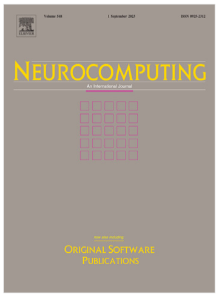
Journal Pre-proof



A survey on vulnerability of federated learning: A learning algorithm perspective

Xianghua Xie, Chen Hu, Hanchi Ren, Jingjing Deng

PII: S0925-2312(23)01348-6
DOI: https://doi.org/10.1016/j.neucom.2023.127225
Reference: NEUCOM 127225

To appear in: *Neurocomputing*

Received date : 18 September 2023
Revised date : 28 November 2023
Accepted date : 30 December 2023

Please cite this article as: X. Xie, C. Hu, H. Ren et al., A survey on vulnerability of federated learning: A learning algorithm perspective, *Neurocomputing* (2024), doi: https://doi.org/10.1016/j.neucom.2023.127225.

A Survey on Vulnerability of Federated Learning: A Learning Algorithm Perspective

Xianghua Xie[a,*], Chen Hu[a], Hanchi Ren[a], Jingjing Deng[b,*]

[a]Department of Computer Science, Swansea University, United Kingdom
[b]Department of Computer Science, Durham University, United Kingdom

Abstract

Federated Learning (FL) has emerged as a powerful paradigm for training Machine Learning (ML), particularly Deep Learning (DL) models on multiple devices or servers while maintaining data localized at owners' sites. Without centralizing data, FL holds promise for scenarios where data integrity, privacy and security and are critical. However, this decentralized training process also opens up new avenues for opponents to launch unique attacks, where it has been becoming an urgent need to understand the vulnerabilities and corresponding defense mechanisms from a learning algorithm perspective. This review paper takes a comprehensive look at malicious attacks against FL, categorizing them from new perspectives on attack origins and targets, and providing insights into their methodology and impact. In this survey, we focus on threat models targeting the learning process of FL systems. Based on the source and target of the attack, we categorize existing threat models into four types, Data to Model (D2M), Model to Data (M2D), Model to Model (M2M) and composite attacks. For each attack type, we discuss the defense strategies proposed, highlighting their effectiveness, assumptions and potential areas for improvement. Defense strategies have evolved from using a singular metric to excluding malicious clients, to employing a multifaceted approach examining client models at various phases. In this survey paper, our research indicates that the to-learn data, the learning gradients, and the learned model at different stages all can be manipulated to initiate malicious attacks that range from undermining model performance, reconstructing private local data, and to inserting backdoors. We have also seen these threat are becoming more insidious. While earlier studies typically amplified malicious gradients, recent endeavors subtly alter the least significant weights in local models to bypass defense measures. This literature review provides a holistic understanding of the current FL threat landscape and highlights the importance of developing robust, efficient, and privacy-preserving defenses to ensure the safe and trusted adoption of FL in real-world applications. The categorized bibliography can be found at: https://github.com/Rand2AI/Awesome-Vulnerability-of-Federated-Learning.

Keywords: Federated Learning, Deep Learning, Model Vulnerability, Privacy Preserving

1. Introduction

In the era of Artifical Intelligence (AI) that is built upon big data, the need to extract valuable insights from massive amounts of information is driving innovation across industries. Achievements of data-driven Deep Learning (DL) models have been witnessed in many areas, ranging from Natural Language Processing (NLP) [1, 2, 3] to visual computing [4, 5, 6, 7]. It is generally agreed upon that the more training data, the greater potential performance of the model. To illustrate, the research work [8] claims if one were able to collect data from all medical facilities, models trained on such dataset would have the potential of "answering many significant questions", such as drug discovery and predictive modeling of diseases. Data centralization scheme for training AI model has been the predominant method for decades. However, methods solely relying on centralized training scheme are becoming less viable, not only due to the cost of computational resources, but more importantly, the growing concerns related to privacy and security, which has triggered the need for alternative learning paradigms. Federated Learning (FL) [9, 10], a distributed learning paradigm emerges as a pioneering solution to address these challenges, where multiple decentralized parties collaborate on a learning task while the data remains with its owner. In contrast to traditional approaches, where all data has to be centralized, FL stemming from the increasing concerns on data privacy allows model to be trained at the source of data creation. This innovative approach not only minimizes the risk of data leakage, maintains the privacy of sensitive information, but also lifts the computational burden of cloud centers, which is considered as a potential alternative for completing multi-party learning in many domains, such as: healthcare [11, 12, 13], finance [14, 15, 16], smart cities [17, 18, 19] and autonomous driving [20, 21, 22]. We observed that there is a significant growth related to FL in both academic research and industrial applications.

Recent studies on exploiting vulnerabilities of FL, have illuminated the fact that the robustness of FL architectures is not as secure as expected, where each building block in FL algorithms, ranging from its data distribution, communication mechanisms, to aggregation processes, is susceptible to malicious attacks [23, 24, 25, 26]. These vulnerabilities can potentially compromise the privacy and security of the participants, meanwhile downgrade the integrity and effectiveness of

*Corresponding Authors: X. Xie (x.xie@swansea.ac.uk) and J. Deng (jingjing.deng@durham.ac.uk).

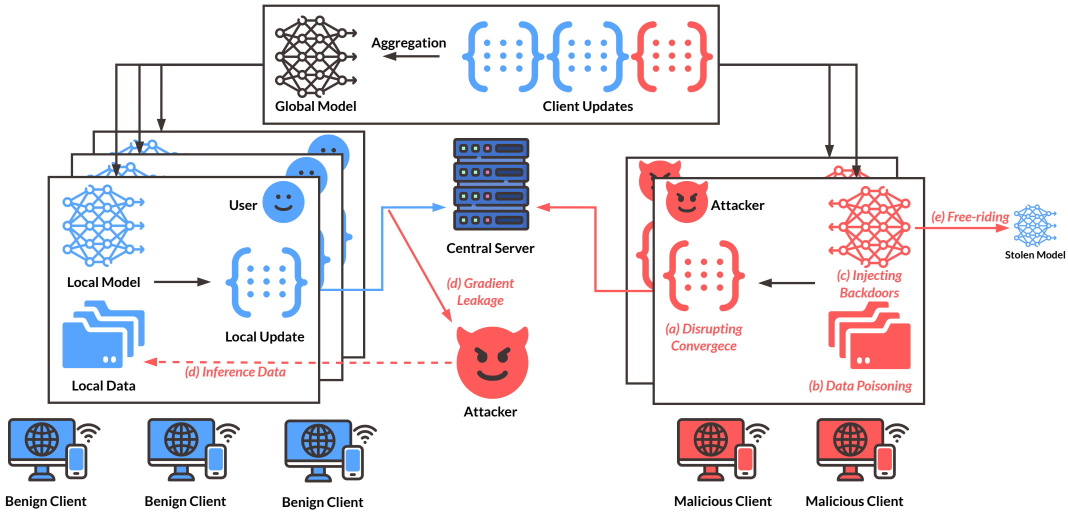


Figure 1: An Overview of Common Vulnerabilities in FL. Malicious attackers can: (a) manipulate model updates to prevent the global model from converging; (b) tamper data labels to induce erroneous predictions after training; (c) inject backdoors into the global model; (d) reconstruct data or inference data properties by eavesdropping model updates; (e) steal the global model while contribute nothing.

the entire learning system. Figure 1 illustrates various common FL attacks and provides a comprehensive overview on different stages and components in the FL that can be targeted by opponents. Specifically, a variety of tactics that a malicious attacker can employ, as follows:

- **Data Tampering:** By disrupting data label or introducing sample noisy the adversary misguides the global model making inaccurate or biased predictions.

- **Model Manipulation:** By changing the model weight during aggregation, the attacker forces the global model to deviate from the desirable convergence. It can be a subtle change over time, or a drastic disruption that leads to significant performance degradation.

- **Data Reconstruction:** By exploring the gradient information or model weight, the opponent attempts to reconstruct or infer specific attributes of the original data, thereby breaching the privacy of data owner.

- **Backdoor Injection:** By embedding backdoor into the global model, the contestant deceives the trained model to give designated prediction when the corresponding trigger pattern in the input is presented.

Despite the promising future of FL aimed at alleviating privacy concerns, FL still faces a wide variety of threats. In contrast to reviewing FL from system and network security perspectives, in this survey, we focus on retrospecting the research advancements of FL vulnerability that is inherited from the nature of machine learning algorithms. As shown in Figure 1, we identify that a malicious attacker can attack every component in the FL system. For example, an opponent may masquerade as a participating client of the system and provide toxic data to degrade the prediction performance of the global model, or intercept client updates and inject backdoor or reconstruct private training data. In this paper, we propose a taxonomy of FL attacks centered around attack origins and attack targets, which are outlined in Table 1. Our taxonomy of FL attacks emphasizes exploited vulnerabilities and their direct victims. For instance, label-flipping is a typical D2M attack, often described as a data poisoning technique. If the local data is tampered by such a designated attack, the trained global model can be compromised by such training data and exhibit anomalous behavior.

The rest of survey is organized as such: In Section 2, we firstly introduce the essential preliminaries of FL algorithm. Then, following the proposed taxonomy, we review each type of attack, including D2M Attack, M2M Attack, M2D Attack and Composite Attack in Section 3, 4, 5 and 6 respectively. Within each section, both threat models and the corresponding defense strategies are presented, compared and discussed. Section 7 concludes our findings and provides our recommendation for future research directions.

2. Preliminaries of Federated Learning

FL can be categorized into horizontal FL, vertical FL, and federated transfer learning, based on how the training data is organized [27]. Since the majority of research on FL vulnerabilities focuses on the horizontal FL setting, therefore, we also focus on horizontal FL as the central topic in this review. FedAvg is the most classic horizontal FL algorithm, where the global model is learned by averaging across all local models trained on clients. Surprisingly, such a simple aggregation scheme has been proven to be effective in many case studies [28, 29, 30], where the convergence is also mathematically sound [31]. Improvements upon FedAvg include incorporating local update corrections [32, 33] or adaptive weighting schemes [34, 35, 36], however, the fundamental aggregation scheme remains similar.

Table 1: Our proposed taxonomy

Type of Attack	Definition	Example
Data to Model (D2M)	tampering the data alone to degrade model performance	label-flipping
Model to Model (M2M)	tampering updates to prevent learning convergence	Byzantine attack
Model to Data (M2D)	intercepting model updates to inference private data information	gradient leakage
Composite (D2M+M2M)	tampering both data and updates to manipulate model behavior	backdoor injection

Therefore, we present FedAvg [10] as an example to demonstrate the potential components in FL system that can be targeted by malicious parties. Firstly, all clients receive the identical global model ω_0 from the central server that is randomly initialized. Then, the local model is trained on each client with its local data. Once the local training steps finish (i.e., the number of pre-set iteration or epoch is reached), individual clients send either the updated local model ω_E or the model difference u to the server. The central server aggregates the global model ω_r by averaging the local models, and send the updated model to each client. To speed up the training, a subset of clients are chosen randomly for the current round of training, which is also considered as a dropout regularization for FL. The pseudo code of original FedAvg algorithm is given in Algorithm 1, where the terms highlighted indicate the entities that can be compromised.

The comparison between surveys on FL attacks and defenses is summarized in Table 2. While most surveys include detailed discussion on defense strategies, some of them only give high-level overviews on threat models, such as explaining the concept of Byzantine attacks (M2M) without delving into diverse attacks as we summarized in Table 4. Our work reviews FL vulnerabilities from the perspective of learning algorithms. Our review includes major threat models that exploits the learning paradigm of FL and discusses defense strategies to counter these threats.

Algorithm 1 FedAvg for Horizontal FL. (*Terms* highlighted are the vulnerable components can be targeted by adversaries.)

n_i is the number of local samples, N_S is the total number of samples among selected clients, D_i is the local training data, ω is model weights

Server:

create and send model to all clients
clients own their respective data D_i
initialize ω_0
for each round $r = 1, 2, ..., R$ **do**
  sample $|S|$ clients, send ω_{r-1} to each clients in S
  for each client $i \in S$ **do**
    ω_r^i or u_i ←Client(i, ω_{r-1})
  end for
  $\omega_r \leftarrow \sum_{i=1}^{|S|} \frac{n_i}{N_S} \omega_r^i$ or $\omega_r \leftarrow \omega_{r-1} + \sum_{i=1}^{|S|} \frac{n_i}{N_S} u_i$
  validate the model with ω_r
end for

Client(i, ω):

for each epoch $e = 1, 2, ..., E$ **do**
  $\omega_e \leftarrow \omega_{e-1} - \eta \cdot \nabla_{\omega_{e-1}} \mathcal{L}(D_i)$
end for
$u \leftarrow \omega_E - \omega$
return ω_E or u to server

3. Data to Model Attacks

We describe Data to Model (D2M) attacks in FL as threat models that are launched by manipulating the local data while the models in training are being targeted as victims. D2M attacks are also considered as black-box attacks because the attackers do not need to access inside information such as client model weights or updates, tampering the data alone is often suffice to launch a D2M attack. However, the attackers can also draw information from local dataset or client models to enhance the effectiveness of D2M attacks. We present the timeline of D2M research in Figure 3. The characteristics of discussed D2M attacks are shown in Table 3.

3.1. D2M Attacks on Class Labels

The D2M attack of poisoning data labels is called label-flipping. Such an attack aims at misleading the training models by feeding tampered labels for training. For instance, the attackers may switch the labels for car images to "planes", resulting in the model to classify car images as planes after training.

Label-flipping attack is first studied and proved its effectiveness in the centralized setting [42]. Later on, [43, 44] demonstrate label-flipping attack in FL scenarios. Theses studies follow [42] and flip the labels from the victim class to a different target class. Authors of [44] show that with only 4% of total clients being malicious, label-flipping attack can cause the recall on victim class to drop by 10% on the Fashion-MNIST dataset [45], indicating that even a small number of malicious clients can effectively degrade the performance of a defenseless FL system through label-flipping attack. In PoisonGAN [46], the label-flipping attack is further improved. Targeting a FL system for image classification, the authors of PoisonGAN use the global model received on clients as the discriminator for Generative Adversarial Network (GAN). The attacker trains a local generator until the global model classifies generated images as the victim class. The attackers can then flip labels of generated images, compromising client models by feeding fake images along with flipped labels. The noteworthy advantage of PoisonGAN is that the attacker now does not need to access clients' data. The attacker can simply generate their own poi-

Table 2: Comparison of Related Surveys on Federated Learning Attacks and Defenses

Surveys	Federated Learning Attacks and Defenses							
	D2M		M2M		M2D		Composite	
	Threat	Defense	Threat	Defense	Threat	Defense	Threat	Defense
Kairouz et al. [23]	○	✓	○	✓			○	✓
Nguyen et al. [37]	✓	✓	✓	✓			✓	✓
Zhang et al. [38]	○	✓	○	✓	○	✓	○	✓
Gong et al. [39]	○		○				○	✓
Yin et al. [40]					✓	✓		
Zhang et al. [41]	○	✓	○	✓	○	✓		
ours	✓	✓	✓	✓	✓	✓	✓	✓

○: high-level overview ✓: detailed review.

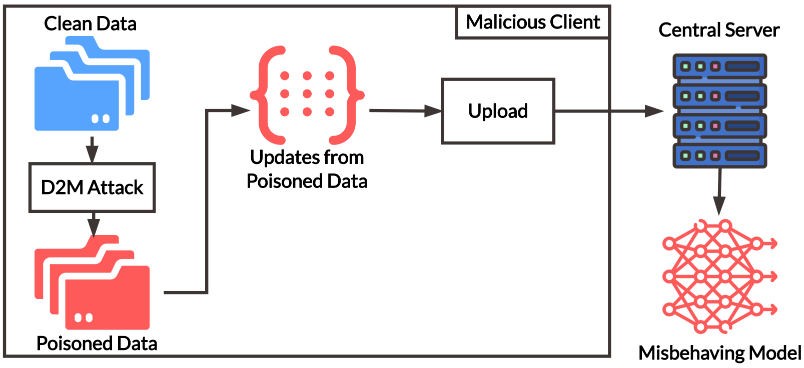


Figure 2: An illustration for D2M attack.

