGLC's validity in different model architectures. We choose to explore VGG-11 [40] as a different architecture and Resnet50 [37] as a larger model.

In this supplementary experiment, we examine the 'no-knowledge' scenario and perform the corresponding experiments on TinyImageNet. The outcomes are depicted in Figure 10 and Figure 11. All inversion results showcase valid semantic features. Almost all reconstructed images achieved SSIM values greater than 0.1, PSNR values greater than 9,

and LPIPS values below 0.66. Our experiments indicate that PGLC can be applied to various network architectures.

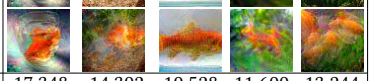
Knowledge Level		No Knowledge				
AGR		NoAGR	MKrum	Bulyan	AFA	Fang
Setting	$\lvert B\rvert \times N$	128x20	128x20	128x21	128x20	128x20
	m	6	6	6	6	6
Original						
Inverse						
PSNR ↑		17.348	14.392	19.528	11.699	13.244
LPIPS ↓		0.250	0.326	0.269	0.435	0.489
SSIM ↑		0.423	0.203	0.512	0.146	0.159

Fig. 10: PGLC's Performance on VGG11.

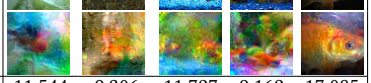
Knowledge Level		No Knowledge				
AGR		NoAGR	MKrum	Bulyan	AFA	Fang
Setting	$\lvert B\rvert \times N$	64x30	64x30	64x30	64x30	64x30
	m	6	6	6	6	6
Original						
Inverse						
PSNR ↑		11.544	9.306	11.727	9.162	17.085
LPIPS ↓		0.387	0.660	0.403	0.558	0.219
SSIM ↑		0.258	0.101	0.170	0.105	0.405

Fig. 11: PGLC's Performance on Resnet50.

Malicious Client Scalability. Considering the adversary may control limited malicious clients, we explore PGLC's performance with the poisoning ratio from 7.5% to 20% on MKrum [17]. Essentially, our additional experiment results indicate that our poisoning attack can succeed with a poisoning ratio as low as 7.5% under the 0.01 threshold. The details can be found in Supplemental Document C.1.

Effect of Random Client Selection. Random selection of clients is a very common condition in FL. In our method, this condition only impacts the poisoning stage (Step 2), and because the radio of malicious clients remains unchanged. Thus, the poisoning attack still can succeed. Detailed results and analysis are given in Supplemental Document C.2.

Input Dimension Scalability. Generally, data with higher dimensionality are more challenging to reconstruct, as high-dimensional images often contain more intricate details, making it harder to accurately recover the original images. Therefore, we explore PGLC on the CalTech256 dataset from an image size of 112×112 to 224×224. Basically, the inversion attack is still effective. Detailed results and analysis are given in Supplemental Document C.3.

VIII. Feasibility Analysis

Based on the mathematical framework discussed in Section IV-B, it is evident that $\mathcal{L}_{gap}$ in Equation 1 determines the success of PGLC. Consequently, this section details experiments undertaken to examine the impact of the loss gap and utilizes experimental results to validate our initial hypothesis.

We denote $\mathcal{L}_{main} = \sum_{y_j \neq y_{tar}} \mathcal{L}((x_i, y_j), \Phi_G)$ and $\mathcal{L}_{tar} = \mathcal{L}((x_{tar}, y_{tar}), \Phi_G)$. The term $\mathcal{L}_{gap}$ essentially measures the ratio of $\Delta\mathcal{L}_{tar}$ to $\Delta\mathcal{L}_{tar} + \Delta\mathcal{L}_{main}$. Here, $\Delta\mathcal{L}$ indicates the difference in loss when comparing samples before and after one step optimization. Thus, the value of $\mathcal{L}_{gap}$ can reflect the extent of dominance. Consequently, $\mathcal{L}_{gap}$ ranges from 0 to 1, and as the total loss becomes increasingly influenced by $\Delta\mathcal{L}_{tar}$, $\mathcal{L}_{gap}$ approaches 1.