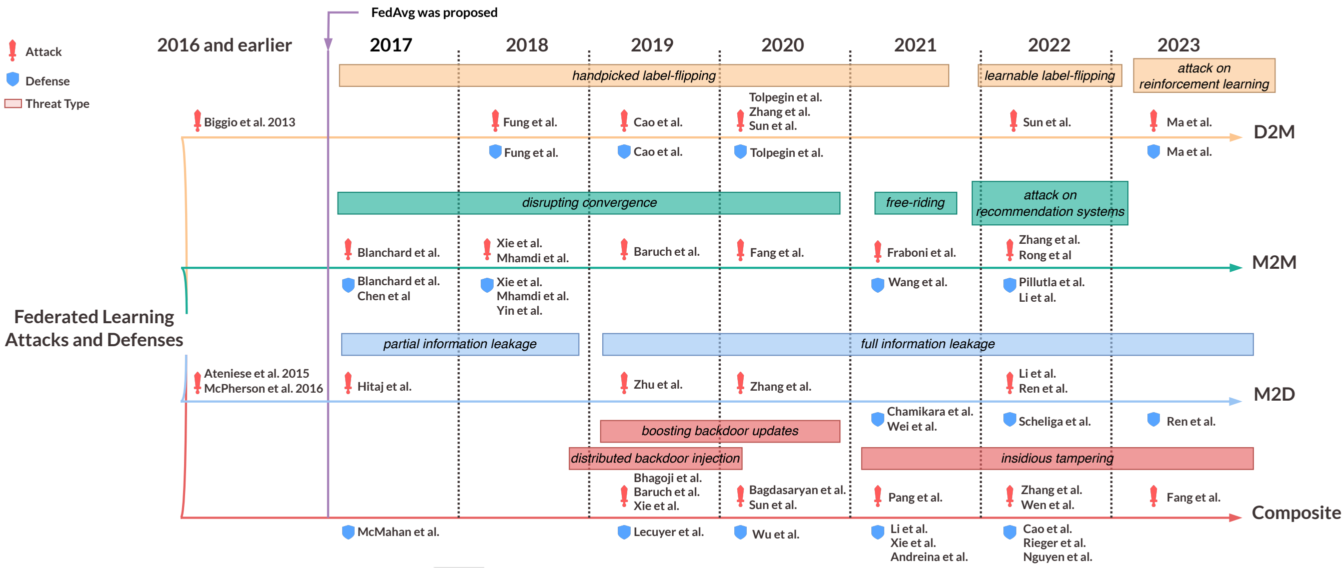


Figure 3: The timeline of research on FL attacks and defenses.

sonous data samples. Instead of arbitrarily choosing the target class to flip, studies such as [47, 48] investigate different heuristic for choosing the target class. Semi-targeted attack proposed in [48] uses distance measures to determine which target class can more easily affect model predictions. The intuition of this attack is that if samples of two different classes are relatively close in the feature space, then label-flipping attack on these two classes is more likely to succeed as the proximity of features suggests easier learning convergence. The authors of [48] consider both the Independent and Identically Distributed (IID) and non-Independent and Identically Distributed (non-IID) scenarios. If client data is IID, the attacker uses the global model to extract features for the local training data. The geometric center of each class is computed based on features of local data and the target class should be the one closest to the victim class. In the non-IID scenario, the local feature space no longer well represents the structure of the global feature space. Thus, the authors leverages the scale of updates to measure which class is closer to the victim class. The attacker feeds local samples of the victim class to the global model and examines the scale of gradients when these samples are annotated as different classes. The class label that induces the smallest gradient is chosen as the target class. Different from [46, 48] that exploit the global model for their attacks, the heuristic of the edge-case attack [47] is built on the distribution of the training data. The edge-case attack flips labels into classes in the tail of the data distribution. Although the edge-case attack only affects a minority of samples, it can severely impair the model's fairness for underrepresented input and may pose great threats in autonomous driving systems [47]. Experiments in [47] show that the attack is most effective when the attacker holds most of the edge samples. As honest clients possess larger portions of edge samples, the attack is erased by benign updates.

3.2. D2M Attacks on Samples

Labels are not the only target in D2M attacks. Depending on the FL scenario, the attackers may choose to poison other relevant client data. A threat model that targets the sample size on clients is proposed in [52]. Based on the fact that FedAvg computes the weighted average of client weights based on the numbers of their corresponding local samples, the attacker can simply falsely report the number of local samples to be a large number such that the aggregated model will be dominated by the attacker's chosen model. AT^{2}FT [50] is another D2M attack that generate poisonous samples. The difference between AT^{2}FT and PoisonGAN [46] is that the former does not flip labels. Authors of AT^{2}FT formulates their attack as a bilevel optimization problem in which the attacker tries to perturb subsets of local training samples such that losses on local clean data are maximized. In essence, the AT^{2}FT algorithm maximizes local losses through gradient ascent where gradients *w.r.t* the perturbed data are approximated by minimizing a dual problem. The D2M attacks are also not limited to classification tasks. The authors of [53] propose a D2M threat model, local environment poisoning, targeting federated Reinforcement Learning (RL). The attacker can influence the learned policy by providing fake rewards during local agent training. Fake rewards are derived from gradient descent such that they minimize the objective function of RL. A D2M threat model on Federated Recommendation (FedRec) systems is proposed in [54]. Specifically, the authors of [54] focused on the graph neural network based FedRec system proposed in [55]. By feeding compromised client models with fake item ratings during training, the attacker can force the recommendation system to show specified item ratings for specific users.

Unlike the above methods that use D2M attacks to influence model predictions, the covert channel attack proposed in [51] aims at secretly transmitting messages between two clients. On the receiver client, the attacker first looks for edge samples from its local training data such that even a small perturbation in the data results in different classification outcomes. Perturbed edge samples along with the transmission interval, the clean and poisoned class predictions are sent to the sender client. The sender client decides whether to fine-tune its local model with the perturbed data depending on the message bit it wishes to send and the local model's prediction. Once the receiver client receives the updated model, it can decode the message bit based on the classification outcome of perturbed samples.

For D2M attacks to be successful, studies in [43, 44, 49] show that it is vital to ensure the availability of malicious clients during training. If no malicious client are selected to participate in the global model update, the effects of their attacks can be quickly erased by updates from benign clients [44]. Recent studies on FL threat models tend to combine D2M attacks with M2M attacks to launch more powerful composite attacks. Since the attacker also manipulates model updates, composite attacks can be stealthier and more persistent. Such attacks also give the attacker more freedom of when and how to trigger the attack.

3.3. Defense Against D2M Attacks

In this section we introduce defense strategies proposed along with studies on label-flipping attacks [43, 44, 49, 53]. Since D2M attacks ultimately induce changes in model updates, FL system administrators may also consider defense mechanisms designed for M2M or composite attacks.

Strategies proposed in [43] and [44] are both inspired by the observation that gradients in FL behave differently in terms of benign and malicious clients. In particular, because of the non-IID nature of data, it is observed in [43] that gradients from benign clients are more diverse than those from malicious clients. This is because benign gradients conform to the non-IID distribution of local data while malicious models have a shared poisoning goal. The defense strategy FoolsGold [43] thus aims at reducing the learning rate of similar model updates while maintaining the learning rate of diverse updates. To determine the similarity of model updates, the history of all model updates are stored and pair-wise cosine similarity between current and historical updates are computed. The defense strategy in [44] requires prior knowledge on the attack target. This method needs the user to first choose a suspect class that is believed to be poisoned. Then only model updates directly contributing to the prediction of the suspect class are collected. These model weights subsequently go through Principal Component Analysis (PCA) and are clustered based on their principal components. Principal components of benign and malicious clients fall in different clusters. Similar to gradients, model weights can also be used to differentiate benign and malicious clients. Sniper [49] is a defense strategy based on the Euclidean distances between model weights. The central server first computes the pair-wise distances between received client models. Then the server constructs a graph based on the distances. Client models are the nodes of the graph, and if the distance between two client models are smaller than the given threshold, these two models are then linked by an edge. If the

Table 3: Characteristics of D2M Attacks.

Threat Model	Threat Objective	Poisoned Data
Label-Flipping [43, 44, 49]	mislassification	class labels
Semi-Target Poisoning [48]	misclassification	class labels
Edge-case Attack [47]	misclassification	class labels
AT^{2}FT [50]	misclassification	general samples
PoisonGAN [46]	misclassification	general samples and class labels
Covert Channel [51]	secretly passing messages	edge samples
Fake Sample Size [52]	disrupting convergence	client dataset size
Local Environment Poisoning [53]	poisoning policy	agent rewards
Poisonous Ratings [54]	controlling item recommendation	item ratings

number of models in the maximum clique of the graph is larger than half of the total number of clients, models in this clique are aggregated to update the global model. Otherwise, the server increases the distance threshold and repeat the above process until a suitable clique can be found.

Parallel learning [56] is a paradigm of RL in which multiple agents learn concurrently to solve a problem. Parallel learning not only alleviates data deficiency but also stabilizes training, as agents learn from diverse experiences. Unlike multi-agent RL, which aims to develop competitive or cooperative strategies among clients, parallel RL focuses on solving single-agent problems through parallel training. This objective is similar to that of conventional federated learning, in which the goal is to obtain a global model through distributed local model training. Therefore, federated reinforcement learning becomes imperative when the learning environment of RL is privacy-sensitive. For the D2M threat model targeting federated RL, a corresponding defense strategy was also proposed in [53]. This method requires the central server to evaluate client agent performance to determine their credibility. Specifically, the central server tests client policies and computes their corresponding rewards. The central server aggregates client policies based on a set of weights derived from normalized rewards.

3.4. Evaluation Metrics for Attacks and Defenses on Classification Tasks

Since the majority of studies on D2M attacks focus on image classification, the most commonly used datasets for D2M attack evaluation are MNIST [57], Fashion-MNIST [45] and CIFAR-10 [58]. Natural language and domain-specific datasets can also be seen [43, 47, 50, 54]. Attack Success Rate (ASR) is widely used to evaluate the effectiveness of an attack. Specifically, for D2M attacks targeting classification tasks, ASR is defined as the proportion of targeted test samples being misclassified, namely,

$$ASR = \frac{\sum_{(x_i,y_i)\in D} \mathbb{1}\{f(x_i) = y_t, y_t \neq y_i\}}{|D|} \tag{1}$$

where D is the test set for evaluation, x_i is the data sample while y_i is its corresponding groundtruth label, y_t is the label chosen by the attacker, $f(\cdot)$ is the attacked global model, and $\mathbb{1}\{\cdot\}$ equals to 1 if the condition inside the brackets is met. ASR is also used to evaluate M2M or composite attacks. The metric respectively reflects how severely the attack disrupts model convergence and how sensitive the model is to backdoor triggers. In addition, the performance of the attack can also be demonstrated by the decrease in overall classification accuracy. For regression tasks, mean absolute error and root mean squared error are employed. While some defenses provide formal proof for their effectiveness, most work on FL defenses is empirically validated by demonstrating the robustness of model performance when the defense is adopted in a malicious environment.

4. Model to Model Attacks

We define Model to Model (M2M) attacks in FL as threat models that manipulate local model updates or weights to affect the global model, as depicted in Figure 4. The primary objective of an M2M attack is to disrupt the convergence of FL algorithms. The presence of M2M attacks is also described as the Byzantine problem [59]. In a distributed system affected by the Byzantine problem, benign and malicious participants coexist in the system. Malicious participants deliberately disseminate confusing or contradicting information to undermine the system's normal operations. Therefore the challenge for the system administrator lies in achieving consensus among benign participants despite the presence of malicious ones. Defending against these M2M attacks means ensuring that the learning algorithm to converge to an optimal minima regardless of poisoned updates from malicious clients. In addition to the above threat model, a special case of M2M attacks, called the free-rider attack, aims to steal the global model itself, infringing on the intellectual property rights of the model owner. An malicious party may pretend to join the FL system solely to obtain

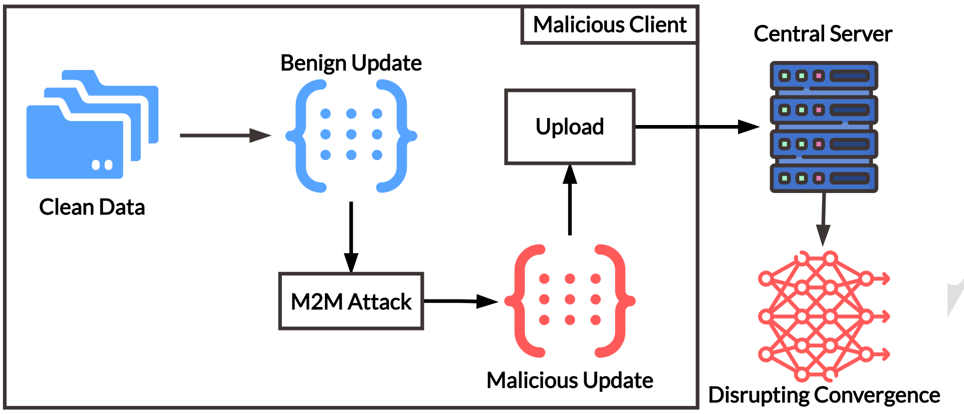


Figure 4: An illustration of M2M attack.

the distributed global model, without contributing to the learning task. Since the threat model of free-rider attack is comparatively straightforward, we discuss this type of attack along with its defense mechanisms in the same section. The characteristics of discussed M2M attacks are shown in Table 4.

4.1. General M2M Threat Models

Existing M2M threat models can be divided into *a priori* and *a posteriori* attacks.*A priori* attacks do not require any knowledge of benign clients while *a posteriori* attacks need to forge poisonous model updates based on information from benign clients.

4.1.1. Priori M2M Attacks

A straightforward *a priori* M2M (prioM2M) attack is sending noise to the central server. This method is dubbed as Gaussian Byzantine in [61]. The Gaussian distribution for noise sampling often has zero mean but large variance to disrupt the convergence of the learning algorithm. Gaussian Byzantine is often used as the baseline attack [62, 63]. Bit-flipping is a prioM2M attack proposed in [62]. On malicious clients, the bit-flipping attack flips four significant bits of certain 32-bit floating numbers in the original gradients as poisoned model updates. Another two prioM2M attacks, same-value attack and sign-flipping attack, are proposed in [63]. For the same-value attack, malicious clients upload vectors with an identical random value on each dimension to the server. In the sign-flipping attack, malicious clients computes their own gradient as normal but flip the sign of gradients before uploading them to the central server. The prioriM2M attack proposed in [64] takes secure aggregation rules into account. It specifically attacks FL systems equipped with median-based aggregation rules such as TrimMedian [70] or Krum [61]. The basic idea of the attack is to report false updates on multiple malicious clients such that with high probability the aggregation rule picks one of the malicious updates as the median for global update. The authors of [64] use a statistical heuristic to find the maximum deviation range which is used to forge the malicious updates. The value on each dimension of the original updates on malicious clients is transformed by the maximum deviation range to attain forged malicious updates. The authors also augment this attack with the D2M attack, which is discussed in Section 3.4.

4.1.2. Posteriori M2M Attacks

For *a posteriori* M2M (postM2M) attacks, omniscient negative gradient approach proposed in [62] is an equally straightforward approach compared to Gaussian Byzantine. This method assumes that the attacker have full knowledge of benign clients, then malicious clients only need to send scaled negative sum of benign gradients to the central server. The scaling factor is a large number on the order of magnitude of 10^{20}. The postM2M attack proposed in [65] takes Bayzantine-resilient aggregation rules into account. Specifically, this attack targets aggregation rules that compute the norms of client gradients to filter out malicious updates. The problem with norm-based aggregation rules is that L^p norms cannot tell if two norms only differ in one specific dimension or every dimension. Thus, the attacker can exploit this by only poisoning one dimension of the gradients. The poisoned value can be scaled by a large factor while still being accepted by the aggregation rule as its norm is not far away from those of the benign gradients. Moreover, as the norm chosen by the aggregation rule approaches the infinite norm, the attacker can poison every dimension of model updates.

The above attacks can be launched individually on clients controlled by the attacker, these approaches does not require malicious clients to coordinate with each other. A colluding postM2M attack is later proposed in [66]. This method targets aggregation rules such as Krum [61] and Buylan [65] that use the Euclidean distance between client models as the criterion for choosing trustworthy models. The threat model in [66] aims at pushing the global model towards the opposite of the benign update direction. To achieve this at the presence of aforementioned aggregation rules, a chosen malicious client is responsible for generating model updates that maximizes the global model update in the opposite direction. Other malicious clients generate updates that are close to the chosen one, conceiving the aggregation rules that malicious clients form a benign cluster and the chosen malicious client should be picked

Table 4: Various M2M threat models

<table>
<tr><th>Threat Model</th><th>Approach</th><th>Type</th><th>Objective</th></tr>
<tr><td>free-riding [60]</td><td>pretend as a client</td><td rowspan="6">a priori</td><td>stealing global model</td></tr>
<tr><td>Byzantine Gaussian[61]</td><td>uploading Gaussian noise</td><td rowspan="7">inhibiting convergence</td></tr>
<tr><td>bit-flipping [62]</td><td>flipping significant bits of floating numbers</td></tr>
<tr><td>same-value attack [63]</td><td>uploading vectors with identical values across all dimensions</td></tr>
<tr><td>sign-flipping [63]</td><td>flipping signs of gradients on attacked clients</td></tr>
<tr><td>median cheating [64]</td><td>cheating the aggregation rule to pick the false median</td></tr>
<tr><td>negative gradient [62]</td><td>uploading the scaled sum of benign gradients from malicious clients</td><td rowspan="6">a posteriori</td></tr>
<tr><td>norm attack [65]</td><td>scaling certain dimensions of gradients</td></tr>
<tr><td>colluding attack [66]</td><td>deceiving the aggregation rule to pick the chosen malicious client</td><td>converging to an inferior minima</td></tr>
<tr><td>PipAttack [67]</td><td>generating item embeddings based on public information</td><td rowspan="3">increasing $ER@K$ of target items</td></tr>
<tr><td>FedRecAttack [68]</td><td>minimizing the rating scores of untargeted items</td></tr>
<tr><td>User Approximation [69]</td><td>generating item embeddings through approximated user embeddings</td></tr>
</table>

by the aggregation rule.

4.2. M2M Threat Models on Federated Recommendation Systems

As mentioned in the introduction section, FL is well-suited for recommendation systems thanks to its ability to provide personalized recommendations and reduce privacy risks. A commonly used FedRec framework is proposed in [71]. Research on the vulnerabilities of domain-specific FL like FedRec is still a nascent area. In this section, we introduce three noteworthy studies [68, 67, 69] focusing on exploiting security vulnerabilities of FedRec.

The common goal of existing attacks on FedRec is to increase the exposure rate of certain items. The affected recommendation system may always present or never show certain items to users. In [68, 67, 69], the attackers are assumed to only have access to item embeddings, local and global models. Embeddings that characterize users are always hidden from the attackers. In PipAttack [67], the attacker increases target items' exposure rate by forging their embeddings to be similar to those of popular items. Since the attacker have no access to the popularity of items in the system, this information is retrieved from the Internet. Based on the retrieved information, the attacker locally train a popularity classifier with item embeddings as input. The weights of the classifier are then fixed, target item embeddings are poisoned by enforcing them to be classified as popular by the classifier. The poisoned item embeddings are uploaded to the central server to mislead the FedRec system.

Authors of FedRecAttack [68] later points out that major limitations of PipAttack include that it may severely degrade the recommendation performance and it needs around 10% of clients to be attacked for it to be effective. Since the exposure rate at rank K ($ER@K$) [67], meaning the fraction of users whose top-K recommended items include the target item, is a non-differentiable function, FedRecAttack uses a surrogate loss function to facilitate the attack. FedRecAttack also assumes that around 5% of user-item interaction histories are publicly available for the attacker to use. The loss function of FedRecAttack encourages the rating scores of recommended non-target items to be smaller than the scores of target items with no interaction history, then the gradients of target item embeddings *w.r.t* this loss function are uploaded to the central server. To further eschew being detected by secure aggregation rules, these gradients are normalized before uploading if their norms are larger than the threshold.

Both PipAttack and FedRedAttack require public prior knowledge to work. In contrast, the $A-ra/A-hum$ attack proposed in [69] does not have this requirement. $A-ra/A-hum$ also uses a surrogate loss function to promote the $ER@K$ for target items, but this attack focuses on approximating the user embeddings which are inaccessible in FedRec. $A-ra$ assumes that the user embeddings are distributed by a zero mean Gaussian with the variance as a hyper-parameter. The attacker first samples a number of user embeddings from the Gaussian distribution, then maximized the interaction scores target items and sampled user embeddings to derive poisonous item embeddings. Instead of sampling from a Gaussian, $A-hum$ uses online hard user mining to generate user embeddings. The attacker first generate hard user embeddings that are not likely to interact with existing items. Then target item embeddings are optimized to increase their interaction chances with the synthesized hard users.

Table 5: Characteristics of M2M Defenses

Type of Defense	**Aggregation Criterion**
GeoMed[72]	geometric median
RFA [73]	Weiszfeld-smoothed geometric median
MarMed[62]	dimension-wise median
MeaMed[62]	mean-around median
TrimMean[70]	dimension-wise trimmed mean
Krum/Multi-Krum [61]	Euclidean distance
Bulyan[65]	Euclidean distance and mean-around median
ELITE[74]	gradient information gain

4.3. Defense Against M2M Attack

Because the median is robust to outliers in statistics, it is widely used in M2M defenses to filter out malicious updates. GeoMed [72] is an exemplar of median-based M2M defenses. In GeoMed, the central server first divides received client gradients into multiple groups and computes the mean of each group. Then the geometric median of group means is used as the gradient for updating the global model. The approach of using geometric median for robust aggregation is further improved by authors of RFA [73]. In RFA, clients compute their aggregation weights based on the aggregation rule inspired by the Weiszfeld algorithm [75]. Including the geometric median, more median-based defenses are studied in [62]. Marginal Median (MarMed) is a generalized form of median proposed in [62]. It computes the median on each dimension for client gradients. Mean-around-Median (MeaMed) in [62] further leverages more values around the median. Built upon MarMed, MeaMed finds the top-k values that are nearest to the median of each dimension, then the mean of these nearest values is used as the gradient on their corresponding dimensions.

Besides median, trimmed mean also has the benefit of being less sensitive to outliers. The authors of [70] introduce coordinate-wise trimmed mean as an aggregation rule. For each dimension of client gradients, this rule removes the top-k largest and smallest values, the mean of the remaining values is treated as the gradient on the corresponding dimension.

Another criterion for filtering out malicious updates is the Euclidean distance between norms. Krum [61] and Bulyan [65] are two exemplary defenses built on this criterion. Krum is motivated by avoiding the drawbacks of square-distance or majority based aggregation rules. The problem pointed out in [61] is that malicious attackers can collude and misguide the center of norms to a bad minima for the sqaure-distance based aggregation, and the majority based aggregation is too computationally expensive as it needs to find a subset of gradients with the smallest distances among them. For a central server that adopts Krum as its aggregation rule, it first finds the $(n-f-2)$ nearest neighbors for each client based on the Euclidean distances between their updates, where n is the number of clients that participate the training, f is the estimated number of malicious clients. Then the central server sums up the distances between each client and their corresponding neighbors as Krum scores. The client with lowest score is chosen by the central server, and its gradient is used to update the global model for the current training round. Multi-Krum [61] is a variation of Krum that balances averaging and Krum. It chooses top-k clients with highest Krum scores. The average of chosen clients' updates is used to update the global model. The prerequisite for Krum to be effective is that the number of malicious clients needs to satisfy $f > (n-2)/2$.

Although the convergence of Krum has been proven in [61], authors of Bulyan [65] point out that the attacker can simply deceive Krum to pick the malicious client that converges to an ineffective local minima. Such an attack is launched by manipulating the gradient norms as discussed above. Bulyan refines norm-based aggregation rules such as Krum by adding an extra stage after a client has been chosen by the central server. The added stage is akin to MeaMed [62]. Bulyan first iteratively move clients chosen by Krum or other rules to a candidate set. Once the number of candidates passes the threshold $2f+3$, Bulyan computes the MeaMed on each dimension of candidate gradients. The resulting vector is regarded as the output of Bulyan and subsequently used to update to global model. For Bulyan to be effective, the number of malicious clients needs to satisfy $f > (n-3)/4$.

Different from the above approaches, ELITE [74] uses information gain to filter out malicious updates. ELITE first computes the empirical probability density function for each dimension of gradients, which allows for deriving the dimension-wise information entropy. The sum of all entropy is computed as the total entropy of updates for the current training round. Then for each participating client, their information gain is defined as the difference between the original total entropy and the total entropy with this client being removed. Clients with largest information gains are considered as malicious and hence excluded from the aggregation. The intuition behind ELITE is that benign gradients tend to roughly point at the same direction, namely the direction of the optimal gradient, whereas malicious gradients tend to point at rather different directions. When the majority of clients are benign, removing malicious gradients results in less total entropy as the uncertainty of gradients is reduced.

4.3.1. Defense Against Free-Rider Attacks

Since the objective of free-rider attacks is to obtain the global model in the FL system, free-rider clients need to upload their own local model such that they can pretend to be benign clients. Free-rider models are constructed with minimum cost. The free-rider can simply upload their received global model to the server [60], or Gaussian noise may be added to the received model before uploading [76]. The key of defending against free-rider attacks is to identify which clients submit free-rider models. Existing defenses can be categorized into watermarking methods and anomaly detection methods. Watermarking methods incorporate watermark learning tasks on clients, while anomaly detection approaches are learned on the server. If a client model fails to trigger watermarked behaviors or being classified as an anomaly, such client is considered as a free-rider.

Watermarking neural networks has been studied in the centralized setting [77, 78] to verify the ownership of deep neural networks. Watermarks are commonly embedded into intermediate features or backdoored test samples. In the FL scenario, WAFFLE [79] is an early work of FL watermarking in which the server embeds watermarks by retraining the aggregated model with backdoored samples. However, watermarking on the server side is not suitable for defending against free-rider attacks, as the free-rider model is identical to the global model. FedIPR [80] addresses the problem by generating secret watermarks on clients. At the initialization stage of FL, FedIPR requires each client to generate their own trigger dataset, watermark embedding matrix and the location of watermarks. In addition to the primary learning task, local models now learns to embed watermarks in both the intermediate features and local trigger set. In the verification stage, client models are fed with their respective trigger set. If the detection error of trigger samples is smaller than a given threshold, this client passes the verification. FedIPR also verifies feature-based watermarks by evaluating the Hamming distance between the watermark in the global model and local secret watermark. One major challenge of FedIPR is that clients may generate conflicting watermarks. Authors of FedIPR proves that different client watermarks can be embedded without conflicts when the total bit-length of watermarks is bounded by the channel number of the global model. If the bit-length exceeds the threshold, FedIPR also gives a lower bound for detecting watermarks.

Anomaly detection based free-rider defense are inspired by anomaly detection approaches in the centralized setting, such as [81, 82]. Authors of [76] concatenate client updates on the server to train an auto-encoder. The auto-encoder learns to reconstruct received client updates. In the verification stage, if the reconstruction error induced by updates from one client is larger than then given threshold, this client is deemed as a free-rider. Another approach proposed in [76] is using DAGMM [82] instead of the vanilla auto-encoder. DAGMM detects anomaly data by feeding the latent representation of the auto-encoder to a Gaussian mixture network to estimate the likelihood of the representation being abnormal.

5. Model to Data Attacks

In this section, we will introduce the Model to Data (M2D) attacks in FL, which is to reveal a specific attribute, partial or full of the data. We summarized the methods to be non-gradient-based leakage and gradient-based data leakage.

5.1. Non-Gradient-Based Data Leakage

We define non-gradient-based data leakage as the disclosure of private information that occurs independently of the gradient generated during the training stage. For instance, the leakage can involve identifying specific attributes or membership details within the training data, or recovering original training images from obscured or masked versions. Typically, such leakage exploits the capabilities of a well-trained model to execute these attacks.

5.1.1. Attribute Inference

The paper [83] is one of the earliest works that targets the leakage of private information from an Machine Learning (ML) model. In this paper, the authors construct a novel meta-classifier that is used to attack other ML classifiers with the aim of revealing sensitive information from the training data. This is considered a white-box attack, as the adversary has knowledge of both the structure and the parameters of the target model. Specifically, the method assumes full access to a well-trained target model and pre-sets a particular attribute to be identified, determining whether or not it exists in the training data. To do this, the authors first create multiple synthetic training datasets, some of which partially contain the pre-set attributes, while the rest do not. They then train several classification models on these synthetic datasets; the architecture of these classification models is identical to that of the target model. Subsequently, the parameters of these classification models are used as input for training the meta-classifier. Finally, the parameters from the well-trained target model are fed into this meta-classifier to determine if the particular attribute exists in the training data. Both the target model and the meta-classifier are ML models, *e.g.*, Artificial Neural Network (ANN), Hidden Markov Model (HMM) [84], Support Vector Machine (SVM) [85], or Decision Tree (DT). The authors provide two example cases to evaluate their method. In one example, they identify the speaker's nationality using a speech recognition dataset processed by an HMM. Later, they use an SVM to set up a network traffic classifier to distinguish between two kinds of traffic conditions, using the meta-classifier to identify the type of traffic. In both examples, the meta-classifiers are DTs.

5.1.2. Membership Identification

The above work is further improved by [86], who focus on membership identification attacks. They propose a shadow training technique to identify whether specific samples are part of the training dataset. The membership inference problem is formulated as a classification task. An attack model is trained to distinguish between the behavior of shadow models when fed with forged training data. These shadow models are designed to behave similarly to the target model. The approach

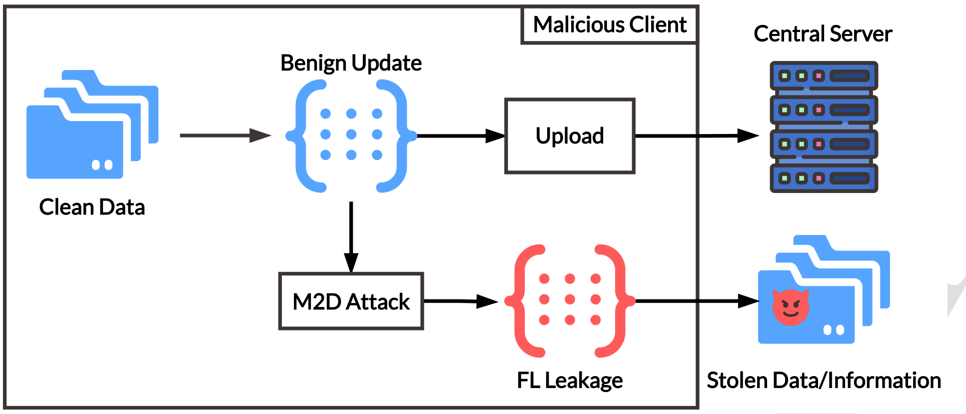


Figure 5: M2D Attack.

qualifies as a black-box attack, meaning that the attacker only possesses knowledge of the output for a given input. Several effective methods have been developed for generating forged training data for the shadow models. The first method utilizes black-box access to the target model to synthesize the data. The second method leverages statistical information related to the target model's training dataset. In the third method, it is assumed that the adversary has access to a noisy version of the target model's training dataset. While the first method operates without assuming any prior knowledge about the distribution of the target model's training data, the second and third methods allow the attacker to query the target model just once before determining whether a particular record was part of its training dataset.

5.1.3. Image Recovery

In terms of recovering valuable information from obfuscated images, [87] is one of the earliest works to the best of our knowledge. Obfuscated images are easily accessible through various data protection techniques (*e.g.*, blur, mask, corrupt, and P3) [88, 89]. In the study [87], the authors utilized a DL model to recover valuable information from obfuscated images for classification tasks. They assumed that the adversary has access to a portion of the original training data and applied one of the encryption methods to those images to train the attack model. For this reason, their method is generally not suitable for most real-world scenarios.

To demonstrate how neural networks can overcome privacy protection measures, they employed four commonly used datasets for recognizing faces, objects, and handwritten digits. Each of these tasks carries substantial privacy concerns. For instance, the successful identification of a face could infringe upon the privacy of an individual featured in a captured video. Recognizing digits could enable the deduction of written text content or vehicular registration numbers.

The final results are impressive. On the MNIST [57] dataset, they achieved an accuracy of about 80% for images encrypted by P3 with a recommended threshold level of 20. Conversely, the accuracy exceeds 80% when the images are masked by windows of resolution 8×8. On the CIFAR-10 [58] dataset, only vehicle and animal images were used for experiments, achieving an accuracy of 75% against P3 with a threshold of 20. When deploying a 4×4 mask on the images, the accuracy is approximately 70%, and it drops to 50% when masking with 8×8 resolution. On the AT&T [90] dataset, the proposed method achieved a remarkable accuracy of 97% against P3 with a threshold of 20, over 95% against various mask sizes, and 57% against face blurring. On the FaceScrub [91] dataset, they achieved an accuracy of 57% against masking the face with a 16×16 window and 40% against P3 with a threshold of 20.