We perform our experiments on the TinyImageNet dataset [35] using 40 clients with a batch size of 32. The adversary has full knowledge. Figure 12 illustrates the impact of the loss gap on the effectiveness of PGLC. When the $\mathcal{L}_{gap}$ is minimal, the targeted gradient becomes entangled with features from other samples, which diminishes PGLC's efficiency. As $\mathcal{L}_{gap}$ increases, PGLC's effectiveness also improves, as the targeted loss becomes more distinguished. When $\mathcal{L}_{gap}$ nears 1, indicating the gradient primarily comprises the targeted gradient, the optimal inversion performance is attained. Among various metrics, PSNR displays greater variability due to the stochastic nature of the inversion attack potentially altering the reconstructed images' background, whereas LPIPS and SSIM remain relatively steady.

IX. Defenses and Mitigation Strategies

PGLC represents a novel client-side attack in FL. Consequently, defending against this type of attack remains an open topic. In this section, we will analyze other existing privacy-preserving methods and their effectiveness in mitigating our proposed attack. Additionally, we also will propose potential directions for developing defense mechanisms.

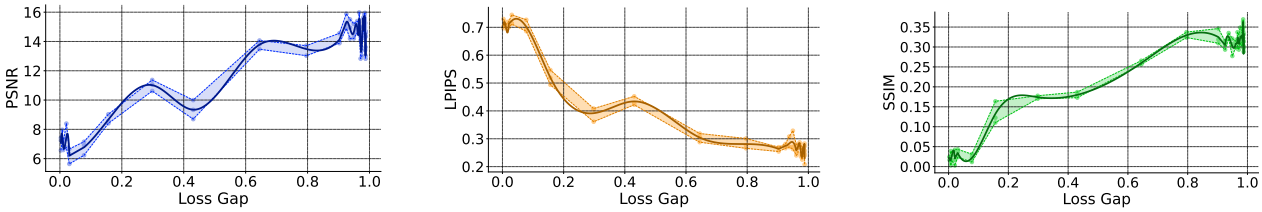
A. Existing Privacy-Preserving Technologies

Secure Multi-Party Computation (SMPC) [41]. SMPC allows multiple clients to collaboratively perform computations (like aggregating gradients or updating weights) without revealing their individual updates to others, even to the central server. However, SMPC solution is not effective in defending against our attack. Although secure aggregation can prevent local updates from exposure, the aggregated update remains disclosed. In PGLC, once the global model has been poisoned, the targeted gradient can still be exposed.

De-duplicating Training [42]. Reducing the repeated occurrence of data samples in training (de-duplication) helps prevent models from memorizing specific features, thereby minimizing privacy leaks. Nevertheless, defense strategies employing de-duplication of training data cannot stop our attack, as it does not depend on memorization.A specific sample can be reconstructed by PGLC even if it occurs just once.

Differential Privacy (DP) [43], [44], [45], [46]. Differential Privacy (DP) is a widely used technology that seeks to implement mechanisms ensuring outputs are indistinguishable. Following the (ϵ, δ)-DP setting [43], we set the clipping norm bound (C) to 1, while varying ϵ with values of $1, 5, 10$ and δ to 1×10^{-5}. The experiment result is given in the Figure 13.

PGLC demonstrates its effectiveness under the evaluated (ϵ, δ)-DP. We remark that, with $(1, 10^{-5})$-DP, the test accuracy of the model is significantly reduced to only 0.450% (meaning that the utility of the model is destroyed by the added noise),



(a) PSNR (b) LPIPS (c) SSIM

Fig. 12: PGLC's Performance on TinyImageNet against the Varying Loss Gap.

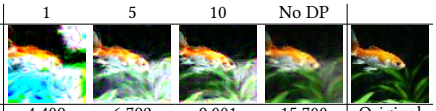
Epsilon	1	5	10	No DP	
Inverse					
PSNR↑	4.409	6.792	9.001	15.700	Original
LPIPS↓	0.540	0.357	0.307	0.198	
SSIM↑	0.138	0.239	0.317	0.456	
Test Acc. (%)	0.450	19.270	24.960	48.770	
Test Loss	5.488	3.629	3.311	3.024	

Fig. 13: PGLC's Performance with DP.

while our PGLC is still able to extract meaningful semantics from the targeted sample.