In more recent work [92], the authors utilize a GAN, trained on a public dataset, to recover missing sensitive regions in images; this is termed the Generative Model-Inversion (GMI) attack, as shown in Figure 6. A diversity loss is proposed to encourage diversity in the images synthesized by the generator when projected into the target network's feature space. This is essential during the training of the GAN on the public dataset because the adversary aims for the generated images to be distinct in the feature space of the target model. If different images map to the same feature space, the adversary cannot discern which generated image corresponds to the private data's features, thus failing to reveal the private information.

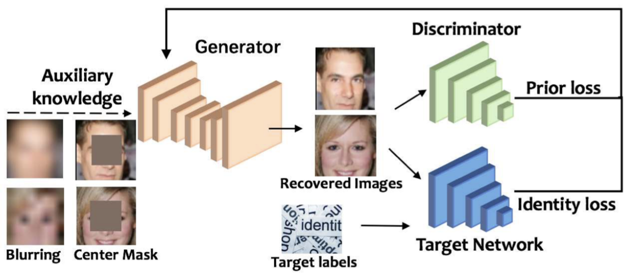


Figure 6: Overview of the GMI attack method. [92]

The authors assume that the adversary has access to the well-trained target model, which serves as a discriminator, as well as to the target label of the input corrupted image. Initially, the generator is used to create an image, which is then fed into two separate discriminators to calculate the prior loss and identity loss. In subsequent rounds, these two losses, along with the corrupted image, are used as inputs for the generator to produce the next iteration of the reconstructed image. Upon completing the training of the GAN, the adversary, during the reveal phase, only needs to continue optimizing the generator's inputs so that the generated images are sufficiently realistic while also maximizing likelihood in the target model.

The datasets employed for evaluation are MNIST [57], ChestX-ray8 [93], and CelebA [94]. The experimental results indicate that without using the corrupted image as an input for the generator, the attack's success rate is approximately 28%, 44%, and 46% on target networks VGG-16 [95], ResNet-152 [96], and face.evoLVe [97], respectively. However, when the corrupted image is incorporated, the accuracy increases to 43%, 50%, and 51% for blurred input images; 78%, 80%, and 82% for center-masked images; and 58%, 63%, and 64% for face T-masked images. Consequently, the inclusion of corrupted images as auxiliary information has a significant impact on the attack's accuracy.

5.2. Gradient-Based Data Leakage

Concerning gradient-based data leakage, this refers to techniques that exploit gradients from the target model to expose privacy-sensitive information. DL models are trained on datasets, and parameter updates occur through alignment with the feature space. This establishes an inherent relationship between the weights or gradients and the dataset. Consequently, numerous studies aim to reveal private information by leveraging these gradients. The effectiveness and success rates of gradient-based approaches have consistently surpassed those of non-gradient-based methods. Unlike non-gradient-based leakage, gradient-based data leakage can occur even in models that have not yet converged.

5.2.1. Partial Recovery

Hitaj *et al.* [98] proposed a data recovery method that utilizes a trained victim model and a target label. The method aims to generate new data closely resembling the distribution of the training dataset. This attack is formulated as a generative process using a GAN. In a FL system, an attacker can pose as a participant to reveal private data from the victim by modeling the feature space. Suppose the attacker masquerades as a malicious participant with a portion of training samples that have correct labels, along with a portion of samples generated via GAN with incorrect labels. The attacker's goal is to produce a dataset that shares the same feature distribution as the other participants, leveraging GAN and the global gradients downloaded from the parameter server.

In Algorithm 2, the victim trains its local model on its own dataset for several iterations until it achieves an accuracy beyond a preset threshold. Subsequently, the malicious actor uses the updated local model as the discriminator. The weights in the discriminator are fixed, and a generator is trained to maximize the confidence of a specific class. This is an indirect data recovery method, sensitive to the variance in the victim's training data [99]. Although the generated images are consistent with the data distribution, they do not correspond to the actual training dataset. In other words, the generated images cannot be mapped back to the training data.

Another related work by Generative Gradient Leakage (GGL) [100] also employs a GAN to generate fake data. In this approach, the weights of the GAN are pretrained and fixed, while the trainable parameters in GGL are the input sequences to the GAN. The label inference part is adapted from Improved DLG (iDLG) [101], requiring a batch size of 1. Unlike other methods, GGL uses Covariance Matrix Adaptation Evolution Strategy (CMA-ES) and Bayesian Optimization (BO) as optimizers to reduce the variability in the generated data. Although the data generated by GGL is not identical to true data, it is sufficiently similar (see Table 6), providing GGL with robustness against various defense strategies like gradient noising, clipping, or compression. The generated images are influenced by two factors: 1) the inferred ground-truth label, which specifies the image classification, and 2) fine-tuning based on gradient information to make the image as similar as possible to the true image.

Algorithm 2 The proposed work from [98]

Assume: two participants V and M who have common learning goals.
Require: V's local dataset D_v with label L_a and L_b.
M's local dataset D_m with label L_b and L_c.

a. Parameter Server

1: build model and initialize weights.
2: send the initial weights to the clients.
3: local training on victim and malicious clients.
4: receive the trained local weights and generate the global model.
5: repeat Step 2 and 3 until the model converges.

b. Victim Client

1: download the global weights from parameter server.
2: train the local model on its local dataset D_v.
3: upload the local model to the parameter server.

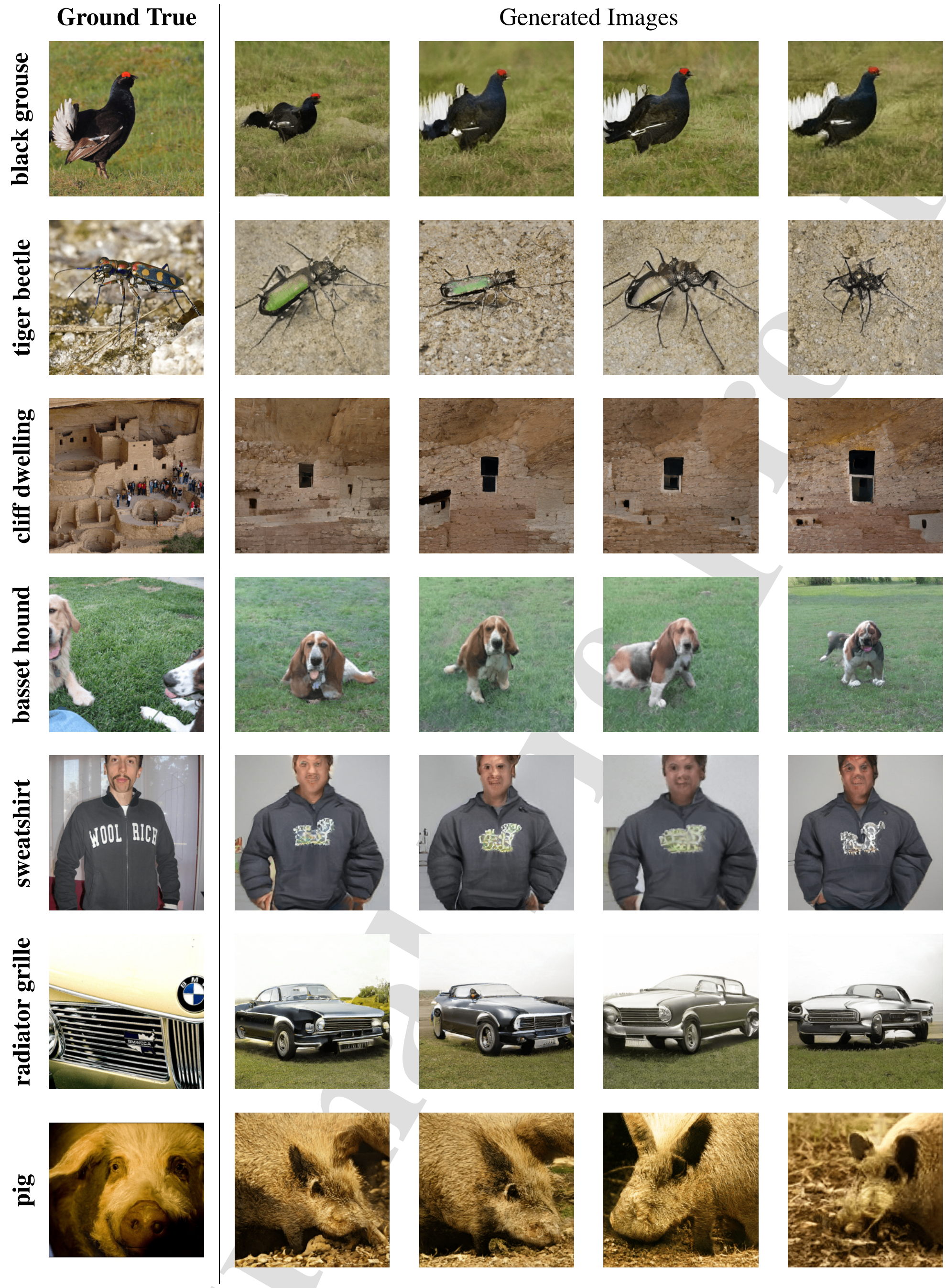
c. Malicious Client

1: download the global weights from parameter server.
2: train a GAN model to generate fake data of class L_a.
3: generate many fake data using GAN and relabel them with L_c to update the local dataset D_m.
4: train the local model on the updated local dataset D_m.
5: upload the local model to the parameter server.

5.2.2. Full Recovery (Discriminative)

Zhu *et al.* [103] introduced Deep Leakage from Gradients (DLG), framing the image recovery task as a regression problem. Initially, the shared local gradient is derived from a victim participant, and a batch of "dummy" images and labels is randomly initialized. These are then used to calculate the "dummy" gradient through standard forward-backward propagation, employing the L-BFGS optimizer [104]. This process leverages regression techniques to decipher intricate pat-

Table 6: Typical experimental results performed on GGL are shown below. The backbone network is *ResNet-18* and the dataset is ILSVRC2012 with a resolution of 256 ∗ 256. [102]



terns within the gradient, thereby reconstructing the private image data. The approach provides a powerful framework for M2D attacks. Importantly, it is the input "dummy" data that is updated—not the model parameters—by minimizing the Mean Square Error (MSE) between the "dummy" gradient and the shared local gradient. This strategy prioritizes the fidelity of the reconstructed image, ensuring preservation of essential features and details. Among existing leakage methods, DLG is unique in achieving precise pixel-wise data revelation without requiring additional information. The technique is innovative and deploys unique algorithms to achieve an unparalleled level of precision. Some results from DLG of batch data are provided in Figure 7. It marks a significant advancement in the field of gradient leakage, opening new avenues for research and application. Although DLG can perform attacks on multiple images simultaneously, the accuracy in label inference remains suboptimal. This limitation is an active area of research, with ongoing efforts to improve label inference accuracy without compromising image recovery fidelity. In conclusion, DLG offers a novel approach to image recovery, utilizing groundbreaking algorithms to attain high precision. Its potential applications extend far beyond existing methods, positioning it at the forefront of technological advancements in the field.

Zhao *et al.* [101] introduced a novel method known as iDLG, which focuses on the identification of labels in a more accurate manner. This technique involves calculating the derivative of the cross-entropy loss with respect to one-hot labels for each class in the classification task. The crux of this approach lies in the distinct ranges of the derivative values that correspond to different labels. The authors discovered that the derivative value for the ground-truth label uniquely falls within the range of [-1, 0], while the derivatives corresponding to incorrect labels lie within the range of [0, 1]. This separation of value ranges provides a solid basis for identifying the correct label. By simply examining the derivative value, the system can distinguish the correct label from incorrect ones. However, this method has a limitation concerning the batch size: the batch size must not exceed 1 during the process. While this constraint may affect efficiency in large-scale applications, the iDLG method's unique approach to label identification through derivative analysis represents a significant contribution to the field of gradient leakage. It opens avenues for future research to potentially refine this technique and mitigate its limitations.

In addition to the low accuracy of label inference, DLG often fails to recover the image from the gradient when the data variance is large, see Figure 8. This is particularly common for datasets with a large number of classes. Inverting Gradient (IG) [105] improved the stability of DLG and iDLG by introducing a magnitude-invariant cosine similarity metric for the loss function, termed Cosine Distance (CD). This approach aims to find images that yield similar prediction changes in the classification model, rather than images that produce closely matching values with a shared gradient. The method demonstrates promising results in recovering high-resolution images (*i.e.*, 224×224) when trained with large batch sizes (*i.e.*, $\#Batch = 100$); however, the Peak Signal-to-Noise Ratio (PSNR) remains unacceptably low.

Similar to [105], Jeon *et al.* [106] argued that relying solely on gradient information is insufficient for revealing private training data. They introduced GIAS, which employs a pre-trained model for data revelation. Yin *et al.* [107] reported that in image classification tasks, the ground-truth label can be easily inferred from the gradient of the last fully-connected layer. Additionally, Batch Normalization (BN) statistics can significantly improve the efficacy of gradient leakage attacks and facilitate the revelation of high-resolution private training images.

Another approach to gradient leakage attacks is based on generative models. Wang *et al.* [108] trained a GAN with a multi-task discriminator, named mGAN-AI, to generate private information based on gradients.

5.2.3. Full Recovery (Generative)

In the work [109], the Generative Regression Neural Network (GRNN) was proposed as a method for reconstructing private training data along with its associated labels. The model is capable of handling large batch sizes and high-resolution images. Some examples are provided in Figure 9 Inspired by both GAN and DLG methods, GRNN introduces a gradient-driven approach for image creation that effectively addresses the challenges of stability and data quality commonly associated with DLG methodologies.

The novel GRNN, which serves as an innovative data leakage attack technique, is capable of retrieving private training images with resolutions up to 256×256 and batch sizes of 256. This makes it particularly well-suited for FL applications, as both the local gradient g and the global model $\mathcal{F}(\bullet)$ are easily accessible within the system's configuration. The GRNN algorithm employs a dual-branch structure to generate fake training data $\hat{x}$ and corresponding labels $\hat{y}$. It is trained to estimate a fake gradient $\hat{g}$, computed from the generated data $\hat{x}$ and labels $\hat{y}$, such that it closely matches the true gradient g associated with the global model. The divergence $\mathcal{D}$ between the true and fake gradients is evaluated using a combination of MSE, Wasserstein Distance (WD), and Total Variation Loss (TVLoss) metrics.

Through empirical testing on various image classification challenges, the GRNN approach has been rigorously compared to cutting-edge alternatives, showing significantly better results across multiple metrics. The trial findings confirm that the proposed method is notably more stable and capable of generating images of superior quality, especially when applied to large batch sizes and high resolutions.

Compared to the most latest work [103, 101, 105], GRNN takes a generative approach, which shows high stability for recovering high-resolution images (*i.e.* up to 256×256) with a large batch size (*i.e.* $\#Batch = 256$). Table 7 presents the key differences between DLG, iDLG IG and GRNN.

Algorithm 3 GRNN: Data Leakage Attack [109]

1: $g \leftarrow \partial\mathcal{L}(\mathcal{F}(<x, y>, \theta))/\partial\theta$; #Produce true gradient on local client.
2: $v \leftarrow$ Sampling from $\mathcal{N}(0, 1)$; #Initialize random vector inputs.
3: **for** each iteration $i \in [1, 2, ..., I]$ **do**
4: $(\hat{x}_i, \hat{y}_i) \leftarrow \mathcal{G}(v|\hat{\theta}_i)$; #Generate fake images and labels.
5: $\hat{g}_i \leftarrow \partial\mathcal{L}(\mathcal{F}(<\hat{x}_i, \hat{y}_i>, \theta))/\partial\theta$; #Get fake gradient on global model.
6: $\mathcal{D}_i \leftarrow \hat{\mathcal{L}}(g, \hat{g}_i, \hat{x}_i)$; #Loss between true and fake gradient.
7: $\hat{\theta}_{i+1} \leftarrow \hat{\theta}_i - \eta(\partial\mathcal{D}_i/\partial\hat{\theta}_i)$; #Update GRNN model.
8: **end for**
9: **return** $(\hat{x}_I, \hat{y}_I)$; #Return generated fake images and labels.

5.3. Defense Against M2D Attacks

The issue of M2D attack methods has garnered significant attention in the world of ML and DL. This issue has sparked concern as it can lead to the unintended exposure of information. In response, numerous methods and techniques have been proposed to understand, mitigate, and control this leakage, *e.g.*, gradient perturbation [103, 110, 111, 112, 109], data obfuscation or sanitization [113, 114, 115, 116, 117], and other

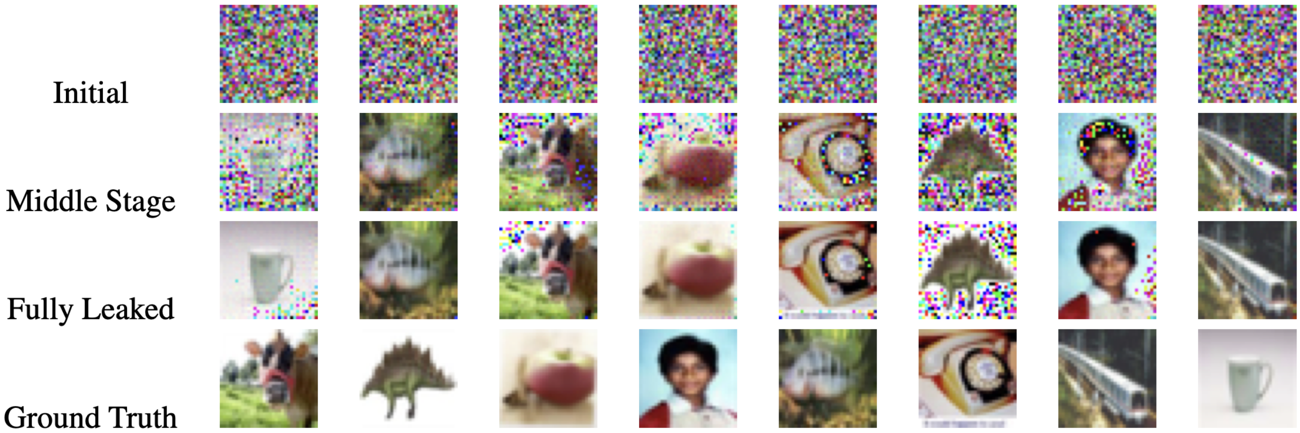


Figure 7: Although the sequence might differ and additional artifact pixels are present, deep leakage in batched data still generates images that closely resemble the original versions. [101]

Table 7: Comparison of different related works on gradient leakage. [109]

Method	**Recovery Mode**	**#Batch**	**Resolution**	**Loss Function**
DLG[103]	Discriminative	Small, up to 8	Low 64×64	MSE
iDLG[101]	Discriminative	Small, only 1	Low 64×64	MSE
IG[105]	Discriminative	Medium, up to 100	High 224×224	CD & TVLoss
GGL[100]	Generative	Small, only 1	High 224×224	CMA-ES & BO
GRNN[109]	Generative	Large, up to 256	High 256×256	MSE & WD & TVLoss

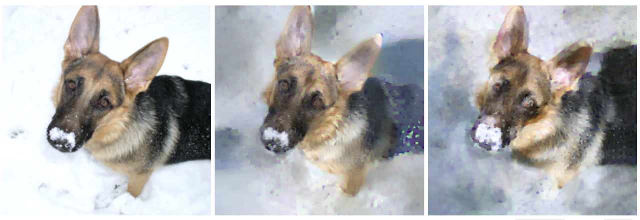


Figure 8: Reconstructed image using its gradient features. On the left is the ground true image taken from the validation dataset. The center image is reconstructed using a trained ResNet-18 model that has been trained on ILSVRC2012 dataset. On the right is the image rebuilt using a trained ResNet-152 model. [105]

methods [118, 36, 119, 120, 121, 102]. These methods aim to limit the extent of information that can be exposed, ensuring that models operate with the requisite confidentiality and integrity. Defense against M2D attacks has emerged as a compelling and dynamic research area within the field. M2D attacks involve malicious attempts to extract or manipulate sensitive information directly from the data used in training models. This field of research explores various strategies and mechanisms to shield against these attacks, preserving the privacy of the data and maintaining the robustness of the models.

Numerous measures have been undertaken to safeguard personal data against the M2D attack. Techniques such as gradient perturbation, data obfuscation or sanitization, Differential Privacy (DP), Homomorphic Encryption (HE), and Secure Multi-Party Computation (MPC) are among the most prominent methods for ensuring the privacy of both the private training data and the publicly shared gradient exchanged between the client and server. Experiments conducted by Zhu *et al.* [103] focused on two specific noise types: Gaussian and Laplacian. Their findings revealed that the key factor affecting the outcome was the magnitude of the distribution variance, rather than the type of noise itself. When the variance exceeds 10^{-2}, the leakage attack fails; concurrently, there is a significant decline in the model's performance at this variance level. Chamikara *et al.* [117] introduced a technique for perturbing data, affirming that this approach maintains model performance without compromising the confidentiality of the training data. In this context, the dataset is treated as a data matrix, and a multidimensional transformation is applied to project it into a new feature space. Various degrees of transformation are used to perturb the input data, guaranteeing an adequate level of alteration. A central server is responsible for creating global perturbation parameters in this technique. Notably, a potential drawback is that the perturbation process could distort the architectural structure of image-related data. Wei *et al.* [121] employed DP to introduce noise into the training datasets of each client and formulated

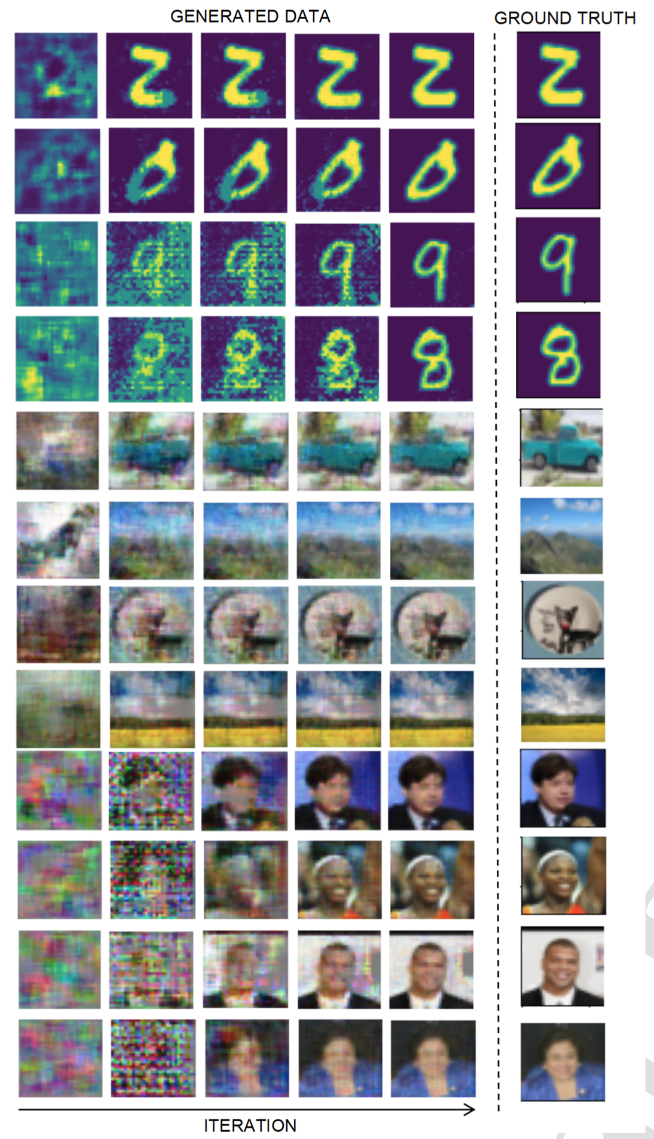


Figure 9: Examples of data leakage attack using the GRNN on the global model. [109]

a per-example-based DP method known as Fed-CDP. They developed a dynamic decay noise injection strategy to improve both inference performance and the level of gradient leakage defense. Nevertheless, experimental findings indicate that, despite successfully hindering the reconstruction of training data from the gradient, this method leads to a considerable decline in inference accuracy. Additionally, since DP is applied to every training instance, the computational overhead becomes substantial.

When computing the gradient, Privacy Enhancing Module (PRECODE) [122] aims to prevent the input information from propagating through the model. PRECODE introduces a module before the output layer to transform the latent representation of features using a probabilistic encoder-decoder. This encoder-decoder is comprised of two fully-connected layers. The first layer encodes the input features into a sequence and then normalizes this sequence based on calculated mean and standard deviation values. The mean is computed from the first half of the sequence, while the standard deviation is derived from the remaining half. Finally, the decoder translates the normalized sequence back into a latent representation, which then serves as input to the output layer. This normalization step between the encoder and decoder prevents the input information from affecting the gradient, thereby allowing PRECODE to resist the leakage of input information through the gradient. However, the insertion of two fully-connected layers in front of the output layer results in a significant computational cost. This is why only three very shallow neural networks were used for experiments in their paper.

Recent studies have uncovered that shared gradients can result in the potential exposure of sensitive data, leading to privacy violations. The work in [102] presents an exhaustive examination and offers a fresh perspective on the issue of gradient leakage. These theoretical endeavors have culminated in the development of an innovative gradient leakage defense strategy that fortifies any model architecture by implementing a private key-lock mechanism. The only gradient communicated to the parameter server for global model aggregation is the one that has been secured with this lock. The newly formulated learning approach, termed FedKL, is designed to withstand attacks that attempt to exploit gradient leakage.

The key-lock component has been meticulously designed and trained to ensure that without access to the private details of the key-lock system: a) the task of reconstructing private training data from the shared gradient becomes unattainable, and b) there is a considerable deterioration in the global model's ability to make inferences. The underlying theoretical reasons for gradients potentially leaking confidential information are explored, and a theoretical proof confirming the efficacy of our method is provided.

The method's robustness has been verified through extensive empirical testing across a variety of models on numerous widely-used benchmarks, showcasing its effectiveness in both maintaining model performance and protecting against gradient leakage.

In the study [102], a theoretical foundation is laid to demonstrate that the feature maps extracted from the fully-connected layer, convolutional layer, and BN layer contain confidential details of the input data. These details are not only encompassed within the feature maps but also coexist within the gradient during the process of backward propagation. Furthermore, it is posited that gradient leakage attacks can only succeed if there is adequate alignment between the gradient spaces of the global and local models.

As a solution, they proposed FedKL, a specialized key-lock module that excels at differentiating, misaligning, and safeguarding the gradient spaces using a private key. This is accomplished while preserving federated aggregation comparable to conventional FL schemes. Specifically, the operations of scaling and shifting in the normalization layer are restructured. A private key, generated randomly, is fed into two fully-connected layers. The resulting outputs function as exclusive coefficients for the scaling and shifting procedures. Both theoretical analysis and experimental results affirm that the proposed key-lock module is efficient and effective in protecting against gradient leakage attacks. This is achieved by masking the uniformity of confidential data in the gradient, thus making it challenging for a malicious attacker to perform forward-backward propagation in the absence of the private key and the lock layer's gradient. Consequently, the task of approximating the shared gradient in the FL framework to reconstruct local training data becomes unachievable.

6. Composite Attacks

Table 8: Characteristics of Composite Attacks

Name of Attack	Distinctive Feature
Direct Boosting [123]	boosting malicious updates
Separated Boosting [123]	regularized update boosting
Model Replacement [124]	replace converging global model
PGD [125]	bounded update projection
Edge case + PGD [47]	PGD on minority samples
Median Interval [64]	median cheating with normalized updates
DBA[126]	distributed backdoor trigger
TrojanDBA [127]	distributed and learnable trigger
Neurotoxin [128]	tampering insignificant model weights
RL Neurotoxin [129]	searching Neurotoxin parameters with RL
F3BA [130]	sign-flipping on insignificant weights
Rare Word Embedding [131]	tampering stale word embeddings
Future Update Approximation [132]	estimating future updates from malicious clients
Sudden Collapse [133]	estimating potent malicious gradients

We define composite attacks as threat models that corrupt multiple aspects of FL. The attacker can combine D2M and M2M attacks to launch backdoor attacks. The attacker surreptitiously adds trigger patterns to local training data, then poisons model updates such that the global model learns how to react to triggers. Backdoored models behave normally when fed with clean data. In the presence of trigger data, these models are trained to give predictions designated by the attacker.

Trigger patterns vary from one attack to the other. We summarize existing triggers in Figure 10. Generic samples of a class or samples with shared patterns are commonly used in label-flipping attacks, these attacks can be further enhanced by incorporating M2M attacks. Triggers based on certain natural patterns are also known as semantic triggers [124] . Handpicked logos or icons are common trigger patterns for backdoor injection. Edge samples, namely samples at the tail of the data distribution, are used in attacks targeting underrepresented data, which can significantly damage the fairness for the minority group. Lastly, learnable triggers is a relatively new strategy appears in recent studies.

Compared to D2M or M2M attacks, now that the attacker also has control over client model updates, composite attacks tend to be stealthier and more destructive. A high-level view of such attacks is illustrated in Figure 11. We group recent composite attacks based on their most notable features. These attacks may also use techniques proposed in other groups. We show the characteristics of composite attacks in Table 8.

6.1. Composite Threat Models

6.1.1. Update Boosting

To boost the effectiveness of model updates derived from poisoned data, scaling up malicious updates is a common strategy in early studies on composite attacks [123, 124]. Given poisoned data with their labels being flipped, authors of [123] propose two types of threat models. The explicit approach is to train client models with the poisoned data, then boost model updates by scaling it up with a predefined coefficient. Although this approach is easy to implement, the boosted updates are statistically different from benign updates, suggesting that secure aggregation rules can easily identify boosted malicious updates. As for the stealthy approach in [123], the attacker instead trains client models on both the clean and poisoned data. Updates from the poisoned data are boosted as the explicit approach while a regularization term is used to ensure that the differences between current malicious updates and last round's average benign updates are bounded. Instead of boosting only the malicious updates, the model replacement attack proposed in [124] seeks to entirely replace the global model with the backdoored model. As the training goes on, benign updates from converging client models tend to cancel each other out. By solving the linear aggregation equation, the attacker can find the solution to scale up malicious updates such that the global model is equal to the model trained with poisoned data, namely the global model is replaced with the one with backdoors.

6.1.2. Bounded Updates

Boosting model updates is an effective way to inject backdoors. However, these updates have distinctive norms compared to benign updates. As mentioned above, boosted updates can be easily filtered out by norm-based aggregation rules. Projected Gradient Descent (PGD) proposed in [125] aims at bypassing norm-based aggregation by projecting boosted updates onto a small ball around the norm of global model weights. PGD can be also seen in later studies [47]. On top of the edge case D2M attack in [47], the attacker can further cover up their intention by projecting model updates derived from edge case data. Another threat model proposed in [47] combines PGD with model replacement [124] in which the boosted malicious updates is bounded through projection before replacing the global model. Another way to generate bounded updates is

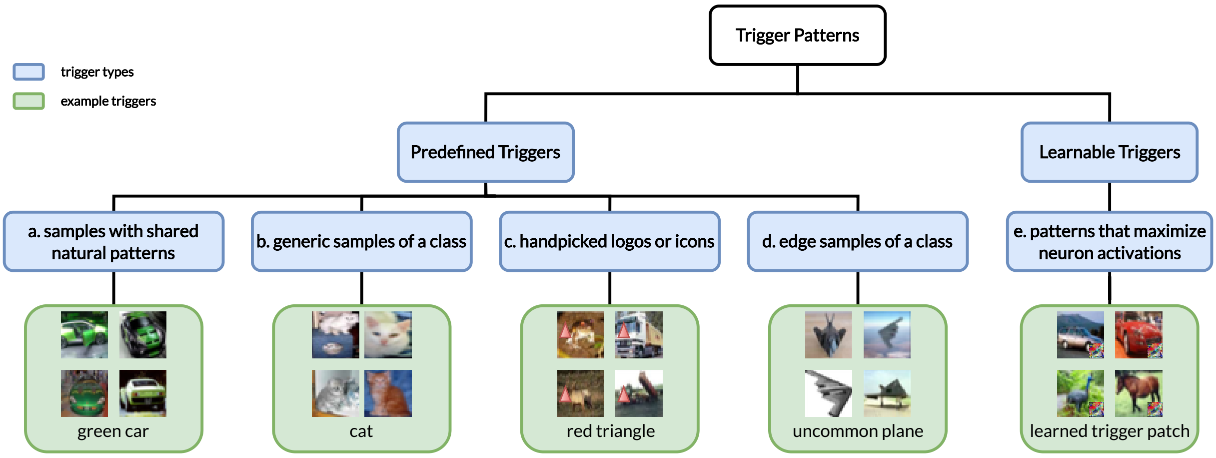


Figure 10: An Overview of Trigger Patterns. Among these trigger types, a and b are mostly associated with label-flipping. Type c is a common strategy for injecting triggers into arbitrary samples. Type d uses samples at the tail of the data distribution to induce erroneous predictions for underrepresented data. Type d appears in more recent studies.