Therefore, differential privacy is ineffective in defending our attack. Our experimental results demonstrate that our PGLC can effectively restore the privacy input from the gradients, even in cases where a substantial amount of added noise has compromised the model's utility.

Adaptive Defense [47]. Adaptive Defense can customize privacy protection for different rounds of communication and clients. Therefore, the quality of the reconstruction may degrade with greater noise, as demonstrated in Figure 13. However, this kind of adaptive defense mechanism cannot ensure that reconstructed inputs will not disclose useful information. Additionally, this kind of adaptive defense should carefully consider model divergence and utility facing our attack.

B. Mitigation Strategies

To mitigate PGLC, we suggest employing cooperative AGRs as a strategy to prevent poisoning. In the collaborating AGRs, the server cooperates with clients to mitigate attacks. Clients can locally identify threats and notify the server, or they might exchange common data representations. This allows the server to control the training framework, which might enhance the privacy budget. However, it is crucial to carefully consider and address the potential impact of heterogeneity among the data samples. In addition, not all clients are honest. It is beneficial to build a balanced risk awareness framework to comprehensively assess risks in the training stage.

X. Limitations

Regarding limitations, PGLC possesses an uncertain time for successful attack. The PGLC demonstrates ambiguity regarding the presence of examples from the targeted class within the inputs received from other clients that are not malicious. Consequently, the attacker might experience an indeterminate waiting period before the appearance of targeted class examples in the inputs of these benign clients. This results in an unpredictable attack timeline for PGLC.

Moreover, to ensure the success of a poisoning attack, a sufficient poisoning ratio (7.5%, See Supplemental Document C.1) may be necessary. This may require some specific conditions. In small-scale systems, such as IoT-based FL networks, limited clients and weak defense mechanisms make it easier for adversaries to gain control over a subset of devices or a central node, potentially compromising the entire system. In large-scale FL systems, adversaries may be required to launch Sybil attacks to create numerous virtual devices to increase the poisoning ratio. Moreover, client selection techniques can be manipulated to enhance the probability of selecting malicious clients. In conclusion, achieving a sufficient poisoning ratio might need particular conditions or additional attacks.

XI. Conclusion

We introduced PGLC, a stealthy client-side method enabling adversaries with minimal FL knowledge to reconstruct training data by inverting batch gradients, even bypassing Byzantine-robust AGRs. Experiments confirm its effectiveness, highlighting the need for stronger defenses. Future work will focus on understanding how gradients carry data features, developing evaluation tools, and designing targeted defenses to mitigate inversion attacks in FL systems.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. PMLR, Apr. 2017, pp. 1273–1282, iSSN: 2640-3498. [Online]. Available: https://proceedings.mlr.press/v54/mcmahan17a.html

[2] J. Konečný, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated Optimization: Distributed Machine Learning for On-Device Intelligence," Oct. 2016, arXiv:1610.02527 [cs]. [Online]. Available: http://arxiv.org/abs/1610.02527

[3] R. Shokri and V. Shmatikov, "Privacy-Preserving Deep Learning," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15. New York, NY, USA: Association for Computing Machinery, Oct. 2015, pp. 1310–1321. [Online]. Available: http://doi.org/10.1145/2810103.2813687

[4] M. Fredrikson, S. Jha, and T. Ristenpart, "Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15. New York, NY, USA: Association for Computing Machinery, Oct. 2015, pp. 1322–1333. [Online]. Available: http://doi.org/10.1145/2810103.2813677

[5] L. Zhu, Z. Liu, and S. Han, "Deep Leakage from Gradients," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/hash/60a6c4002cc7b29142def8871531281a-Abstract.html

[6] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting Gradients - How easy is it to break privacy in federated learning?" in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 16 937–16 947. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/c4ede56bbd98819ae6112b20ac6bf145-Abstract.html