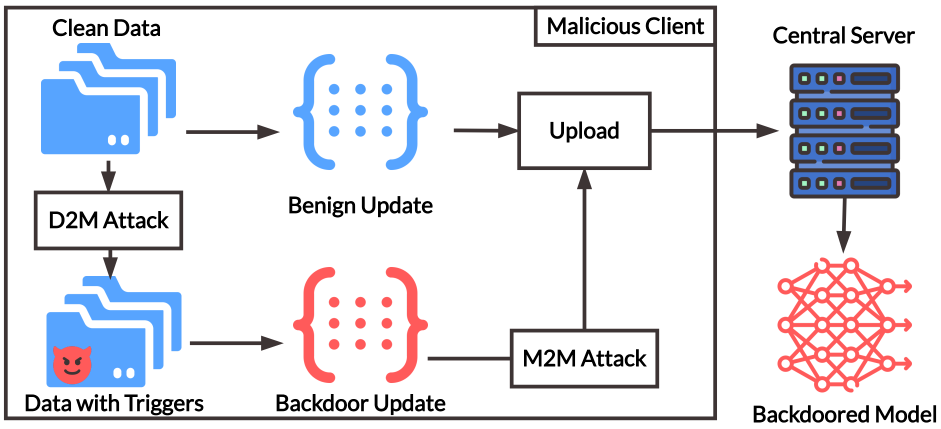


Figure 11: A High-level View of Injecting Backdoors with a Composite Attack. The attacker chooses a preferable trigger and tampers local data with the trigger. Local model is also trained on clean data to avoid detection. Most attacks aim at poisoning the global model with only a few clients.

proposed in [64]. In stead of projecting malicious updates, they are normalized by the maximum deviation range discussed in the M2M attack section.

6.1.3. Distributed Triggers

One common trait of the above composite attacks is that their backdoor triggers are stand-alone, namely the trigger patterns are identical across all clients and tampered samples. Even though there are experiments on concurrently employing multiple triggers [125], these triggers are still independent from each other and they lack the ability to collude. The Distributed Backdoor Attack (DBA) [126] instead assigns local triggers to multiple clients. Local triggers can be assembled to form a stronger global trigger. The triggers used in DBA is similar to the ones used in BadNets [134], which are colored rectangles placed around the corners of images. Malicious updates of DBA are scaled up by a coefficient similar to [123]. Another attack with distributed triggers is proposed in [127]. Unlike DBA whose triggers are predefined, triggers in [127] are based on [135] with learn-able parameters that generate local trigger patterns. In the trigger generation stage of [127], the attacker first determines the target class. By feeding various samples of the target class to the received global model, the attacker finds the internal neuron that is most sensitive to the target class. This is achieved by comparing the sum of connected weights and the number of activation. The attacker then optimizes trigger pattern parameters such that they maximize the activated value of the most sensitive neuron. In the distributed training stage of [127], each malicious client only trains from the most sensitive neuron's layer to the final output layer.

6.1.4. Insidious Tampering

More recent composite attacks focus on making malicious updates more insidious and persistent, which is usually achieved by tampering with weights that are unimportant to the clean data. For instance, Neurotoxin [128] only updates insignificant parameters to prevent backdoors from being erased by benign updates. Neurotoxin considers parameters with largest gradients to be most used by benign clients, therefore parameters with with smaller gradients are less accessed by benign clients. The attacker can only optimize less important parameters to achieve their backdoor objectives. Neurotoxin is recently enhanced by authors of [129] who employ RL to find better hyperparameters for the attack. Rare word embedding attack proposed in [131] shares a similar idea with Neurotoxin in the sense that it manipulates word embeddings of rare words as they are not likely to be updated by benign clients. The effectiveness of the rare word embedding attack can be further amplified by the gradient ensembling method [131]. The attacker intentionally stores the global models from multiple rounds, then gradients of backdoor word embeddings are computed for all these models. The exponential moving average of these gradients is used to update backdoor embeddings in the current round. Focused Flip Federated Backdoor Attack (F3BA) is a recent threat model that falls into the category of insidious tampering. Intuitively, F3BA tries to flip the signs of lease important weights such that they are most sensitive to trigger patterns. The importance of a weight is measured by the product of its gradient and weight value. F3BA only modifies least important weights found by this metric, and empirically 1% of weights are enough to degrade model performance. Sign-flipping of F3BA is conducted between consecutive layers. In the first layer, the attacker reshapes the trigger patterns such that it aligns with the convolution kernel. Signs of least important weights of this kernel are flipped if they are different from the signs of the aligned trigger pixels. In subsequent layers, the attacker respectively feeds the model with clean and poisoned data, records their activation differences, and flips signs of the chosen weights such that the activation differences are maximized. When sign-flipping is completed, the model is fine-tuned to associate flipped weights with the labels of poisoned data. The model's local updates will also be more similar to benign updates after fine-tuning. Like [127], trigger patterns is also learn-able. F3BA learns the trigger pattern's pixel values by maximizing the clean-poisoned activation difference of the first layer.

6.1.5. Update Approximation

Composite attacks introduced so far directly optimize model weights on the backdoor classification task. There are also attacks seeking to optimize niche objectives. These objectives are often intractable (*e.g.* estimating future updates of other clients), thus the attacker needs to find proper approximations to implement practical solutions. If an omniscient attacker knows all future updates of a FL system, the optimal way of injecting backdoors is differentiating through the computation graph of all future updates *w.r.t* the weights of the attacker's model. This is the intuition behind [132] and the authors propose a method to approximate updates in the near future. The attack in [132] requires the attacker to control a subset of client models. The attacker uses these models to simulate future updates by running FedAvg. Throughout the simulation, only clean data sampled from the malicious client is used. In the first round of the simulation, all models are fed with data. The malicious models are left out in the following rounds, which is simulating the scenario in which the malicious client is not chosen by the central server. Once future updates are approximated, client model weights are optimized through the classification losses on both clean and poisoned data similar to [123]. Accumulative Poisoning Attack (APA) [133] is another method that indirectly optimizes model weights for the backdoor task. The objective of APA is to clandestinely poison model weights while maintaining a good test performance. As soon as the model is fed with trigger data, its performance drastically drops, leaving the system administrator with minimum time to respond to the attack. APA learns two functions: an accumulative function and a poisoning function. The accumulative function is used to manipulate model updates such that the model is more sensitive to trigger gradients. The poisoning function is used to transform benign gradients from validation data into malicious gradients, leading to performance degradation. Intuitively, degrading model performance can be viewed as maximizing the validation loss. By taking the first order Taylor polynomial of the validation loss, the maximization problem is transformed into minimizing the first order gradient *w.r.t* the accumulative and poisoning functions. The authors of APA further simplify the minimization problem with its first order approximation. The final optimization objective then becomes simultaneously aligning the directions of poisoned gradients with benign gradients as well as the second order gradients of the validation loss. All gradients from APA are all projected through PGD [125] to enhance stealth. While it is not mandatory to use trigger patterns with APA, the authors demonstrate that explicit triggers makes APA more potent.

6.2. Defense Against Composite Attack

In this section, we introduce defenses that are specifically designed to counter D2M+M2M composite attacks. Since this type of attack also manipulates model weights or updates, defenses against M2M attacks such as Krum [61] or Bulyan [65] are also evaluated in many existing studies on defense against composite attacks. Depending on the subjects being processed by the defense strategy, we divide defenses again composite attacks into update cleansing and model cleansing.

6.2.1. Update Cleansing

Defenses based on update cleansing filter out uploads or mitigate influence from malicious clients by examining model updates. Robust-LR [136] is an update cleansing defense built on the heuristics that directions of malicious updates are different from benign ones. The authors of Robust-LR take a majority voting over model updates. The voting computes the sum of signs of model updates on each dimension. If the sum is below a pre-defined threshold, meaning that malicious clients participate in the current round of update, the learning rate on that dimension is multiplied by -1 to apply gradient ascent to suspicious updates.

Training models with DP has been mathematically proven as an effective way of defending against backdoor injections [137, 125]. This approach is first introduced to FL by authors of DP-FedAvg [138]. Compared to the vanilla FedAvg shown in Algorithm 1, DP-FedAvg requires the central server to bound client updates first. Client updates are clipped by comparing its $L2$-norm against a given parameter, which could be an overall parameter for all model weights or a set of layer-wise clipping parameter. When the global model is updated by taking in bounded client updates, noise from a zero-mean Gaussian is also added.

6.2.2. Model Cleansing

A pruning based method is proposed in [139]. This approach asks clients to rank the average activation values of the last layer of their models. The central server prunes neurons in the descending order based on the aggregated rankings of neurons. Knowledge distillation is also considered as a defense against composite backdoor attacks [140, 130]. By aligning the attention maps of the teacher model and the student model, Neural Attention Distillation (NAD) [140] manages to erase backdoors injected in the model. The distillation process of [140] assumes that clean data is available to the defender. This requirement is also inherited by FedRAD [141], a knowledge distillation based defense for FL. FedRAD needs to prepare synthetic data [142] on the central server for model evaluation. Client models are fed with the synthesized data for evaluation, then the central server counts how many times a client's logit obtains the median value for its corresponding class. The median frequencies of client models are normalized and used as global model aggregation coefficients. The distillation process of FedRAD is built on FedDF [143]. The central server distills knowledge from client models by minimizing the KL divergence between the global model's predictions and the average prediction of client models.

Some research considers certified robustness [144] as the way to defend against composite backdoor attacks. A ML model is said to have certified robustness if its predictions are still stable even if the input is perturbed. CRFL [145] is a defense designed to counter the model replacement attack. By controlling how the global model parameters update during training, CRFL grants the global model certified robustness under the condition that the backdoor trigger is bounded. Specifically, when the conventional global model aggregation completes, parameters of the global model are first clipped, then Gaussian noise is added to these parameters. At test time, a set of Gaussian noise is sampled from the previous noise distribution and added to the aggregated global model, resulting in a set of noisy global models. A majority voting is conducted among these noisy models to decide the classification results of test samples. Another defense with certified robustness is proposed in [146]. This method achieves certified robustness through the majority voting among a number of concurrently trained global models. Given n clients, the defense in [146] trains $\binom{n}{k}$ global models, where k is the number clients chosen without replacement for each model. Although the authors of [146] applies Monte Carlo approximation to speed up the defense, it still needs to train hundreds of global models, making this method more computationally expensive than other defenses.

The idea of majority voting is not exclusive to defenses with certified robustness. Authors of BaFFLe [147] rely on diversified client data to validate and provide feedback to the global model. BaFFLe adds an extra stage to conventional FL pipeline. When the global model for current global training round is aggregated, it is sent to randomly selected clients to validate if the global model is poisoned. A set of recently accepted global models are also sent to selected clients as reference. The validation process s of BaFFLe requires these clients to test global models with their local data. In particular, each client computes the misclassification rate for samples of a specific class, the client also computes the rate of other classes' samples being misclassified as the examined class. For benign models, the gap between these two rates are relatively stable during training. However, drastic changes can happen for backdoored models. If the misclassification gap of the newly aggregated global model deviates too much from the average gap of past models, the client votes the global model as malicious. Finally, based on the result of the majority voting, the central server decides whether to discard the newly obtained global model.

6.2.3. Composite Cleansing

Like composite attacks that manipulate multiple aspects of FL to enhance their capability, recent defenses also examine both model updates and weights to systematically mitigate composite attacks.

Authors of DeepSight [148] propose various metrics to evaluate if the upload from a client is malicious. The central server first computes the pairwise cosine similarities between received updates. Two other metrics, clients' Division Differences (DDif) and NormalizEd UPdate Energy (NEUP), are also computed. DDif measures the prediction differences between the global and client models. This is achieved by feeding models with random input on the server. Backdoored models are prone to produce larger activation for the trigger class even if the input is merely random noise [149], which is a telltale sign for DDif to identify compromised models. NEUP measures the update magnitude for neurons in the output layer. Local data with similar distributions results in models with similar NEUP patterns. Based on the above metrics, DeepSight clusters received client models on the central server with HDBSCAN [150]. The server also needs to maintain a classifier based on NEUP to label client models as either benign or malicious. Depending on the number of models being labeled as malicious, the server determines whether to accept or reject a client model cluster. Models from accepted clusters are deemed as safe for aggregation.

FLAME [151] is another example of composite defense. Authors of FLAME summarize the pipeline of their approach as clustering, clipping and noising. In the clustering stage, the central server computes CDs between model updates. HDBSCAN is subsequently used to filter out malicious models based on the angular differences derived from CDs. In the clipping stage, the median of remaining models' updates is chosen as the bound to clip model updates. In the final noising stage, Gaussian noise is added to the global model weights to further erase injected back doors.

Table 9: Summarization of defense techniques toward different types of attacks

Defense Method	Defense Strategy	Type of Attack	Attack Strategy
Fung et al.[43] (FoolsGold)	Dynamic learning rate	D2M	Label Attack Sample Attack
Tolpegin et al.[44]	Cluster for PCA		
Cao et al.[49] (Sniper)	Clique from Euclidean distance		
Ma et al.[53]	Rewards based aggregation		
Chen et al.[72] (GeoMed)	Geometric median	M2M	Priori Attack Posteriori Attack
Pillutla et al.[73] (RFA)	Weiszfeld-smoothed geometric median		
Xie et al.[62] (MarMed)	Dimension-wise median		
Xie et al.[62] (MeaMed)	Mean-around median		
Yin et al.[70] (TrimMean)	Dimension-wise trimmed mean		
Blanchard et al.[61] (Krum)	Euclidean distance		
El Mhamdi et al.[65] (Bulyan)	Euclidean distance		
Wang et al.[74] (ELITE)	Gradient information gain		
Tekgul et al.[79] (WAFFLE)	The server embeds watermarks		
Li et al.[80] (FedIPR)	Generate secret watermarks on client		
Lin et al.[76]	Auto-encoder		
Zong et al.[82] (DAGMM)	Gaussian mixture network		
Zhu et al.[103]	Adding noise to gradients	M2D	Attribute Inference Membership Identification Image Recovery
Chamikara et al.[117]	Perturbing data		
Wei et al.[121]	DP on data		
Scheliga et al.[122] (PRECODE)	Transform feature representation		
Ren et al.[102] (FedKL)	Hide the input from gradient		
Ozdayi et al.[136] (Robust-LR)	Update cleansing	Composite	Updates Attack Distributed Triggers Insidious Tampering
McMahan et al.[138] (DP-FedAvg)	DP		
Wu et al.[139]	Model pruning		
Sturluson et al.[141] (FedRAD)	Knowledge distillation		
Xie et al.[145] (CRFL)	Certified robustness from updates		
Cao et al.[146]	Certified robustness		
Andreina et al.[147] (BaFFLe)	Validation on diversified client data		
Rieger et al.[148] (DeepSight)	Various metrics		
Nguyen et al.[151] (FLAME)	Clustering, clipping and noising		

7. Conclusion and Future Directions

7.1. Conclusion

In recent years, FL has become a transformative paradigm for training ML models, especially in decentralized environments where data privacy and security are critical. Our comprehensive review categorized known FL attacks according to attack origin and target. It provides a clear structure for understanding the scope and depth of FL inherent vulnerabilities:

D2M Attacks: These attacks (*e.g.*, label-flipping) manipulate data to corrupt the global model. Since FL often relies on data from numerous potentially untrusted sources, it is highly vulnerable to such threats.

M2M Attacks: This type of attack tampers with model updates, thereby disrupting the learning process. For example, Byzantine attacks involve sending malformed or misleading model updates, indicating that one or more malicious clients have the potential to degrade the performance of the global model. Such attacks emphasize the importance of a robust aggregation approach in a federated environment.

M2D Attacks: Focus on exploiting vulnerabilities that arise when models interact with data, such as gradient leakage, where an attacker can infer private data from gradient updates. Gradient leakage is a prime example where malicious entities exploit the shared model updates to infer sensitive information about the training data, emphasizing on the need for defense strategies that mask or generalize gradients.

Composite Attacks: These attacks are more sophisticated in nature and often combine multiple attack methods or vectors to enhance their impact. Backdoor injection is a classic example, where an attacker subtly introduces a backdoor during training and then exploits it during reasoning.

A summarization of defense techniques toward different types of attacks is provided in Table 9

7.2. Future Directions

As FL continues to evolve, the sophistication of potential attacks will continue to increase. By reviewing the recent advancements in this domain, we identify several promising research directions that include:

Robust Aggregation Mechanisms: The aggregation process in FL is a key link where local model updates from different participants are combined to update the global model. Given its central role, the aggregation step becomes a vulnerable point, especially to malicious interference. For example, a single participant with malicious intentions may submit misleading updates with the intention of degrading the performance of the global model. This adverse activity is of particular concern in M2M attacks, of which the Byzantine attack is a prime example. In a Byzantine attack, an adversary sends arbitrary or strategically designed updates to a server with the intent of disrupting the aggregated model. Addressing these vulnerabilities requires re-evaluating and redesigning the traditional aggregation mechanisms used in FL. By delving into the development of more resilient aggregation strategies, methods can can be designed to identify, isolate, or reduce the impact of these malicious updates. These advanced aggregation techniques, based on robust statistical measures, consensus algorithms and even outlier detection methods, can ensure that the integrity of the global model remains intact in the presence of hostile participants.

Gradient Sparse Attack: In terms of M2D attack methods, it is worth noting that the gradients exchanged between the server and the client often contain a large amount of redundant details [107], and this redundancy may play a negative role in the effectiveness of the attack. If an attacker can filter out valuable gradients, the efficiency of the attack can be dramatically improved, especially in large-scale model training. This gradient sparse process eliminates irrelevant and noisy data, thus potentially improving the accuracy of the attack.

Automatic Attack Detection: As the complexity and scale of FL environments continues to grow, automated safety measures become critical. Meta-learning [152, 153, 154, 155], often referred to as "learning to learn", offers a promising avenue to address this challenge. By employing meta-learning techniques, systems can be trained to leverage prior knowledge about different types of attacks to quickly adapt to new, unforeseen threats. In addition, anomaly detection algorithms help identify outliers or unusual patterns in traditional datasets that can be fine-tuned for federated environments. These algorithms can monitor incoming model updates from different clients or nodes and flag any updates that deviate from the expected pattern to indicate potential malicious activity. Such an automated system not only identifies threats, but also combines with defense mechanisms to immediately counteract or eliminate suspicious activity, ensuring a smoother and safer FL process.

Holistic Defense Strategies: In the rapidly evolving FL environment, the need for holistic defense strategies is becoming increasingly prominent. These strategies advocate the development and implementation of defense mechanisms that are inherently versatile and capable of responding to multiple attack vectors simultaneously. A holistic approach would integrate various protection measures to create a more resilient and adaptive security framework, rather than a solo approach that develops defenses against specific threats. This multi-pronged defense system not only ensures broader security coverage, but also minimizes potential vulnerabilities and overlaps. As adversarial tactics become increasingly complex, utilizing an integrated solution that anticipates and responds to a wide range of threats will be key to protecting the FL ecosystem.

Domain-specific Attacks and Defenses Although we have witnessed nascent studies on exploiting the vulnerabilities in Federated Recommendation System and Federated RL, few defenses are proposed to defend against such threats. Furthermore, a majority of the current research tends to focus on image classification as the principal learning task for both attacks and defenses. This observation underscores a pressing need and opportunity to delve deeper into domain-specific threat models and tailored defense strategies for federated learning. Investigating this avenue not only holds promise for enhancing security but also ensures the more comprehensive protection of diverse applications within FL.

Interdisciplinary Approaches: Harnessing the wealth of insights from different fields is particularly instructive for enhancing FL systems. For example, frameworks and theories from disciplines such as game theory and behavioral science can help to understand the motivations and behaviors of participants in a FL environment. By understanding these motivations, tailored incentive structures or deterrence mechanisms can be designed to encourage positive contributions and discourage malicious or negligent behaviors in FL ecosystems. In addition, the fields of cryptography and cyber-security are constantly evolving, offering a plethora of innovative techniques and protocols. By integrating these advances into FL, we can strengthen systems against identified vulnerabilities and ensure not only the privacy and integrity of data, but also the trustworthiness of the learning process. As the stakes for FL grow, especially in critical areas of application, the convergence of these areas is critical to creating a robust, secure and collaborative learning environment.

Bibliography

[1] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al., Improving language understanding by generative pre-training (2018).

[2] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al., Language models are unsupervised multitask learners, OpenAI blog 1 (8) (2019) 9.

[3] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., Language models are few-shot learners, NIPs 33 (2020) 1877–1901.

[4] J. Ho, A. Jain, P. Abbeel, Denoising diffusion probabilistic models, NIPs 33 (2020) 6840–6851.

[5] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, S. Ganguli, Deep unsupervised learning using nonequilibrium thermodynamics, in: ICML, PMLR, 2015, pp. 2256–2265.

[6] Y. Song, S. Ermon, Generative modeling by estimating gradients of the data distribution, NIPs 32 (2019).

[7] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, B. Poole, Score-based generative modeling through stochastic differential equations, arXiv (2020).

[8] G. A. Kaissis, M. R. Makowski, D. Rückert, R. F. Braren, Secure, privacy-preserving and federated machine learning in medical imaging, NMI 2 (6) (2020) 305–311.

[9] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, D. Bacon, Federated learning: Strategies for improving communication efficiency, arXiv (2016).

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

[11] R. S. Antunes, C. André da Costa, A. Küderle, I. A. Yari, B. Eskofier, Federated learning for healthcare: Systematic review and architecture proposal, TIST 13 (4) (2022) 1–23.

[12] D. C. Nguyen, Q.-V. Pham, P. N. Pathirana, M. Ding, A. Seneviratne, Z. Lin, O. Dobre, W.-J. Hwang, Federated learning for smart healthcare: A survey, CSUR 55 (3) (2022) 1–37.

[13] J. Xu, B. S. Glicksberg, C. Su, P. Walker, J. Bian, F. Wang, Federated learning for healthcare informatics, JHIR 5 (2021) 1–19.

[14] G. Long, Y. Tan, J. Jiang, C. Zhang, Federated learning for open banking, in: FLPI, Springer, 2020, pp. 240–254.

[15] D. Byrd, A. Polychroniadou, Differentially private secure multi-party computation for federated learning in financial applications, in: ICAIF, 2020, pp. 1–9.

[16] W. Yang, Y. Zhang, K. Ye, L. Li, C.-Z. Xu, Ffd: A federated learning based method for credit card fraud detection, in: BigData, Springer, 2019, pp. 18–32.

[17] Z. Zheng, Y. Zhou, Y. Sun, Z. Wang, B. Liu, K. Li, Applications of federated learning in smart cities: recent advances, taxonomy, and open challenges, Connection Science 34 (1) (2022) 1–28.

[18] J. C. Jiang, B. Kantarci, S. Oktug, T. Soyata, Federated learning in smart city sensing: Challenges and opportunities, Sensors 20 (21) (2020) 6230.

[19] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, H. V. Poor, Federated learning for internet of things: A comprehensive survey, CST 23 (3) (2021) 1622–1658.

[20] H. Zhang, J. Bosch, H. H. Olsson, End-to-end federated learning for autonomous driving vehicles, in: IJCNN, IEEE, 2021, pp. 1–8.

[21] A. Nguyen, T. Do, M. Tran, B. X. Nguyen, C. Duong, T. Phan, E. Tjiputra, Q. D. Tran, Deep federated learning for autonomous driving, in: IV, IEEE, 2022, pp. 1824–1830.

[22] H. Zhang, J. Bosch, H. H. Olsson, Real-time end-to-end federated learning: An automotive case study, in: COMPSAC, IEEE, 2021, pp. 459–468.

[23] P. Kairouz, H. B. McMahan, et al., Advances and open problems in federated learning, Foundations and Trends® in Machine Learning (2021).

[24] L. Lyu, H. Yu, Q. Yang, Threats to federated learning: A survey, arXiv preprint arXiv:2003.02133 (2020).

[25] R. Zhang, S. Guo, J. Wang, X. Xie, D. Tao, A survey on gradient inversion: Attacks, defenses and future directions, arXiv preprint arXiv:2206.07284 (2022).

[26] Y. Liu, Y. Kang, T. Zou, Y. Pu, Y. He, X. Ye, Y. Ouyang, Y.-Q. Zhang, Q. Yang, Vertical federated learning, arXiv preprint arXiv:2211.12814 (2022).

[27] H. Zhu, J. Xu, S. Liu, Y. Jin, Federated learning on non-iid data: A survey, Neurocomput. (2021).

[28] M. Rasouli, T. Sun, R. Rajagopal, Fedgan: Federated generative adversarial networks for distributed data, arXiv preprint arXiv:2006.07228 (2020).

[29] M. Liu, S. Ho, M. Wang, L. Gao, Y. Jin, H. Zhang, Federated learning meets natural language processing: A survey, arXiv preprint arXiv:2107.12603 (2021).

[30] Y. Liu, A. Huang, Y. Luo, H. Huang, Y. Liu, Y. Chen, L. Feng, T. Chen, H. Yu, Q. Yang, Fedvision: An online visual object detection platform powered by federated learning, Proceedings of the AAAI Conference on Artificial Intelligence (2020).

[31] X. Li, K. Huang, W. Yang, S. Wang, Z. Zhang, On the convergence of fedavg on non-iid data, arXiv preprint arXiv:1907.02189 (2019).

[32] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, A. T. Suresh, Scaffold: Stochastic controlled averaging for federated learning, in: ICML, PMLR, 2020, pp. 5132–5143.

[33] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, V. Smith, Federated optimization in heterogeneous networks, Proceedings of Machine learning and systems 2 (2020) 429–450.

[34] S. Ji, S. Pan, G. Long, X. Li, J. Jiang, Z. Huang, Learning private neural language modeling with attentive aggregation, in: 2019 International joint conference on neural networks (IJCNN), IEEE, 2019, pp. 1–8.

[35] X. Wu, Z. Liang, J. Wang, Fedmed: A federated learning framework for language modeling, Sensors (2020).

[36] H. Ren, J. Deng, X. Xie, X. Ma, Y. Wang, Fedboosting: Federated learning with gradient protected boosting for text recognition, arXiv (2020).

[37] T. D. Nguyen, T. Nguyen, P. L. Nguyen, H. H. Pham, K. D. Doan, K.-S. Wong, Backdoor attacks and defenses in federated learning: Survey, challenges and future research directions, Engineering Applications of Artificial Intelligence 127 (2024) 107166.

[38] Y. Zhang, D. Zeng, J. Luo, Z. Xu, I. King, A survey of trustworthy federated learning with perspectives on security, robustness and privacy (2023).

[39] X. Gong, Y. Chen, Q. Wang, W. Kong, Backdoor attacks and defenses in federated learning: State-of-the-art, taxonomy, and future directions (2023).

[40] X. Yin, Y. Zhu, J. Hu, A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions (2021).

[41] J. Zhang, M. Li, S. Zeng, B. Xie, D. Zhao, A survey on security and privacy threats to federated learning (2021).

[42] B. Biggio, B. Nelson, P. Laskov, Poisoning attacks against support vector machines, in: ICICML, ICML'12, 2012, p. 1467–1474.

[43] C. Fung, C. J. Yoon, I. Beschastnikh, Mitigating sybils in federated learning poisoning, arXiv preprint arXiv:1808.04866 (2018).

[44] V. Tolpegin, S. Truex, M. E. Gursoy, L. Liu, Data poisoning attacks against federated learning systems, in: ESORICS 2020, Springer, 2020, pp. 480–501.

[45] H. Xiao, K. Rasul, R. Vollgraf, Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, arXiv preprint arXiv:1708.07747 (2017).

[46] J. Zhang, B. Chen, X. Cheng, H. T. T. Binh, S. Yu, Poisongan: Generative poisoning attacks against federated learning in edge computing systems, ITJ 8 (5) (2021) 3310–3322.

[47] H. Wang, K. Sreenivasan, S. Rajput, H. Vishwakarma, S. Agarwal, J.-y. Sohn, K. Lee, D. Papailiopoulos, Attack of the tails: Yes, you really can backdoor federated learning, NIPs 33 (2020) 16070–16084.

[48] Y. Sun, H. Ochiai, J. Sakuma, Semi-targeted model poisoning attack on federated learning via backward error analysis, in: IJCNN, IEEE, 2022, pp. 1–8.

[49] D. Cao, S. Chang, Z. Lin, G. Liu, D. Sun, Understanding distributed poisoning attack in federated learning, in: ICPADS, 2019.

[50] G. Sun, Y. Cong, J. Dong, Q. Wang, L. Lyu, J. Liu, Data poisoning attacks on federated machine learning, ITJ (2022).

[51] G. Costa, F. Pinelli, S. Soderi, G. Tolomei, Turning federated learning systems into covert channels, IEEE Access (2022).

[52] J. Shi, W. Wan, S. Hu, J. Lu, L. Y. Zhang, Challenges and approaches for mitigating byzantine attacks in federated learning, in: TrustCom, IEEE, 2022, pp. 139–146.

[53] E. Ma, R. Etesami, et al., Local environment poisoning attacks on federated reinforcement learning, arXiv preprint arXiv:2303.02725 (2023).

[54] M. Arazzi, M. Conti, A. Nocera, S. Picek, Turning privacy-preserving mechanisms against federated learning, arXiv preprint arXiv:2305.05355 (2023).

[55] Z. Liu, L. Yang, Z. Fan, H. Peng, P. S. Yu, Federated social recommendation with graph neural network, TIST 13 (4) (aug 2022).

[56] A. V. Clemente, H. N. Castejón, A. Chandra, Efficient parallel methods for deep reinforcement learning, arXiv preprint arXiv:1705.04862 (2017).

[57] Y. LeCun, The mnist database of handwritten digits, http://yann. lecun. com/exdb/mnist/ (1998).

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

[59] L. Lamport, R. Shostak, M. Pease, The byzantine generals problem, TPLS (1982).

[60] Y. Fraboni, R. Vidal, M. Lorenzi, Free-rider attacks on model aggregation in federated learning, in: ICAIS, PMLR, 2021, pp. 1846–1854.

[61] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, J. Stainer, Machine learning with adversaries: Byzantine tolerant gradient descent, NIPs 30 (2017).

[62] C. Xie, O. Koyejo, I. Gupta, Generalized byzantine-tolerant sgd, arXiv

preprint arXiv:1802.10116 (2018).

[63] L. Li, W. Xu, T. Chen, G. B. Giannakis, Q. Ling, Rsa: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets, in: AAAI, Vol. 33, 2019, pp. 1544–1551.

[64] G. Baruch, M. Baruch, Y. Goldberg, A little is enough: Circumventing defenses for distributed learning, NIPs 32 (2019).

[65] E. M. El Mhamdi, R. Guerraoui, S. L. A. Rouault, The hidden vulnerability of distributed learning in byzantium, in: ICML, 2018, p. 13.

[66] M. Fang, X. Cao, J. Jia, N. Gong, Local model poisoning attacks to {Byzantine-Robust} federated learning, in: 29th USENIX security symposium (USENIX Security 20), 2020, pp. 1605–1622.

[67] S. Zhang, H. Yin, T. Chen, Z. Huang, Q. V. H. Nguyen, L. Cui, Pipattack: Poisoning federated recommender systems for manipulating item promotion, in: ACM ICWSDM, 2022, pp. 1415–1423.

[68] D. Rong, S. Ye, R. Zhao, H. N. Yuen, J. Chen, Q. He, Fedrecattack: model poisoning attack to federated recommendation, in: ICDE, IEEE, 2022, pp. 2643–2655.

[69] D. Rong, Q. He, J. Chen, Poisoning deep learning based recommender model in federated learning scenarios, arXiv preprint arXiv:2204.13594 (2022).

[70] D. Yin, Y. Chen, R. Kannan, P. Bartlett, Byzantine-robust distributed learning: Towards optimal statistical rates, in: J. Dy, A. Krause (Eds.), ICML, Vol. 80 of Proceedings of Machine Learning Research, PMLR, 2018, pp. 5650–5659.

[71] M. Ammad-Ud-Din, E. Ivannikova, S. A. Khan, W. Oyomno, Q. Fu, K. E. Tan, A. Flanagan, Federated collaborative filtering for privacy-preserving personalized recommendation system, arXiv preprint arXiv:1901.09888 (2019).

[72] Y. Chen, L. Su, J. Xu, Distributed statistical machine learning in adversarial settings: Byzantine gradient descent, MACS 1 (2) (2017) 1–25.

[73] K. Pillutla, S. M. Kakade, Z. Harchaoui, Robust aggregation for federated learning, IEEE Transactions on Signal Processing (2022).

[74] Y. Wang, Y. Xia, Y. Zhan, Elite: Defending federated learning against byzantine attacks based on information entropy, in: CAC, 2021, pp. 6049–6054.