[7] H. Yin, P. Molchanov, J. M. Alvarez, Z. Li, A. Mallya, D. Hoiem, N. K. Jha, and J. Kautz, "Dreaming to Distill: Data-Free Knowledge Transfer via DeepInversion," 2020, pp. 8715–8724. [Online]. Available: https://openaccess.thecvf.com/content_CVPR_2020/html/Yin_Dreaming_to_Distill_Data-Free_Knowledge_Transfer_via_DeepInversion_CVPR_2020_paper.html

[8] H. Yin, A. Mallya, A. Vahdat, J. M. Alvarez, J. Kautz, and P. Molchanov, "See Through Gradients: Image Batch Recovery via GradInversion," 2021, pp. 16 337–16 346. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2021/html/Yin_See_Through_Gradients_Image_Batch_Recovery_via_GradInversion_CVPR_2021_paper.html

[9] B. Zhao, K. R. Mopuri, and H. Bilen, "iDLG: Improved Deep Leakage from Gradients," Jan. 2020, arXiv:2001.02610 [cs, stat]. [Online]. Available: http://arxiv.org/abs/2001.02610

[10] Y. Wen, J. Geiping, L. Fowl, M. Goldblum, and T. Goldstein, "Fishing for User Data in Large-Batch Federated Learning via Gradient Magnification," Jun. 2022, arXiv:2202.00580 [cs]. [Online]. Available: http://arxiv.org/abs/2202.00580

[11] L. Fowl, J. Geiping, W. Czaja, M. Goldblum, and T. Goldstein, "Robbing the Fed: Directly Obtaining Private Data in Federated Learning with Modified Models," Mar. 2022, arXiv:2110.13057 [cs]. [Online]. Available: http://arxiv.org/abs/2110.13057

[12] F. Boenisch, A. Dziedzic, R. Schuster, A. S. Shamsabadi, I. Shumailov, and N. Papernot, "When the Curious Abandon Honesty: Federated Learning Is Not Private," Dec. 2021, arXiv:2112.02918 [cs]. [Online]. Available: http://arxiv.org/abs/2112.02918

[13] J. Jeon, j. Kim, K. Lee, S. Oh, and J. Ok, "Gradient Inversion with Generative Image Prior," in *Advances in Neural Information Processing Systems*, vol. 34. Curran Associates, Inc., 2021, pp. 29 898–29 908. [Online]. Available: https://proceedings.neurips.cc/paper/2021/hash/fa84632d742f2729dc32ce8cb5d49733-Abstract.html

[14] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu, "Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning," in *Proceedings of the 2020 USENIX Annual Technical Conference (USENIX ATC 2020)*, 2020.

[15] D. Usynin, D. Rueckert, and G. Kaissis, "Beyond Gradients: Exploiting Adversarial Priors in Model Inversion Attacks," Mar. 2022, arXiv:2203.00481 [cs]. [Online]. Available: http://arxiv.org/abs/2203.00481

[16] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and Improving the Image Quality of StyleGAN," 2020, pp. 8110–8119. [Online]. Available: https://openaccess.thecvf.com/content_CVPR_2020/html/Karras_Analyzing_and_Improving_the_Image_Quality_of_StyleGAN_CVPR_2020_paper.html

[17] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent," in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/hash/f4b9ec30ad9f68f89b29639786cb62ef-Abstract.html

[18] E. M. E. Mhamdi, R. Guerraoui, and S. Rouault, "The Hidden Vulnerability of Distributed Learning in Byzantium," in *Proceedings of the 35th International Conference on Machine Learning*. PMLR, Jul. 2018, pp. 3521–3530, iSSN: 2640-3498. [Online]. Available: https://proceedings.mlr.press/v80/mhamdi18a.html

[19] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "FLTrust: Byzantine-robust Federated Learning via Trust Bootstrapping," Apr. 2022, arXiv:2012.13995 [cs]. [Online]. Available: http://arxiv.org/abs/2012.13995

[20] M. Fang, X. Cao, J. Jia, and N. Gong, "Local Model Poisoning Attacks to {Byzantine-Robust} Federated Learning," 2020, pp. 1605–1622. [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presentation/fang

[21] C. Xie, O. Koyejo, and I. Gupta, "Generalized Byzantine-tolerant SGD," Mar. 2018, arXiv:1802.10116 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1802.10116

[22] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates," in *Proceedings of the 35th International Conference on Machine Learning*. PMLR, Jul. 2018, pp. 5650–5659, iSSN: 2640-3498. [Online]. Available: https://proceedings.mlr.press/v80/yin18a.html

[23] G. Baruch, M. Baruch, and Y. Goldberg, "A Little Is Enough: Circumventing Defenses For Distributed Learning," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/hash/ec1c59141046cd1866bbbcdfb6ae31d4-Abstract.html

[24] V. Shejwalkar and A. Houmansadr, "Manipulating the Byzantine: Optimizing Model Poisoning Attacks and Defenses for Federated Learning," in *Proceedings 2021 Network and Distributed System Security Symposium*. Virtual: Internet Society, 2021. [Online]. Available: https://www.ndss-symposium.org/wp-content/uploads/ndss2021_6C-3_24498_paper.pdf

[25] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How To Backdoor Federated Learning," in *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*. PMLR, Jun. 2020, pp. 2938–2948, iSSN: 2640-3498. [Online]. Available: https://proceedings.mlr.press/v108/bagdasaryan20a.html

[26] H. Wang, K. Sreenivasan, S. Rajput, H. Vishwakarma, S. Agarwal, J.-y. Sohn, K. Lee, and D. Papailiopoulos, "Attack of the Tails: Yes, You Really Can Backdoor Federated Learning," in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 16 070–16 084. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/b8ffa41d4e492f0fad2f13e29e1762eb-Abstract.html

[27] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing Federated Learning through an Adversarial Lens," in *Proceedings of the 36th International Conference on Machine Learning*. PMLR, May 2019, pp. 634–643, iSSN: 2640-3498. [Online]. Available: https://proceedings.mlr.press/v97/bhagoji19a.html

[28] M. Kahla, S. Chen, H. A. Just, and R. Jia, "Label-Only Model Inversion Attacks via Boundary Repulsion," 2022.

[29] "Plug & Play Attacks: Towards Robust and Flexible Model Inversion Attacks," Jun. 2022.

[30] J. Zhu and M. B. Blaschko, "R-GAP: Recursive Gradient Attack on Privacy," Feb. 2022. [Online]. Available: https://openreview.net/forum?id=RSU17UoKfJF

[31] L. Shen, Y. Zhang, J. Wang, and G. Bai, "Better together: Attaining the triad of byzantine-robust federated learning via local update amplification," in *Proceedings of the 38th Annual Computer Security Applications Conference*, 2022, pp. 201–213.

[32] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli, "Towards Poisoning of Deep Learning Algorithms with Back-gradient Optimization," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, ser. AISec '17. New York, NY, USA: Association for Computing Machinery, Nov. 2017, pp. 27–38. [Online]. Available: http://doi.org/10.1145/3128572.3140451

[33] D. Ye, H. Chen, S. Zhou, T. Zhu, W. Zhou, and S. Ji, "Model Inversion Attack against Transfer Learning: Inverting a Model without Accessing It," Mar. 2022, arXiv:2203.06570 [cs]. [Online]. Available: http://arxiv.org/abs/2203.06570

[34] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[35] M. A. mnmoustafa, "Tiny imagenet," 2017. [Online]. Available: https://kaggle.com/competitions/tiny-imagenet

[36] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," 2007.

[37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[38] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 586–595.

[39] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[40] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[41] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*,

ser. CCS '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 1175–1191.

[42] N. Kandpal, E. Wallace, and C. Raffel, "Deduplicating training data mitigates privacy risks in language models," in *Proceedings of the 39th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, Eds., vol. 162. PMLR, 17–23 Jul 2022, pp. 10 697–10 707.

[43] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 308–318.

[44] N. Wang, X. Xiao, Y. Yang, J. Zhao, S. C. Hui, H. Shin, J. Shin, and G. Yu, "Collecting and analyzing multidimensional data with local differential privacy," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 2019, pp. 638–649.

[45] L. Sun, J. Qian, and X. Chen, "Ldp-fl: Practical private aggregation in federated learning with local differential privacy," *arXiv preprint arXiv:2007.15789*, 2020.

[46] S. Truex, L. Liu, K.-H. Chow, M. E. Gursoy, and W. Wei, "Ldp-fed: Federated learning with local differential privacy," in *Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking*, 2020, pp. 61–66.

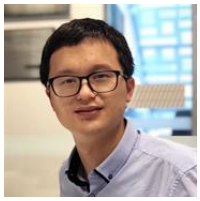
[47] J. Hu, Z. Wang, Y. Shen, B. Lin, P. Sun, X. Pang, J. Liu, and K. Ren, "Shield against gradient leakage attacks: Adaptive privacy-preserving federated learning," *IEEE/ACM Transactions on Networking*, 2023.

Jiaheng Wei is a Ph.D. student from RMIT University. He received his bachelor's degree and honors bachelor's degree from Deakin University in 2021 and 2022. His research interests include machine learning security and federated learning privacy.

Yanjun Zhang is currently a Lecturer at the University of Technology Sydney, Australia. She received her PhD degree in computer science from the University of Queensland, in 2021. Her research interests include cybersecurity and privacy, trustworthy machine learning, and software engineering.

Leo Yu Zhang (Member, IEEE) is currently a Senior Lecturer, Griffith University, Brisbane, QLD 4215, Australia. He received the bachelor's and master's degrees in computational mathematics from Xiangtan University in 2009 and 2012, and the Ph.D. degree from the City University of Hong Kong, in 2016. His research interests include applied cryptography and AI related security.

Chao Chen (Member, IEEE) is currently a Senior Lecturer, RMIT University, Australia. He received the Ph.D. degree in computer science from Deakin University, Geelong, VIC, Australia, in 2017. His research interests include network traffic classification, insider threat detection, and machine learning security, such as inference attacks on machine learning (ML) models.



Shirui Pan (Senior Member, IEEE) is currently a Professor and an ARC Future Fellow with the School of Information and Communication Technology, Griffith University, Brisbane, QLD 4215, Australia. He received the Ph.D. degree in computer science from the University of Technology Sydney, Ultimo, NSW, Australia, in 2015. He was a Lecturer with the School of Software, UTS. He has authored or coauthored over 80 research papers in top-tier journals and conferences, including the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, and KDD. His research interests include data mining and machine learning.

Kok-Leong Ong is a Professor of Business Analytics in the College of Business & Law, RMIT University. He received the BASc (Hons) and PhD degrees from the Nanyang Technological University, Singapore, in 2000 and 2004, respectively. He is currently working on a range of analytics projects making data actionable through analytics-2-business translation, automation, and applications.

Jun Zhang (Senior Member, IEEE) is the cofounder and director with the Cybersecurity Lab, Swinburne University of Technology, Australia. He received the PhD degree in computer science from the University of Wollongong, Australia. His research interests include cybersecurity and applied machine learning. He is the chief investigator of several projects in cybersecurity, funded by the Australian Research Council (ARC). He has published more than 100 research papers in many international journals and conferences.

Yang Xiang (Fellow, IEEE) is currently a full professor and the dean of Digital Research & Innovation Capability Platform, Swinburne University of Technology, Australia. He received the PhD degree in computer science from Deakin University, Australia. His research interests include cyber security, which covers network and system security, data analytics, distributed systems, and networking. In particular, he is currently leading his team developing active defense systems against large-scale distributed network attacks. He is the chief investigator of several projects in network and system security, funded by the Australian Research Council (ARC). He has published more than 200 research papers in many international journals and conferences. He served as an associate editor of the IEEE Transactions on Computers, the IEEE Transactions on Parallel and Distributed Systems, Security and Communication Networks (Wiley), and an editor of the Journal of Network and Computer Applications. He is a fellow of the IEEE.