[75] E. Weiszfeld, F. Plastria, On the point for which the sum of the distances to n given points is minimum, Ann Oper Res (2009).

[76] J. Lin, M. Du, J. Liu, Free-riders in federated learning: Attacks and defenses, arXiv preprint arXiv:1911.12560 (2019).

[77] Y. Adi, C. Baum, M. Cisse, B. Pinkas, J. Keshet, Turning your weakness into a strength: Watermarking deep neural networks by backdooring, in: 27th USENIX Security Symposium (USENIX Security 18), 2018, pp. 1615–1631.

[78] Y. Uchida, Y. Nagai, S. Sakazawa, S. Satoh, Embedding watermarks into deep neural networks, in: ACM ICMR, 2017, pp. 269–277.

[79] B. A. Tekgul, Y. Xia, S. Marchal, N. Asokan, Waffle: Watermarking in federated learning (2021).

[80] B. Li, L. Fan, H. Gu, J. Li, Q. Yang, Fedipr: Ownership verification for federated deep neural network models, TPAMI 45 (4) (2022) 4521–4536.

[81] M. Sakurada, T. Yairi, Anomaly detection using autoencoders with nonlinear dimensionality reduction, in: MLSDAW, 2014, pp. 4–11.

[82] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, H. Chen, Deep autoencoding gaussian mixture model for unsupervised anomaly detection, in: ICLR, 2018.

[83] G. Ateniese, L. V. Mancini, A. Spognardi, A. Villani, D. Vitali, G. Felici, Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers, IJSN 10 (3) (2015) 137–150.

[84] L. E. Baum, T. Petrie, Statistical inference for probabilistic functions of finite state markov chains, The annals of mathematical statistics 37 (6) (1966) 1554–1563.

[85] B. E. Boser, I. M. Guyon, V. N. Vapnik, A training algorithm for optimal margin classifiers, in: CLTW, 1992, pp. 144–152.

[86] R. Shokri, M. Stronati, C. Song, V. Shmatikov, Membership inference attacks against machine learning models, in: SP, IEEE, 2017, pp. 3–18.

[87] R. McPherson, R. Shokri, V. Shmatikov, Defeating image obfuscation with deep learning, arXiv (2016).

[88] D. Carrell, B. Malin, J. Aberdeen, S. Bayer, C. Clark, B. Wellner, L. Hirschman, Hiding in plain sight: use of realistic surrogates to reduce exposure of protected health information in clinical text, JAMIIA 20 (2) (2013) 342–348.

[89] F. Li, Z. Sun, A. Li, B. Niu, H. Li, G. Cao, Hideme: Privacy-preserving photo sharing on social networks, in: INFOCOM, IEEE, 2019, pp. 154–162.

[90] L. C. AT&T, The database of faces (1994).

[91] H.-W. Ng, S. Winkler, A data-driven approach to cleaning large face datasets, in: ICIP, 2014, pp. 343–347.

[92] Y. Zhang, R. Jia, H. Pei, W. Wang, B. Li, D. Song, The secret revealer: Generative model-inversion attacks against deep neural networks, in: CVPR, 2020, pp. 253–261.

[93] X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri, R. M. Summers, Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases, in: CVPR, 2017, pp. 2097–2106.

[94] Z. Liu, P. Luo, X. Wang, X. Tang, Deep learning face attributes in the wild, in: ICCV, 2015, pp. 3730–3738.

[95] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv (2014).

[96] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: CVPR, 2016, pp. 770–778.

[97] Y. Cheng, J. Zhao, Z. Wang, Y. Xu, K. Jayashree, S. Shen, J. Feng, Know you at one glance: A compact vector representation for low-shot learning, in: ICCVW, 2017, pp. 1924–1932.

[98] B. Hitaj, G. Ateniese, F. Perez-Cruz, Deep models under the gan: Information leakage from collaborative deep learning, in: CCCS, 2017, pp. 603–618.

[99] L. Melis, C. Song, E. De Cristofaro, V. Shmatikov, Exploiting unintended feature leakage in collaborative learning, in: SP, 2019, pp. 691–706.

[100] Z. Li, J. Zhang, L. Liu, J. Liu, Auditing privacy defenses in federated learning via generative gradient leakage, in: CVPR, 2022, pp. 10132–10142.

[101] B. Zhao, K. R. Mopuri, H. Bilen, Idlg: Improved deep leakage from gradients, arXiv (2020).

[102] H. Ren, J. Deng, X. Xie, X. Ma, J. Ma, Gradient leakage defense with key-lock module for federated learning, arXiv (2023).

[103] L. Zhu, Z. Liu, S. Han, Deep leakage from gradients, NIPs 32 (2019).

[104] D. C. Liu, J. Nocedal, On the limited memory bfgs method for large scale optimization, Mathematical programming 45 (1-3) (1989) 503–528.

[105] J. Geiping, H. Bauermeister, H. Dröge, M. Moeller, Inverting gradients-how easy is it to break privacy in federated learning?, NIPs 33 (2020) 16937–16947.

[106] J. Jeon, K. Lee, S. Oh, J. Ok, Gradient inversion with generative image prior, NIPs 34 (2021) 29898–29908.

[107] H. Yin, A. Mallya, A. Vahdat, J. M. Alvarez, J. Kautz, P. Molchanov, See through gradients: Image batch recovery via gradinversion, in: CVPR, 2021, pp. 16337–16346.

[108] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, H. Qi, Beyond inferring class representatives: User-level privacy leakage from federated learning, in: ICCC, 2019, pp. 2512–2520.

[109] H. Ren, J. Deng, X. Xie, Grnn: Generative regression neural network—a data leakage attack for federated learning, TIST 13 (4) (2022) 1–24.

[110] X. Yang, Y. Feng, W. Fang, J. Shao, X. Tang, S.-T. Xia, R. Lu, An accuracy-lossless perturbation method for defending privacy attacks in federated learning, in: WC, 2022, pp. 732–742.

[111] L. Sun, J. Qian, X. Chen, LDP-FL: Practical private aggregation in federated learning with local differential privacy, arXiv (2020).

[112] J. Sun, A. Li, B. Wang, H. Yang, H. Li, Y. Chen, Soteria: Provable defense against privacy leakage in federated learning from representation perspective, in: CVPR, 2021, pp. 9307–9315.

[113] A. T. Hasan, Q. Jiang, J. Luo, C. Li, L. Chen, An effective value swapping method for privacy preserving data publishing, SCN 9 (16) (2016) 3219–3228.

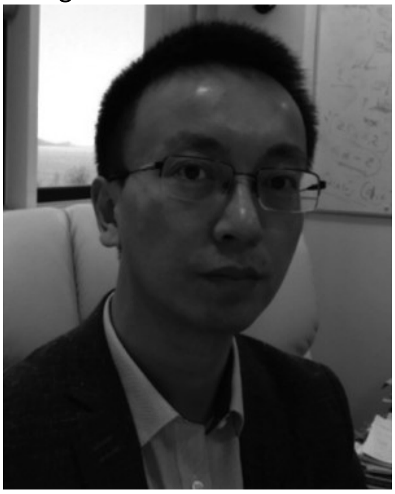
[114] M. A. P. Chamikara, P. Bertók, D. Liu, S. Camtepe, I. Khalil, Efficient data perturbation for privacy preserving and accurate data stream mining, PMC 48 (2018) 1–19.

[115] M. Chamikara, P. Bertok, D. Liu, S. Camtepe, I. Khalil, Efficient privacy preservation of big data for accurate data mining, IS 527 (2020) 420–443.

[116] H. Lee, J. Kim, S. Ahn, R. Hussain, S. Cho, J. Son, Digestive neural networks: A novel defense strategy against inference attacks in federated

learning, Computers & Security 109 (2021) 102378.
[117] M. A. P. Chamikara, P. Bertok, I. Khalil, D. Liu, S. Camtepe, Privacy preserving distributed machine learning with federated learning, Computer Communications 171 (2021) 112–125.
[118] Z. Bu, J. Dong, Q. Long, W. J. Su, Deep learning with gaussian differential privacy, Harvard data science review 2020 (23) (2020).
[119] Y. Li, Y. Zhou, A. Jolfaei, D. Yu, G. Xu, X. Zheng, Privacy-preserving federated learning framework based on chained secure multiparty computing, ITJ 8 (8) (2020) 6178–6186.
[120] K. Yadav, B. B. Gupta, K. T. Chui, K. Psannis, Differential privacy approach to solve gradient leakage attack in a federated machine learning environment, in: ICCDSN, Springer, 2020, pp. 378–385.
[121] W. Wei, L. Liu, Y. Wut, G. Su, A. Iyengar, Gradient-leakage resilient federated learning, in: ICDCS, IEEE, 2021, pp. 797–807.
[122] D. Scheliga, P. Mäder, M. Seeland, Precode-a generic model extension to prevent deep gradient leakage, in: WCACV, 2022, pp. 1849–1858.
[123] A. N. Bhagoji, S. Chakraborty, P. Mittal, S. Calo, Analyzing federated learning through an adversarial lens, in: ICML, PMLR, 2019, pp. 634–643.
[124] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, V. Shmatikov, How to backdoor federated learning, in: ICAIS, PMLR, 2020, pp. 2938–2948.
[125] Z. Sun, P. Kairouz, A. T. Suresh, H. B. McMahan, Can you really backdoor federated learning?, arXiv preprint arXiv:1911.07963 (2019).
[126] C. Xie, K. Huang, P.-Y. Chen, B. Li, Dba: Distributed backdoor attacks against federated learning, in: ICLR, 2019.
[127] X. Gong, Y. Chen, H. Huang, Y. Liao, S. Wang, Q. Wang, Coordinated backdoor attacks against federated learning with model-dependent triggers, IEEE Network 36 (1) (2022) 84–90.
[128] Z. Zhang, A. Panda, L. Song, Y. Yang, M. Mahoney, P. Mittal, R. Kannan, J. Gonzalez, Neurotoxin: Durable backdoors in federated learning, in: ICML, 2022, pp. 26429–26446.
[129] H. Li, C. Wu, S. Zhu, Z. Zheng, Learning to backdoor federated learning, arXiv preprint arXiv:2303.03320 (2023).
[130] P. Fang, J. Chen, On the vulnerability of backdoor defenses for federated learning, arXiv preprint arXiv:2301.08170 (2023).
[131] K. Yoo, N. Kwak, Backdoor attacks in federated learning by rare embeddings and gradient ensembling, arXiv preprint arXiv:2204.14017 (2022).
[132] Y. Wen, J. Geiping, L. Fowl, H. Souri, R. Chellappa, M. Goldblum, T. Goldstein, Thinking two moves ahead: Anticipating other users improves backdoor attacks in federated learning, arXiv preprint arXiv:2210.09305 (2022).
[133] T. Pang, X. Yang, Y. Dong, H. Su, J. Zhu, Accumulative poisoning attacks on real-time data, NIPs 34 (2021) 2899–2912.
[134] T. Gu, B. Dolan-Gavitt, S. Garg, Badnets: Identifying vulnerabilities in the machine learning model supply chain, arXiv preprint arXiv:1708.06733 (2017).
[135] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, X. Zhang, Trojaning attack on neural networks, in: NDSS, Internet Soc, 2018.
[136] M. S. Ozdayi, M. Kantarcioglu, Y. R. Gel, Defending against backdoors in federated learning with robust learning rate, in: AAAI, Vol. 35, 2021, pp. 9268–9276.
[137] Y. Ma, X. Zhu, J. Hsu, Data poisoning against differentially-private learners: Attacks and defenses, arXiv preprint arXiv:1903.09860 (2019).
[138] H. B. McMahan, D. Ramage, K. Talwar, L. Zhang, Learning differentially private recurrent language models, arXiv preprint arXiv:1710.06963 (2017).
[139] C. Wu, X. Yang, S. Zhu, P. Mitra, Mitigating backdoor attacks in federated learning, arXiv preprint arXiv:2011.01767 (2020).
[140] Y. Li, X. Lyu, N. Koren, L. Lyu, B. Li, X. Ma, Neural attention distillation: Erasing backdoor triggers from deep neural networks, arXiv preprint arXiv:2101.05930 (2021).
[141] S. P. Sturluson, S. Trew, L. Muñoz-González, M. Grama, J. Passerat-Palmbach, D. Rueckert, A. Alansary, Fedrad: Federated robust adaptive distillation, arXiv preprint arXiv:2112.01405 (2021).
[142] G. K. Nayak, K. R. Mopuri, V. Shaj, V. B. Radhakrishnan, A. Chakraborty, Zero-shot knowledge distillation in deep networks, in: ICML, PMLR, 2019, pp. 4743–4751.
[143] T. Lin, L. Kong, S. U. Stich, M. Jaggi, Ensemble distillation for robust model fusion in federated learning, NIPs 33 (2020) 2351–2363.
[144] M. Lecuyer, V. Atlidakis, R. Geambasu, D. Hsu, S. Jana, Certified robustness to adversarial examples with differential privacy, in: SP, IEEE, 2019, pp. 656–672.
[145] C. Xie, M. Chen, P.-Y. Chen, B. Li, Crfl: Certifiably robust federated learning against backdoor attacks, in: ICML, PMLR, 2021, pp. 11372–11382.
[146] X. Cao, Z. Zhang, J. Jia, N. Z. Gong, Flcert: Provably secure federated learning against poisoning attacks, TIFS (2022).
[147] S. Andreina, G. A. Marson, H. Möllering, G. Karame, Baffle: Backdoor detection via feedback-based federated learning, in: ICDCS, IEEE, 2021, pp. 852–863.
[148] P. Rieger, T. D. Nguyen, M. Miettinen, A.-R. Sadeghi, Deepsight: Mitigating backdoor attacks in federated learning through deep model inspection, arXiv preprint arXiv:2201.00763 (2022).
[149] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, X. Zhang, Trojaning attack on neural networks, in: 25th Annual Network And Distributed System Security Symposium (NDSS 2018), Internet Soc, 2018.
[150] R. J. Campello, D. Moulavi, J. Sander, Density-based clustering based on hierarchical density estimates, in: Pacific-Asia conference on knowledge discovery and data mining, Springer, 2013, pp. 160–172.
[151] T. D. Nguyen, P. Rieger, e. a. De Viti, {FLAME}: Taming backdoors in federated learning, in: 31st USENIX Security Symposium (USENIX Security 22), 2022, pp. 1415–1432.
[152] C. Finn, P. Abbeel, S. Levine, Model-agnostic meta-learning for fast adaptation of deep networks, in: ICML, PMLR, 2017, pp. 1126–1135.
[153] J. Snell, K. Swersky, R. Zemel, Prototypical networks for few-shot learning, NIPs 30 (2017).
[154] K. Lee, S. Maji, A. Ravichandran, S. Soatto, Meta-learning with differentiable convex optimization, in: CVPR, 2019, pp. 10657–10665.
[155] K. Cao, J. You, J. Leskovec, Relational multi-task learning: Modeling relations between data and tasks, arXiv preprint arXiv:2303.07666 (2023).

Xianghua Xie



received the M.Sc. and Ph.D. degrees in computer science from the University of Bristol, Bristol, U.K., in 2002 and 2006, respectively. He is currently a Full Professor with the Department of Computer Science, Swansea University, Swansea, U.K., and is leading the Computer Vision and Machine Leaning Laboratory, Swansea University. He has published around 200 refereed conference and journal publications and (co-)edited several conference proceedings. His research interests include various aspects of pattern recognition and machine intelligence and their applications to real-world problems. He is a member of BMVA. He is an Associate Editor of a number of journals, including Pattern Recognition and IET Computer Vision.

Chen Hu

received the M.Sc. by Research degree in Visual Computing from Swansea University. He is currently a computer science Ph.D. student at Swansea University. His research area includes federated learning and generative models.

Hanchi Ren

received the M.Sc. and Ph.D. degree in computer science from Swansea University, U.K., in 2016 and 2023. He is currently an academic tutor in the Computer Vision and Machine Learning Laboratory, Department of Computer Science, Swansea University. His research subject is on privacy-preserving federated learning and machine learning and artificial intelligence in general and applications in computer vision and medical image analysis.

Jingjing Deng

received his Ph.D. in Visual Computing from Swansea University, UK, in 2017. Presently, he holds the position of Assistant Professor in the Department of Computer Science at Durham University, UK. He founded the Rand2AI Lab in 2022 which actively engages in cutting-edge research in computer vision and artificial intelligence. In recent years, the team has focused on developing computational models that can cultivate and generalize intelligence from and for the complex world.

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐ The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: