3DFed: Adaptive and Extensible Framework for Covert Backdoor Attack in Federated Learning

Haoyang Li[1], Qingqing Ye[1], Haibo Hu[1]* , Jin Li[2], Leixia Wang[3], Chengfang Fang[4], Jie Shi[4]
[1]The Hong Kong Polytechnic University, *hao-yang9905.li@connect.polyu.hk, {qqing.ye, haibo.hu}@polyu.edu.hk*
[2]Guangzhou University, *lijin@gzhu.edu.cn*, [3]Renmin University of China, *leixiawang@ruc.edu.cn*
[4]Huawei International, Singapore, *{fang.chengfang, shi.jie1}@huawei.com*

2023 IEEE Symposium on Security and Privacy (SP) | 978-1-6654-9336-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/SP46215.2023.10179401

***Abstract*—Federated Learning (FL), the de-facto distributed machine learning paradigm that locally trains datasets at individual devices, is vulnerable to backdoor model poisoning attacks. By compromising or impersonating those devices, an attacker can upload crafted malicious model updates to manipulate the global model with backdoor behavior upon attacker-specified triggers. However, existing backdoor attacks require more information on the victim FL system beyond a practical black-box setting. Furthermore, they are often specialized to optimize for a single objective, which becomes ineffective as modern FL systems tend to adopt in-depth defense that detects backdoor models from different perspectives. Motivated by these concerns, in this paper, we propose 3DFed, an adaptive, extensible, and multi-layered framework to launch covert FL backdoor attacks in a black-box setting. 3DFed sports three evasion modules that camouflage backdoor models: backdoor training with constrained loss, noise mask, and decoy model. By implanting indicators into a backdoor model, 3DFed can obtain the attack feedback in the previous epoch from the global model and dynamically adjust the hyper-parameters of these backdoor evasion modules. Through extensive experimental results, we show that when all its components work together, 3DFed can evade the detection of all state-of-the-art FL backdoor defenses, including Deepsight, Foolsgold, FLAME, FL-Detector, and RFLBAT. New evasion modules can also be incorporated in 3DFed in the future as it is an extensible framework.**

I. Introduction

Thanks to its privacy protection and resource efficiency, Federated Learning (FL) has become the de-facto distributed machine learning paradigm in AI research and industry [21], [24], [26], [45], [48]. However, the vulnerability of participating devices, especially those end devices such as smartphones, IoT, and edge devices, opens the door for attackers to compromise or impersonate them and attack FL systems. As illustrated in Figure 1, such devices can upload poisoned gradient updates to manipulate the resulting global model to satisfy malicious intents, a.k.a., *model poisoning attacks* [4], [6], [15]. Based on their intents, model poisoning attacks can be classified into targeted and untargeted attacks [26]. Unlike untargeted attacks, which aim to impede the convergence of global models [3], [6], targeted attacks cause these models to perform abnormally on attacker-specified tasks [4].

As a special targeted model poisoning attack, the backdoor attack manipulates the global model to activate a backdoor behavior when test examples contain attacker-specified triggers [1], [2], [26]. Since the poisoned global model behaves normally for test examples without triggers, most defense mechanisms focus on identifying backdoor attacks during the FL training phase [17], [28], [51], [53]. Recently, there has been a trend for FL systems to adopt multi-layered defense that incorporates multiple backdoor detection mechanisms from different perspectives [31], [33]. This has been very effective because most backdoor attacks attempt to optimize a single objective in the training phase, e.g., by adding a supplementary term to the loss function or projecting gradients to satisfy a single constraint [2], [4], [40], [52], [57]. As such, it is difficult or even impossible to design a backdoor attack that satisfies multiple objectives or constraints simultaneously to evade a multi-layered defense.

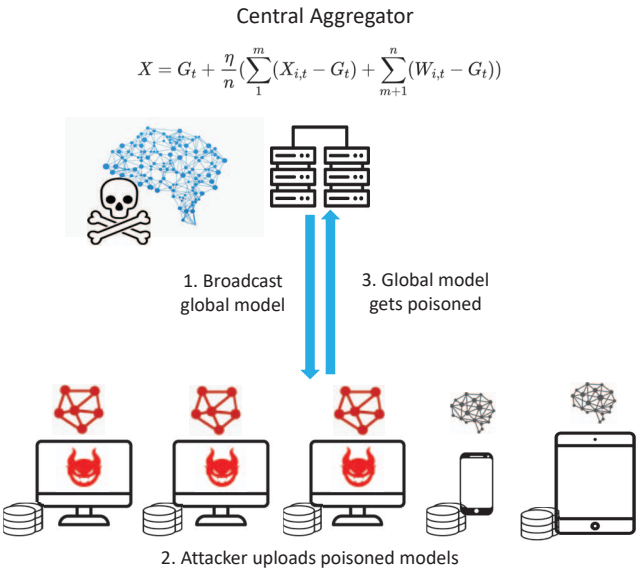


Fig. 1: **Model Poisoning Attacks in Federated Learning**

Besides, existing backdoor attacks often **require information about the FL system beyond a black-box setting**. For example, the model replacement attack in [2] needs the global learning rate, the total number of participants, and even the clipping bound of the central aggregator to calculate the scaling factor. Although the black-box versions of some untargeted poisoning attacks exist, they often suffer from significant degradation in terms of attack success rates [6], [35].

Motivated by these challenges in FL backdoor attacks, in this paper, we introduce 3DFed, an adaptive and extensible framework to launch covert FL backdoor attacks in the black-

* Corresponding author

DOI 10.1109/SP46215.2023.00158

box scenario. "3D" stands for both a multi-layered evasion structure and the three key components in this framework, namely indicator, adaptive tuning, and decoy models. To begin with, 3DFed boasts an indicator mechanism, which helps the attacker to obtain feedback from the global model on the attack in the preceding epoch, so that various camouflage modules of 3DFed can estimate those hyper-parameters in the FL system and adaptively adjust their strategies accordingly. 3DFed also sports three orthogonal evasion modules that camouflage a backdoor model from different perspectives. Specifically, backdoor training has been enhanced with a constraining term to counteract the norm clipping defense; noise masks alleviate the over-concentrated neuron updates and high pairwise cosine similarities in backdoor models; decoy models, which are uploaded together with backdoor models, can confuse the dimension reduction defense. Furthermore, 3DFed is an extensible framework as it can incorporate new backdoor evasion modules in the future by extending its process flow.

To summarize, our contributions in this paper are as follows:

- To the best of our knowledge, 3DFed is the first backdoor attack framework that integrates multiple evasion strategies to launch covert, adaptive, and extensible FL backdoor attacks in a black-box setting.
- We propose the indicator mechanism, which empowers the attacker to obtain essential feedback from his attack in the preceding epoch. Based on indicators, all evasive strategies can support adaptive tuning to work in black-box FL systems.
- We propose noise masks that extenuate neuron updates' concentration and sparsify backdoor models' distribution.
- We introduce the notion of decoy model and demonstrate that by uploading additional decoy models, the attacker can easily fool the dimension reduction defense in FL systems.
- We conduct extensive experiments and verify that 3DFed successfully evades the detection of state-of-the-art FL backdoor defenses, including Deepsight [33], Foolsgold [9], FLAME [31], FL-Detector [51], and RFLBAT [41].

The rest of this paper is organized as follows. In Section II and Section III, we review existing FL backdoor attacks and defenses and formulate the problem. Then 3DFed is overviewed in Section IV, followed by four sections to describe its core components elaborately. Specifically, Section V introduces indicators and presents mechanisms to implant them in a model and obtain the attack feedback in the preceding epoch. Section VI analyzes the constraining term for backdoor training against norm clipping. Section VII introduces noise masks for backdoor models after the backdoor training. In Section VIII, we show how the attacker can fool some detection methods by uploading decoy models. In Section IX, we conduct extensive experiments on several popular datasets to verify that 3DFed defeats state-of-the-art FL backdoor defenses. Ablation study is also carried out to demonstrate the robustness and effectiveness of various 3DFed evasion modules. In Section X, we discuss several potential adaptive defenses against 3DFed, followed by the conclusion in Section XI.

II. Related Work

A. FL Backdoor Attacks

It is well known that a close-to-convergence global model has few significant gradient updates, so an attacker can scale the backdoor update with a scaling factor γ to replace the global model with an attacker-trained backdoor model in one epoch, a.k.a. the Model Replacement Attack [2]. Exploiting the distributed nature of FL, Distributed Backdoor Attack (DBA) decomposes the global trigger pattern to multiple local triggers and assigns each compromised device with a unique local trigger [47]. Based on DBA, Gong *et al.* [11] propose a coordinate backdoor attack with local triggers being model dependent. Semantic backdoor is a variant of FL backdoor attack, where the trigger in the semantic backdoor can be some inherent features in the target images, such as the color of a car [2]. Bagdasaryan *et al.* [2] point out that the success and persistence of a semantic backdoor depend on whether the trigger's features frequently occur in other clients' datasets. As such, Wang *et al.* [40] propose an edge-case backdoor, which is similar to the semantic backdoor but ensures that the backdoor datasets are at the tail of the global datasets' distribution, so that the attacker's datasets are unlikely to appear on other clients' devices.

B. FL Backdoor Defenses

In the literature, there are various outlier detection techniques against backdoor attacks. Deepsight performs deep model inspection for each model by checking its Normalized Update Energies (NEUPs) and Division Differences (DDifs) [33]. Foolsgold examines the historical updates for each client and punishes those with high pairwise cosine similarities by a low learning rate [9]. FL-Detector predicts the global model by model update consistency and detects the outlier based on the distance to the predicted model [51]. RFLBAT applies Principal Component Analysis (PCA) to reduce the dimension of gradient updates so that malicious models are separated from benign models in the low dimensional projection space [41]. There is another line of research on robust defense to mitigate FL backdoor attacks by applying weak Differential Privacy (DP) [56] to the global model [55]. Through norm clipping and adding Gaussian noise to each gradient update, weak DP is shown to mitigate FL backdoor attacks effectively [46]. As Gaussian noise may deteriorate the global model's main task accuracy and norm clipping requires a clipping bound, FLAME adapts the weak DP method by noise boundary proof and a dynamic clipping bound, which is shown to alleviate the backdoor attack while still retaining a high main task accuracy [31].

III. Problem Formulation

This paper adopts a classic FL paradigm, where a central server broadcasts the global model and receives gradient

Notation	Meaning
Parameters defined by the central aggregator	
η, lr	Global and local learning rate
$n, \mathcal{L}, \mathcal{D}$	# of participants, loss function, and local dataset for SGD
$G_t, W_{i,t}$	Global model and client i's local model at epoch t
Parameters used by the attacker	
m, k_t	# of backdoor models and decoy models at epoch t
ω_{adv}, ω_b	Attacker's trained backdoor and benign reference model
$\hat{\Delta} = \{\Delta_1, \Delta_2, ..., \Delta_m\}, \lambda$	A set of noise masks and Lagrange multiplier
$X, X^{'}, S, S^{'}$	Noise-masked backdoor models and decoy models without and with indicators
$\beta, \hat{\alpha} = \{\alpha_{t,i}\}$	Hyper-parameters for backdoor training and noise masks
I_t, A	An index set and acceptance status
θ_{ω,I_i}	The parameter of model ω at index I_i

TABLE I: **Table of Notations**

updates trained by n clients using their local datasets. The notations in this paper are summarized in Table I.

Federated Learning. Each user i who holds a batch of private training data $\mathcal{D}_i$ iteratively optimizes the global model G_t according to the protocol (e.g., FedAvg) specified by the central server. For example, when Stochastic Gradient Descent (SGD) is the optimization method, each participant i at t-th epoch first generates a local model $W_{i,t}$ as follows:

$$W_{i,t} = G_t - lr \cdot \nabla \mathcal{L}_{task}(G_t, \mathcal{D}_i) \tag{1}$$

$\nabla \mathcal{L}_{task}(G_t, \mathcal{D}_i)$ denotes the gradient of cross-entropy loss for the learning task of model G_t and dataset $\mathcal{D}_i$. lr is the local learning rate. After each user uploads $W_{i,t}$, the central aggregator will aggregate them by an aggregation rule (e.g., FedAvg) and obtain the global model G_{t+1} for the next epoch $t+1$. Without loss of generality, we adopt FedAvg with global learning rate η in this paper:

$$G_{t+1} = G_t + \frac{\eta}{n} \sum_{i=1}^{n} (W_{i,t} - G_t) \tag{2}$$

Backdoor Attack. An attacker with backdoor datasets can participate in FL by either compromising existing devices or stealing the credentials of benign users. Through violating the FL protocol, the attacker can train a backdoor model $\omega_{adv,t}$ with both backdoor and benign datasets $\mathcal{D} = \{\mathcal{D}_{normal}, \mathcal{D}_{backdoor}\}$:

$$\begin{aligned} W_{adv,t} = G_t - lr \cdot \nabla((1-\beta) \cdot \mathcal{L}_{task}(G_t, \mathcal{D}) \\ + \beta \cdot \mathcal{L}_{constrain}) \end{aligned} \tag{3}$$

$$\omega_{adv,t} = G_t + \gamma \cdot (W_{adv,t} - G_t), \tag{4}$$

where $\mathcal{L}_{constrain}$ refers to the constraining term that limits the loss optimization, β is the weight parameter of $\mathcal{L}_{constrain}$, and γ is a scaling factor of backdoor update $W_{adv,t} - G_t$.

Backdoor Defense. An advanced central server aware of the potential backdoor attack will deploy a certain defense algorithm, noted as $Aggr()$, in its central aggregator to detect backdoor models in every epoch. After receiving all the local models $\hat{W}_t = \{\omega_{adv,t}, W_{1,t}, W_{2,t}, ..., W_{n-1,t}\}$, the central aggregator will obtain the aggregated global model G_{t+1} through replacing the Equation 2 by $G_{t+1} = Aggr(\hat{W}_t)$, so that the backdoor models in $\hat{W}_t$ will be detected and removed.

Problem Definition. In this paper, we aim to design covert backdoor attacks that can adaptively adjust their strategies without having white-box knowledge of the server's defense strategy. In every epoch, the attacker trains a set of malicious models $\hat{W}_{adv,t} = \{X_1, ..., X_m, S_1, ..., S_k\}$ as the solution for the following optimization problem:

$$\min_{\hat{W}_{adv,t}} \mathcal{L}_{backdoor}(Aggr(\hat{W}_{adv,t}, \hat{W}_{b,t}), \mathcal{D}^{'}), \tag{5}$$

where $Aggr(\hat{W}_{adv,t}, \hat{W}_{b,t})$ denotes the aggregated global model using $\hat{W}_{adv,t}$ and other benign updates $\hat{W}_{b,t}$ by unknown server-side defense mechanism $Aggr()$. $\mathcal{L}_{backdoor}(W_{i,t}, \mathcal{D}^{'})$ is the cross-entropy loss of model $W_{i,t}$ and the test datasets $\mathcal{D}^{'}$, which measures the misclassification degree to the target label. In this paper, we assume the attacker does not have additional power, i.e., does not know anything other than his local datasets. In other words, $Aggr()$ and $\hat{W}_{b,t}$ are agnostic to the attacker.

IV. 3DFed Overview

Figure 2 illustrates 3DFed framework for covert backdoor attacks against three example defenses, namely, norm clipping, deep model inspection, and dimension reduction. To evade these defenses, it consists of one adaptive mechanism (indicator) and three camouflage modules (backdoor training with constraints, noise masks, and decoy model). The details of 3DFed are shown in Algorithm 1. Taking the global model G_t, attacker's datasets $\mathcal{D}$, and the preceding-epoch attack settings $\alpha_{t-1}, k_{t-1}, I_{t-1}$ as input, 3DFed first reads the indicator of the global model and determines the attack strategy in this epoch through adaptive tuning (Lines 1 and 2). After that, the attacker performs three camouflage modules in sequence (Lines 3-5). Finally, all the generated backdoor and decoy models are implanted with indicators so the attacker can receive feedback again in the next epoch of poisoning (Line 6).

The details of adaptive mechanism and camouflage modules will be elaborated in the following four sections. Specifically,

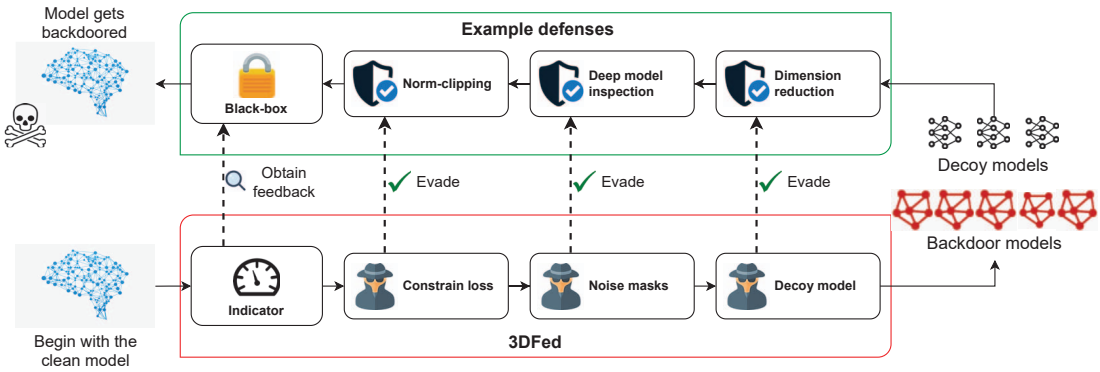


Fig. 2: **Overview of 3DFed Framework against Three Example Backdoor Defenses**

the implantation and feedback of indicator will be discussed in Section V. Section VI studies the selection of constraining term for backdoor training. In Section VII, we design noise masks to alleviate the over-concentrated neuron updates and high pairwise cosine similarities in backdoor models. In Section VIII, we propose decoy models to evade backdoor detection by dimension reduction.

Algorithm 1 3DFed

Input: Global model G_t, Local dataset $\mathcal{D}$, Previous attack settings α_{t-1},k_{t-1},I_{t-1}

Output: Backdoor models $X^{'} = \{X_1, X_2, ..., X_m\}$, decoy models $S^{'} = \{S_1, S_2, ..., S_k\}$

1: $A \leftarrow ReadIndicator(G_t, I_{t-1})$
2: $\alpha_t, k_t \leftarrow AdaptiveTuning(\alpha_{t-1}, k_{t-1}, A)$
3: $\omega_{adv} \leftarrow BackdoorTraining(G_t, \mathcal{D})$
4: $X \leftarrow NoiseMaskDesign(\omega_{adv}, A, \alpha_t)$
5: $S \leftarrow DecoyModelDesign(\mathcal{D}, X, A, S_{t-1}, k_{t-1})$
6: $X^{'}, S^{'}, I_t \leftarrow IndicatorDesign(X, S, \mathcal{D})$
7: **return** $X^{'}, S^{'}$

V. Indicator

The main limitation of existing backdoor and other model poisoning attacks in FL is that they either assume a white-box defense mechanism that they can clearly evade [1], [2] or accept a compromise for the black-box scenario with an attack performance drop [6], [35]. To eliminate this limitation and be more practical, in 3DFed we propose the indicator mechanism that can track the acceptance status of a malicious model by the central aggregator. The key idea of the indicator is that some neurons in the global model do not change much especially when the global model is already close to convergence [57]. We call them *redundant neurons*, as they are insensitive to most benign and malicious gradient updates. By changing the parameters of redundant neurons in a backdoor model to unique values, the attacker can infer whether this model has been successfully accepted by checking the global model's changes on those neurons. Besides, these neurons should be invariant to the classification tasks, so variations in their parameters have little impact on model's performance.

Indicator is essential in 3DFed, which provides the critical information about whether the last-epoch's attack is successful and boasts adaptive tuning of hyper-parameters in the later attack algorithms. The details of adaptive turning for each camouflage modules are discussed in their respective sections.

A. Finding Indicators

There are two necessary conditions for a redundant neuron:

C1. The gradient update in this neuron is much smaller in magnitude than those in other neurons.

C2. A small change in the parameters of this neuron does not significantly increase the tasks' loss.

Algorithm 2 illustrates the procedures of finding such neurons as indicators. Taking backdoor and decoy models $\{X, S\}$, attacker's local datasets $\mathcal{D}$, and the global model G_t as input, it first searches parameters with the smallest absolute value in the gradient update for global model G_t. However, since we do not assume the attacker has any information about the datasets from benign participants, the true gradient update of the global model in this epoch is invisible to the attacker. As such, the attacker uses his local datasets to obtain an approximate gradient $\hat{\omega}_G$ for G_t by cross-entropy loss function $\mathcal{L}_{task}$ with the input of $\mathcal{D} = \{\mathcal{D}_{normal}, \mathcal{D}_{backdoor}\}$ (Line 1). Afterwards, the attacker finds a set of parameters $\theta_{\hat{\omega}_G, \hat{I}_i}$ whose absolute gradient updates in $\hat{\omega}_G$ are the smallest and stores their indexes $\hat{I}_i$ in a candidate set $\hat{I}$ for further refinement (Line 2). In this paper, we use index to denote a parameter's position in the model and $\theta_{\omega,I}$ to denote the parameter in model ω at index I.

Algorithm 2 Finding Redundant Neurons as Indicators

Input: Backdoor models X, Decoy models S, Local dataset $\mathcal{D}$, Global model G_t

Output: Indicator-implanted backdoor and decoy models $X^{'}, S^{'}$, Index set of indicator I_t

1: $\hat{\omega}_G \leftarrow |\frac{\partial \mathcal{L}_{task}(\mathcal{D})}{\partial G_t}|$
2: $\hat{I} = \{\hat{I}_1, \hat{I}_2, ..., \hat{I}_j\} \leftarrow \arg\min_{\hat{I}_i} \theta_{\hat{\omega}_G, \hat{I}_i}$
3: $H_G \leftarrow |\frac{\partial^2 \mathcal{L}_{task}(\mathcal{D})}{\partial G_t^2}|$
4: $I_t = \{I_1, I_2, ..., I_{m+k}\} \leftarrow \arg\min_{I_i \in \hat{I}} \theta_{H_G, I_i}$
5: **for** each model $\omega_i \in X \cup S$ **do**
6: $\quad \theta^{'}_{\omega_i - G_t, I_i} \leftarrow \kappa \cdot \theta_{\omega_i - G_t, I_i}$
7: **return** $X^{'}, S^{'}, I_t$

The refinement is to guarantee condition **C2** that the parameters in these neurons are not critical for the tasks. To this end, the algorithm adopts *curvature*, the second derivative of cross-entropy loss $\mathcal{L}_{task}$. The main idea is that a parameter with a curvature close to zero is more durable for minor changes without deteriorating the loss, because the curvature measures how relevant each neuron parameter is to the performance of G_t [18], [49]. Besides, as Zhou *et al.* [57] point out, the loss function's Hessian implies the changing direction of gradient updates. Thus, in Line 3, the algorithm again utilizes the local datasets $\mathcal{D}$ to compute the second derivative of the loss function, noted as H_G, by G_t and $\mathcal{D}$. Then it selects an index set I of parameters θ_{H_G,I_i} with the minimum curvature in H_G from candidate set $\hat{I}$ (Line 4).

To implant the indicators into backdoor and decoy models, the attacker assigns each model $\omega_i \in X \cup S$ with an indicator index I_i. In Line 6, $\theta_{\omega_i - G_t, I_i}$ denotes the parameter of ω_i's gradient update at index I_i. The attacker then multiplies $\theta_{\omega_i - G_t, I_i}$ with κ, where κ is a scaling factor for the parameter update, and the resulting $\theta'_{\omega_i - G_t, I_i}$ is recorded to read the indicator feedback from the returned global model. Finally, the algorithm will output those models implanted with indicators X', S' and the index set of indicators I_t for this epoch.

B. Read Indicator Feedback

Algorithm 3 Read Indicator

Input: Current and previous global model G_t, G_{t-1}, Previous index set of indicators I_{t-1}, Previous backdoor and decoy models X'_{t-1}, S'_{t-1}

Output: Acceptance status A

1: **for** $I_i \in I_{t-1} \& \omega_i \in X'_{t-1} \cup S'_{t-1}$ **do**
2: $\quad feedback_i \leftarrow \frac{\theta_{G_t - G_{t-1}, I_i}}{\theta_{\omega_i - G_{t-1}, I_i}}$
3: $\quad$ **if** $|feedback_i| > 1$ **then**
4: $\quad\quad$ early stop and return
5: $\quad$ **if** $feedback_i \leq \frac{1}{\kappa}$ **then** $A_i \leftarrow Rejected$
6: $\quad$ **else if** $feedback_i < \frac{max(feedback)}{2}$ **then**
7: $\quad\quad A_i \leftarrow Clipped$
8: $\quad$ **else** $A_i \leftarrow Accepted$
9: **return** A

Algorithm 3 describes the procedures to retrieve an indicator's feedback on whether a malicious model ω_i was accepted in the previous $(t-1)$-th epoch through the global model update $G_t - G_{t-1}$, the index set of preceding-epoch indicator I_{t-1}, and the recorded indicator value $\theta_{\omega_i - G_{t-1}, I_{t-1}}$. Specifically, let $feedback_i$ denote the parameter change of global model at index I_i over that of the attacker's model at the same index (Line 2). The algorithm then determines the acceptance status A_i (i.e., "Accepted", "Clipped"[1], or "Rejected") based on $feedback_i$ for each model. If the corresponding $feedback_i$ is less than $\frac{1}{\kappa}$, the attacker will label the model ω_i with an acceptance status of "Rejected" (Line 5). Theorem 1 and the proof demonstrate its correctness. For those $feedback_i > \frac{1}{\kappa}$, the model ω_i will be labeled with an acceptance status of "Clipped' if its $feedback_i$ is significantly less than the maximum $feedback$ (e.g., less than half of the maximum $feedback$), or "Accepted" otherwise (Lines 6-8).

Theorem 1: The attacker's model $\omega_{adv,t-1}$ is "Rejected" if its corresponding $feedback \leq \frac{1}{\kappa}$.

Proof of Theorem 1: The actual change of a global model parameter corresponding to an indicator during epoch $t-1$, noted as $\theta_{G_t - G_{t-1}, I}$, is mainly caused by the weighted average of the indicator value of the attacker's model $\omega_{adv,t-1}$ and the values of other models $\{W_{1,t-1}, ..., W_{n-1,t-1}\}$.

$$\theta_{G_t - G_{t-1}, I} = \frac{\eta}{n}(\kappa \cdot \theta_{\omega_{adv,t-1} - G_{t-1}, I} + \sum_{1}^{n-1} \theta_{W_{i,t-1} - G_{t-1}, I}) \tag{6}$$

If the attacker-submitted model $\omega_{adv,t-1}$ has been discarded by the central aggregator, then the contribution of this parameter change will only come from the other $n-1$ models. Therefore, $feedback$, which is the ratio of the change on global model to the change made by the attacker, becomes:

$$feedback = \frac{\theta_{G_t - G_{t-1}, I}}{\theta'_{\omega_{adv,t-1} - G_{t-1}, I}} = \frac{\frac{\eta}{n}\sum_{1}^{n-1} \theta_{W_{i,t-1} - G_{t-1}, I}}{\kappa \cdot \theta_{\omega_{adv,t-1} - G_{t-1}, I}} = \frac{\eta \cdot \Delta\bar{\theta}_{W_{t-1}, I}}{\kappa \cdot \theta_{\omega_{adv,t-1} - G_{t-1}, I}} \tag{7}$$

The $\Delta\bar{\theta}_{W_{t-1}, I}$ represents the average update on this parameter by other $n-1$ clients. We assume that usually the global learning rate $\eta \leq 1$ and parameter changes on redundant neurons between $\omega_{adv,t-1}$ and $\{W_{1,t-1}, ..., W_{n-1,t-1}\}$ are close, i.e., $\Delta\bar{\theta}_{W_{t-1}, I} \approx \theta_{\omega_{adv,t-1} - G_{t-1}, I}$, then,

$$\frac{\eta \cdot \Delta\bar{\theta}_{W_{t-1}, I}}{\kappa \cdot \theta_{\omega_{adv,t-1} - G_{t-1}, I}} \approx \frac{\eta}{\kappa} \leq \frac{1}{\kappa} \tag{8}$$

■

Figure 3 illustrates the linear relationship between the average $feedback$ of 20 backdoor models in the same epoch and the global learning rate η for CIFAR10 dataset and ResNet18 model. The solid line shows that $feedback$ can reflect the backdoor model's amplitude of being clipped by the central aggregator. Furthermore, the dash line denotes the condition for $feedback = \frac{1}{\kappa}$ and its intersection with the solid line is close to the origin, illustrating that $\frac{1}{\kappa}$ is a reliable criterion to decide whether the model is accepted or not.

It is worth noting that the indicator is inapplicable against defense algorithms that apply weak DP methods, such as FLAME. By adding Gaussian noise to the global model, weak DP methods make each parameter in the model perturb randomly [29], [31], [36], [46]. This dramatically reduces the accuracy of the indicator's feedback because redundant neurons are no longer modified by the attacker only. In Line 3 of Algorithm 3, if any $feedback_i$ is greater than 1, the algorithm will stop reading the indicator, disable the adaptive

[1]In this paper, we reserve "Clipped" for future use and now 3DFed treats it the same as "Rejected" in the following Adaptive Tuning parts of Section VII-B and VIII-A.

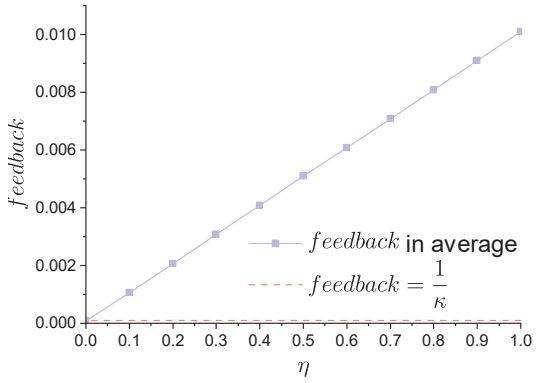


Fig. 3: **Feedback to different η for CIFAR10 and ResNet18**

tuning, and resume the attack with fixed hyper-parameters. This is because as FedAvg dilutes the contribution of each model with $\frac{\eta}{n}$, and $\frac{\eta}{n} < 1$. As such, when any $feedback_i > 1$, there must be an external force other than the backdoor models modifying the redundant neurons.

However, in the defender's perspective, using DP to mitigate indicator is challenging because DP noise will degrade the main task accuracy [16], [31], [34], [36], [42]. As such, if future defenses aim to exploit DP noise on redundant neurons to mitigate the effect of indicators, there are just too many neurons requiring perturbations: over 9 million redundant neuron parameters whose update values $< 10^{-9}$ for a ResNet18 model trained on CIFAR10 for 200 epochs (Table II).

Magnitude	0	$[10^{-12}, 10^{-9})$	$[10^{-9}, 10^{-5})$	$[10^{-5}, 10^{-3}]$
Count	2.18M	6.94M	1.96M	81.2K

TABLE II: **Distribution of parameter updates from a ResNet18 model trained on CIFAR10 for 200 epochs.**

Besides, one may doubt that $\hat{\omega}_G$ and H_G might be unreliable for approximation when other clients' local datasets are highly non-Independent and Identically Distributed (non-IID). As the higher the non-IID degree, the more diverse other models are [22], the selected redundant neurons might be esstential for other benign models. In fact, the chance of indicators being significantly modified by other models is negligible because the indicator's selection range is wide. Furthermore, even if some indicators are affected, since adaptive tuning makes accept/reject decisions on a group level (e.g., Algorithm 4 Line 3), individual failure can be tolerated.

VI. Backdoor Training against Norm Clipping

Although various FL backdoor attacks have been proposed, such attacks require scaled gradients (as in Equation 4) to counteract the benign gradients from other users [2], [4], [47] and are fragile when the central aggregator performs norm clipping [31], [33], [36]. In Section IX, we experimentally demonstrate that an aggregator who applies norm clipping with a dynamic clipping bound for every gradient update can significantly mitigate state-of-the-art backdoor attacks that rely on gradient scaling, whether or not the attacks are single-shot or multi-shot.

To illustrate the rationale behind norm clipping against backdoor attacks, we visualize the effect of scaled backdoor gradients on a neural network model in a complex parameter space (Figures 4-6). Figures 4 and 5 illustrate the change of the global model's parameters by the backdoor attack at epoch t without and with norm clipping. In Figure 4, through gradient scaling, the attacker successfully guides the global model from the original optimum to another local optimum. Such attack can sustain benign users' gradient updates in subsequent epochs, which can only help the model to converge at the current sub-optimum. As such, after the attack the model achieves sub-optimum for the main task and the optimum for the backdoor task.

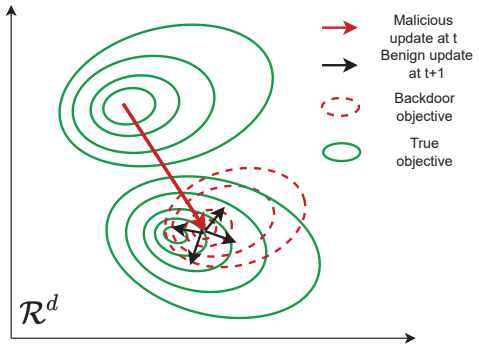


Fig. 4: **Backdoor Attack Without Norm Clipping**

However, if the aggregator performs norm clipping for each gradient update as in Figure 5, the parameters of the global model will no longer be replaced by the scaled gradient within one epoch. Instead, the attacker can only make the global model halfway between these two optimums. As long as it is still closer to the original optimum than the new local optimum, gradient updates of subsequent epochs will pull the global model back to the original optimum, which significantly offsets the impact of the backdoor attack in the previous epoch.

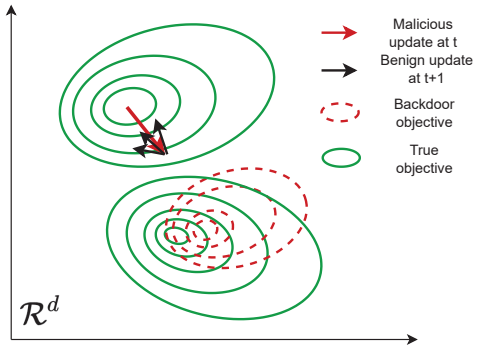


Fig. 5: **Backdoor Attack With Norm Clipping**

Therefore, we propose to add a constraining term to the loss function when training a backdoor model (Equation 9). Proportional to the Euclidean distance between the backdoor ω_{adv} and the global model G_t, this term encourages the optimizer to find a backdoor objective not faraway from G_t in the $l2$-norm as in Figure 6 and thus easier to achieve for the attacker.

$$\mathcal{L} = (1-\beta)\cdot\mathcal{L}_{task} + \beta\cdot||\omega_{adv} - G_t||_2, \tag{9}$$

where β is the weight of this constraining term.

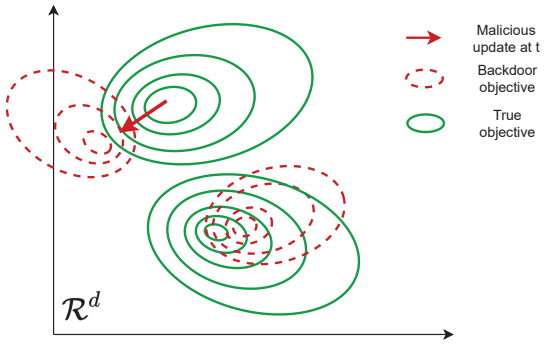


Fig. 6: **Our Attack against Norm Clipping**

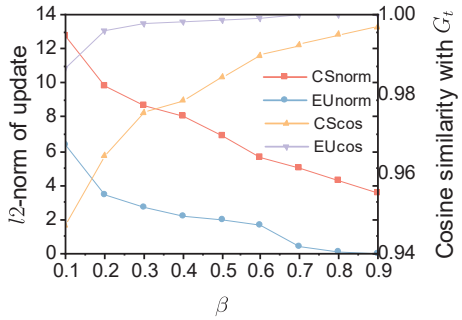


Fig. 7: **Comparison between EU Constraint and CS Constraint**

According to [2], the cosine similarity (CS) between the backdoor model and G_t can also serve as the constraining term. However, we argue that the constraining term of Euclidean distance (EU) can achieve a more covert backdoor model, which is supported by the comparison in Figure 7. We can observe that under various β, EU constraint (EUnorm) generates updates with lower $l2$-norm than CS constraint (CSnorm), and EU constraint (EUcos) also generates updates with less cosine similarity than CS constraint (CScos).

As a final note, in the literature, Projected Gradient Descent (PGD) [36] constrains the $l2$-norm in a similar manner. However, we argue that it cannot be applied in practical backdoor training against norm clipping because PGD requires the exact projection radius, and thus both the clipping bound and $\frac{\eta}{n}$ must be known [40]. Even with the help of our proposed indicator mechanism, obtaining the clipping bound has two challenges. First, it might be dynamic. For instance, in FLAME the clipping bound is the median of the $l2$-norm among all gradient updates in this epoch [31]. As such, an estimated clipping bound in the previous epoch does not help to predict the projection radius in the current epoch. Second, if the aggregator enforces a very strict clipping bound, PGD might fail to train a backdoor model in such extreme conditions.

VII. Noise Masks

The intuition of noise mask is to add each backdoor model's parameters some noise so that these models no longer exhibit those features that can be detected by the central aggregator. Furthermore, the sum of these noise is zero, so if the aggregator accepts all these noise-masked models to the global model, these noise will cancel each other and these models can restore the original poisoning effect. Noise mask has a unique advantage that it is compatible with other evasion methods because it does not modify the training process of the backdoor model.

In this paper, we aim to hide two high-profile features in backdoor models that are commonly detected — over-concentrated neuron updates and high pairwise cosine similarities. For over-concentrated neuron updates, since there is a large proportion of backdoor data mislabeled as the target class in the attacker's training dataset, the backdoor gradient update tends to over-concentrate on a few neurons related to the target class [33]. We adopt the degree of each neuron's update, denoted by Update Energies (UPs) as in Deepsight [33]. Noise masks can increase the UPs, making a backdoor model appear to be trained on a diverse dataset. Besides, high pairwise cosine similarities in backdoor models are mainly because gradient updates submitted by the attacker all have the same malicious objective [9], [41], while gradient updates of benign users with non-IID data do not have a unique objective. Foolsgold takes advantage of this by imposing a learning rate penalty on a batch of gradient updates whose pairwise cosine similarity is the highest among all updates in epoch t [9]. Adding noise masks will significantly decrease the pairwise cosine similarities of backdoor models, making the gradient distribution of backdoor models similar to those of benign models.

A. Noise Mask Optimization

We treat noise mask as a conditional multi-variate optimization problem, and adopt SGD to optimize it. First, to increase the UPs of neurons, we set the first term of the loss function, denoted by $\mathcal{L}_{UPs}(X_i)$, to the sum of the reciprocal of neurons' UPs, where X_i is the noise-masked backdoor model by adding noise mask Δ_i to the original backdoor model ω_{adv} (Equation 10). The reason for reciprocal is that the optimizer will slow down some neurons' updates when they already have high UPs.

In each layer, the attacker randomly selects some neurons with low UPs to add noise, while retaining the other neurons. The main reason is to guard against future adaptive defense by the aggregator who might identify noise-masked models as malicious models. The indexes for those randomly sampled neurons will be recorded as I_{nm}. Then the attacker obtains the UPs of those neurons by a function $UPs(\theta_{X_i,I_j})$, which is the sum of absolute weight and bias in the neuron θ_{X_i,I_j} in model X_i at I_j ($I_j \in I_{nm}$). After that, the attacker builds the first term of the loss function as follows.

$$\mathcal{L}_{UPs}(X_i) = \frac{1}{UPs(\theta_{X_i,I_1})} + \frac{1}{UPs(\theta_{X_i,I_2})} ... + \frac{1}{UPs(\theta_{X_i,I_j})} \tag{10}$$

Furthermore, to prevent a very large $l2$-norm on noise masks, we add $\mathcal{L}_{norm}(\Delta_i) = ||\Delta_i||_2$ as the second term in the loss function. Besides, since the sum of all noise masks should have a zero $l2$-norm, we adopt Lagrange Relaxation [7] to relax the constraint to the unconstrained optimization

algorithm and add $\mathcal{L}_{constrain} = ||\sum_1^m \Delta_i||_2$ as the third term of the loss function, which is the $l2$-norm of the sum for all noise masks. To sum up, the complete loss function for noise mask optimization is:

$$\mathcal{L} = \alpha \cdot \mathcal{L}_{UPs} + (1-\alpha) \cdot \mathcal{L}_{norm} + \lambda \cdot \mathcal{L}_{constrain}, \quad (11)$$

where α and λ are two hyperparameters to control the impact of the first and third terms. Section VII-B will introduce an adaptive tuning mechanism to dynamically adjust α. In Lagrange Relaxation, λ can be gradually increased by Dual Ascent based on $||\sum_1^n \Delta_i||_2$, whose increment step is controlled by ϵ in each Dual Ascent:

$$\lambda^{'} = \epsilon \cdot \mathcal{L}_{constrain} + \lambda \quad (12)$$

Lines 11-18 in Algorithm 4 show the steps to solve the noise mask optimization problem. In the optimization, each noise mask obtains an individual optimizer, and all optimizers share the same loss function for gradient update. After the optimization, each noise mask will be added to the backdoor model and the algorithm will output the noise-masked backdoor model set X.

Algorithm 4 Noise Mask Optimization

Input: Original backdoor model ω_{adv}, Acceptance status A, Previous-epoch parameters α_{t-1}
Output: Noise-masked backdoor models X

1: get acceptance feedback from A
2: **for** each group $N_{adv,i} \in \hat{N}_{adv}$ **do**
3: **if** $N_{adv,i}$ is accepted at $t-1$ **then**
4: append $\alpha_{i,t-1}$ to $\hat{\alpha}$
5: sort $\hat{\alpha}$ with ascending order
6: **for** each group $N_{adv,i} \in \hat{N}_{adv}$ **do**
7: **if** length of $\hat{\alpha} > 1$ **then**
8: sample $\alpha_{i,t}$ from [$\hat{\alpha}$[0],$\hat{\alpha}$[1]]
9: **else if** length of $\hat{\alpha} > 0$ **then**
10: sample $\alpha_{i,t}$ from [$\hat{\alpha}$[0],$\hat{\alpha}$[0]+0.1]
11: initialize noise masks $\hat{\Delta} = \{\Delta_1, \Delta_2, ..., \Delta_m\}$ and λ
12: **for** each group $N_{adv,i} \in \hat{N}_{adv}$ in each local epoch **do**
13: $\mathcal{L}_{UPs} \leftarrow \mathcal{L}_{UPs}(\Delta_1 + \omega_{adv}) + \mathcal{L}_{UPs}(\Delta_2 + \omega_{adv}) + ... + \mathcal{L}_{UPs}(\Delta_m + \omega_{adv})$
14: $\mathcal{L}_{norm} \leftarrow ||\Delta_1||_2 + ||\Delta_2||_2 + ... + ||\Delta_m||_2$
15: $\mathcal{L}_{constrain} \leftarrow ||\sum_1^m \Delta_i||_2$
16: $\mathcal{L} = \alpha_{i,t} \cdot \mathcal{L}_{UPs} + (1-\alpha_{i,t}) \cdot \mathcal{L}_{norm} + \lambda \cdot \mathcal{L}_{constrain}$
17: update all the noise masks Δ_i by SGD for $\mathcal{L}$
18: $\lambda^{'} \leftarrow \epsilon \cdot \mathcal{L}_{constrain} + \lambda$
19: **for** $\Delta_i \in \hat{\Delta}$ **do**
20: $X_i \leftarrow \Delta_i + \omega_{adv}$
21: **return** $X = \{X_1, X_2, ..., X_m\}$

B. Adaptive Tuning For Noise Masks

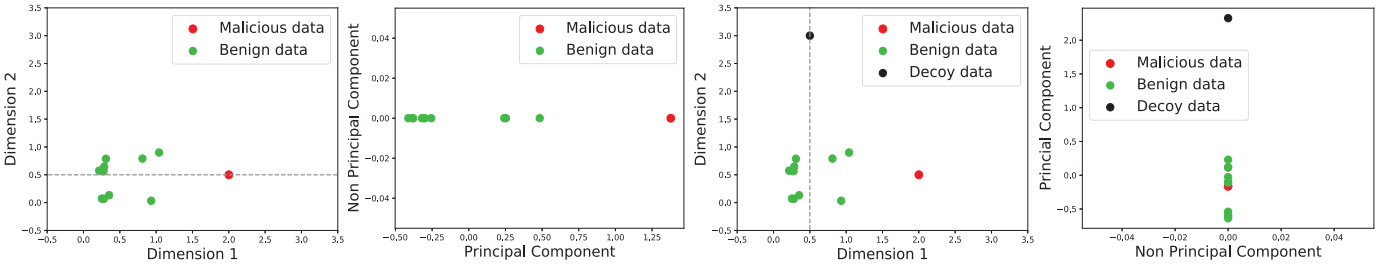
Lines 2-10 in Algorithm 4 shows the adaptive tuning mechanism to determine α. The idea is for the attacker to divide those malicious participants he controls into different groups $\hat{N}_{adv}$, and each group $N_{adv,i}$ carries out backdoor attacks with a different α. In the next epoch, the attacker can know the acceptance of each group's models through indicators and then decide which range of α can be accepted by the central aggregator.

At the beginning, α_i of each group is randomly sampled from the interval $[0, 1]$. Then in each epoch t, the attacker adds those accepted groups' $\alpha_{i,t-1}$ in the previous epoch $t-1$ into a candidate set $\hat{\alpha}$. Then $\alpha_{i,t}$ of each group in this epoch is sampled from the interval having the lower bound and upper bound as the first and second minimum in $\hat{\alpha}$ (Line 8). Since there are potential conflicts between different modules (e.g., a strong noise might increase the $l2$-norm of backdoor updates, thus conflicting with the constraining loss of Euclidean distance), our adaptive tuning minimizes possible conflicts among them by always selecting the minimum applicable $\alpha_{i,t-1}$ as the lower bound. If only one group is accepted in $t-1$, we set the accepted $\alpha_{i,t-1}$ as the lower bound and the upper bound of the interval to be slightly (e.g., 0.1) larger than its lower bound (Line 10). Otherwise, if all groups' models are rejected in epoch $t-1$, the attacker will increase each $\alpha_{t,i}$ so that α can get closer to the acceptable range.

VIII. Decoy Models

There is a new direction in FL defense to adopt dimensionality reduction techniques, such as Principal Component Analysis (PCA), and project gradient updates in every epoch to a low-dimensional space to separate malicious models from benign ones [14], [23], [38], [41]. However, we argue that such defense does not guarantee the projected low-dimensional space to be the determinant dimension to distinguish attackers [35]. Instead, it only guarantees that specific data properties (e.g., variance) in those principal components are maximized. Furthermore, the attacker can *fool the dimension reduction algorithm to select garbage dimensions where backdoor models are not separated from benign ones*. A toy example in Figure 8 illustrates how the result of PCA can be easily manipulated to garbage dimensions. First, we project a set of two-dimensional data points to one-dimensional space through PCA (Figure 8 (a)), where malicious points are located to the right of the benign points. As such, PCA will select the horizontal axis as the principal component as it maximizes the data variance (Figure 8 (b)). So the horizontal axis is the determinant dimension. Now suppose the attacker generates another decoy point at the top of the benign points with a much larger distance (Figure 8 (c)). As a result, PCA will select the vertical axis as the principal component since the data variance on the vertical axis becomes more significant than that on the horizontal axis. As such, this vertical axis becomes a garbage dimension because the true malicious points are still hidden in the benign points after projection to this dimension (Figure 8 (d)).

Generalizing the above example to FL, in this section we investigate how the attacker can upload additional decoy models to the central aggregator, each of which manipulates one principal component towards a garbage dimension. A unique challenge is the number of decoy models to upload. If the



(a) (b) (c) (d)

Fig. 8: **A toy example of manipulating PCA.** (a): drawing of two-dimensional malicious and benign data; (b): PCA results of (a), with (a)'s horizontal axis as the principal component; (c): drawing of two-dimensional malicious, benign and decoy data; (d): PCA results of (c), with (c)'s vertical axis as the principal component.

central aggregator applies PCA with n principal components, the attacker must upload no fewer than n decoy models to successfully fool the PCA. Otherwise, if the attacker only uploads k ($k < n$) models, although the first k dimensions in the PCA result will be garbage dimensions, the remaining $n-k$ dimensions may still be determinant dimensions. As such, in the next subsection we exploit the indicator to speculate on n and adaptively determine the number of decoy models k.

A. Adaptive Tuning For Decoy Models

Let k_t denote the number of decoy models for the t-th epoch. Lines 2-8 in Algorithm 5 show the adaptive tuning of k_t. Specifically, if k_{t-1} decoy models are insufficient at $(t-1)$-th epoch, all the decoy and backdoor models would be discarded, so all acceptance status A from reading indicators in Section V-B are "Rejected" (Lines 2 and 4). Therefore, in this t-th epoch, the attacker will adaptively increase k_t by 1 (Line 5). On the other hand, if $k_{t-1} > n$ or there is no dimension reduction at the central aggregator, some excessive decoy models are accepted into the global model in $(t-1)$-th epoch. Our adaptive tuning will reduce k_t by the number of decoy models accepted in the previous epoch (Line 7). For the same reason behind Alrogithm 4 Line 8, adaptive tuning will alleviate decoy model's potential conflicts with other parts by choosing the smallest k_t and will skip optimizing this module if $k_t = 0$.

Furthermore, we also devise an "exit plan" for this decoy model module. In case these decoy models would be detected by the aggregator in the future, the attacker will fail to fool the dimension reduction. Therefore, in Line 4 we set a timeout to prevent from endless increment of k_t. It is based on the Cumulative Variance Contribution Rate (CVCR), which is the proportion of data variance on the principal components to the variance in the original data in PCA [54]. If the CVCR on the first n principal components is already huge (e.g., > 0.99 as in Line 4), it will be unnecessary for the central aggregator to use the $(n+1)$-th principal component. Therefore, the attacker can disable the decoy model module for the above situation. CVCR can be approximated by the attacker (Line 3) using the existing k_{t-1} decoy models S_{t-1}, the backdoor models X, and a benign reference model ω_b trained at the initial stage (Line 1).

Algorithm 5 Decoy Model Design

Input: Local dataset $\mathcal{D}$, Backdoor models X, Acceptance status A, Previous decoy models S_{t-1}, Previous number of decoy models k_{t-1}

Output: Decoy models S

1: train a benign reference model ω_b by local benign datasets $\mathcal{D}_{normal}$
2: count the number of accepted backdoor models and decoy models as $A_{backdoor}$ and A_{decoy}
3: $CVCR^{'} \leftarrow$ approximate CVCR by X, ω_b, S_{t-1}
4: **if** $A_{backdoor}, A_{decoy} = 0$ & $CVCR < 0.99$ **then**
5: $\quad k_t \leftarrow k_{t-1} + 1$
6: **else if** $A_{decoy} > 0$ **then**
7: $\quad k_t \leftarrow k_{t-1} - A_{decoy}$
8: **else** $k_t \leftarrow k_{t-1}$
9: $I_{dec} = \{I_1, I_2, ..., I_k\} \leftarrow \arg\min_{I_i}(\theta_{|\omega_b - \overline{X}|, I_i})$
10: **for** each $S_i \in S$ **do**
11: $\quad L_1 \leftarrow -\theta_{|S_i - \omega_b|, I_i}$
12: $\quad L_2 \leftarrow ReLU(L_{task}(S_i, \mathcal{D}) - L_{task}(\omega_b, \mathcal{D}))$
13: $\quad \mathcal{L} \leftarrow L_1 + L_2$
14: $\quad$ perform SGD on S_i by $\mathcal{L}$
15: **return** $S = \{S_1, S_2, ..., S_k\}$

B. Optimizing Decoy Models

To prevent accepted decoy models from excessively deteriorating the global model's performance, in 3DFed we further apply training to optimize decoy models. Lines 9-14 of Algorithm 5 show the procedures for decoy model optimization. The attacker first searches an index set I_{dec} of parameters $\theta_{|\omega_b - \overline{X}|, I_i}$ where the difference between benign reference model ω_b and the average backdoor model $\overline{X}$ is minimum (Line 9). Each decoy model, denoted as S_i, is then assigned with one index I_i, and the first loss L_1 is the negative of parameter difference $\theta_{|S_i - \omega_b|, I_i}$ between S_i and ω_b at index I_i (Line 11). It will invoke dimension reduction in the central aggregator to select the I_i-th parameter as one of its principal components, which is a garbage dimension. After that, the second loss term L_2 is constructed to constrain the accuracy of S_i. In Line 12, the attacker calculates $L_{task}(S_i, \mathcal{D}) - L_{task}(\omega_b, \mathcal{D})$, the difference of cross-entropy loss between S_i and ω_b using $\mathcal{D}$, and then encloses it in

an *ReLU* function. This will make the constraint of L_2 less important when the main task performance of S_i is better than ω_b. Finally, the attacker perform SGD training on these decoy models in Line 14.

IX. Evaluations

A. Experimental Setup

Our code is open-source[2]. We set up a standard FL task to train a neural network model for image classification with three popular datasets, CIFAR10, MNIST, and Tiny-Imagenet [20], [27]. Following [19], in Tiny-Imagenet, we reshape the training images from 64x64 to 224x224, and the central aggregator starts with a pre-trained model with image size of 64x64. We adopt the same FL settings in [2] for CIFAR10 and those in [47] for MNIST and Tiny-Imagenet. Table III summarizes these system settings. For CIFAR10 and Tiny-Imagenet, Resnet18 [12] is used as the model, and a simple 2-conv-2-fully-connected model is used for MNIST dataset. To simulate the non-IID dataset, we sample each participant's local dataset with Dirichlet distribution [30].

There are 100 FL participants for CIFAR10 and Tiny-Imagenet, and 20 for MNIST. They all participate in all epochs in terms of local training and gradient updates. After receiving gradient updates, the central aggregator applies FedAvg with or without backdoor defense to aggregate those updates. We implement five state-of-the-art FL backdoor defense algorithms introduced in Section II-B, namely, Deepsight [33], Foolsgold [9], FLAME [31], FL-Detector [51], and RFLBAT [41].[3]

B. Adversary's Setting

We assume that the attacker compromises 20% of the devices and performs the pixel-pattern backdoor attack for 20 epochs. We assume the attacker will construct the backdoor dataset by adding a pixel pattern to the corner of normal pictures and mislabeling them as the 8th target class. The attacker holds 20 compromised devices[4] for CIFAR10 and Tiny-Imagenet, and four for MNIST.

We consider the *Model Replacement Attack* [2] as our baseline with the same attack setting as 3DFed. In 3DFed, the SGD setting for training the backdoor models, noise masks and decoy models are same as the baseline. For κ in the indicator, we set it as a constant 10^5 for all the experiments.

The experiment is in the black-box: the attacker cannot know any external information beyond his compromised devices and is agnostic to the central aggregator's aggregation rule and other benign participants' datasets. Furthermore, we assume that the central aggregator will not block some devices detected as potential attackers so that the attacker can participate in each epoch of FL and adaptively adjust its attack strategy.

C. Experimental Results

Figures 9, 10 and 13 show the experimental results for CIFAR10, Tiny-Imagenet and MNIST respectively without and with the defense (Figure 13 is in the Appendix). When there is no defense in the central aggregator, the baseline attack outperforms 3DFed in terms of efficiency and backdoor accuracy. However, the baseline never succeeds once the central aggregator applies any single defense method. On the contrary, 3DFed can still successfully poison the global model to exhibit backdoor behavior against all backdoor defense algorithms within similar number of epochs as the no-defense case.

The result details are as follows. In Deepsight (Figure 10 (a), (f) and Figure 13 (a)), since the noise mask increases the UPs of neurons, our model's NEUPs are significantly higher than those of backdoor models without noise masks, so Deepsight cannot detect our backdoor models. Another effect of noise mask is alleviating the dense distribution among backdoor models, which lowers the learning rate punishment to our backdoor models by Foolsgold (Figure 10 (b), (g) and Figure 13 (b)). It should be noted that Foolsgold calculates the similarities based on each client's accumulated historical gradient updates, so when the attacker continuously performs poisoning attacks using the same set of compromised devices, their similarities of historical updates increase slightly. Our adaptive tuning is aware of this and adaptively adjusts α to improve the intensity of noise masks. For FLAME and FL-Detector, our constraining loss on Euclidean distance generates backdoor updates with low $l2$-norm, thereby mitigating the inhibition from FLAME (Figure 10 (c), (h) and Figure 13 (c)) and getting closer to the predicted model of FL-Detector (Figure 10 (d), (i) and Figure 13 (d)). Even though the DP noise in FLAME mitigates 3DFed's indicator, evading FLAME does not require the attacker to apply adaptive tuning on noise mask and decoy model. For FL-Detector in CIFAR10 and Tiny-Imagenet (Figure 10 (d) and (i)), although the baseline can poison the global model at the beginning, its malicious score continuously increases and is detected as outliers at 10-th epochs after the attack. Since FL-Detector will restart the training by excluding all the attacker's devices after the restart, the backdoor accuracy of the global model in baseline restores to a low level and no longer increases. For FL-Detector in MNIST (Figure 13 (d)), the baseline attack is immediately detected at the start so that its backdoor accuracy never increases. For RFLBAT (Figure 10 (e), (j) and Figure 13 (e)), our attack adaptively increases the number of decoy models k_t until k_t equals to the number of principal components in the central aggregator. As such, our decoy models successfully distract RFLBAT and make the global model accept the backdoor models.

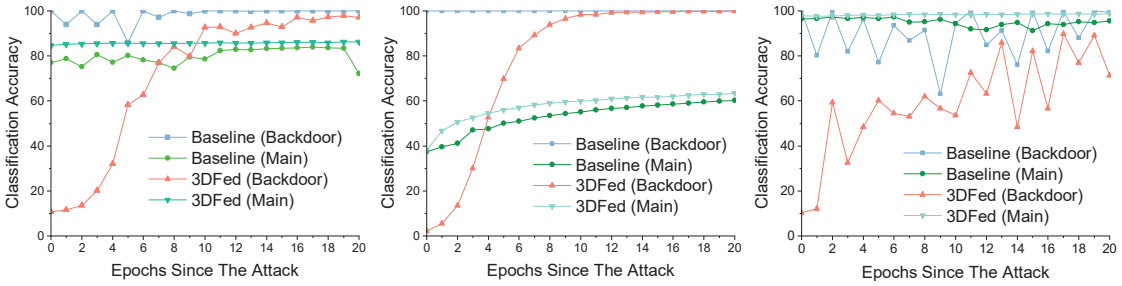
[2]https://github.com/haoyangliASTAPLE/3DFed

[3]Since RFLBAT does not provide a method to determine the number of clusters required by its KMeans clustering, we adapt the Gap Statistics [37] in FL-Detector to RFLBAT for unsupervised learning on the number of clusters in the PCA results.

[4]For a good effect of equally partitioning the devices into multiple groups and estimating the hyper-parameters in adaptive tuning, we set compromised devices to 20 by default. In Figure 11 (a) of Ablation Study, we will show that 3DFed still works well even when this number is 4. Furthermore, 20 compromised devices would represent a much smaller proportion of devices overall in real FL systems with thousands of devices.

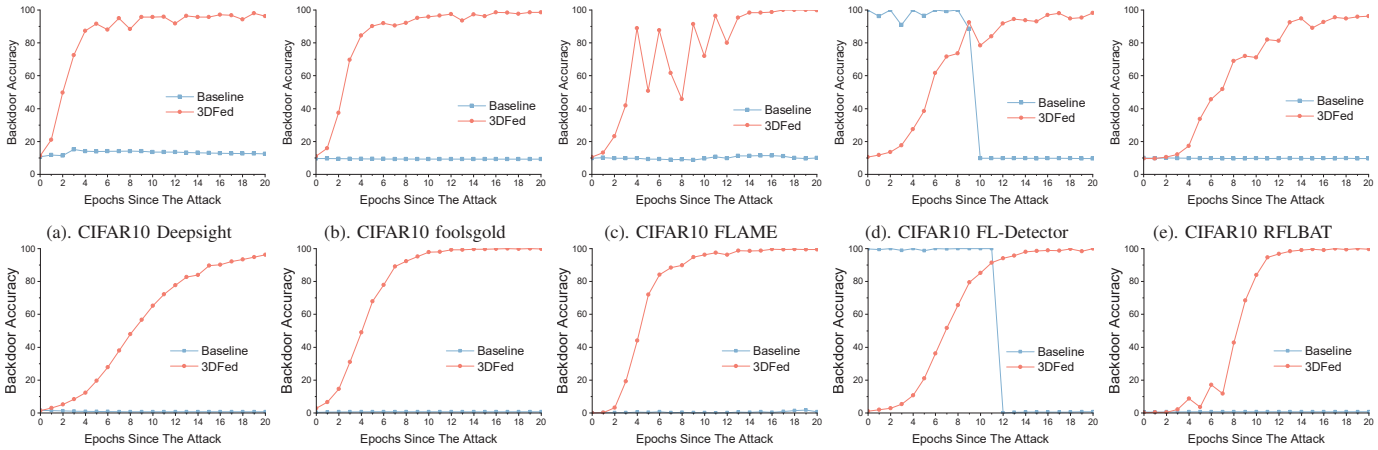
	Size (Train/Test)	# of features	# of classes	Model	Learning Rate
CIFAR10	50k/10k	32×32	10	ResNet18	0.1
MNIST	60k/10k	28×28	10	2 conv and 2 fc	0.1
Tiny-Imagenet	100k/10k	224×224	200	ResNet18	0.001

TABLE III: **Dataset and FL Settings**



(a). CIFAR10 (b). Tiny-Imagenet (c). MNIST

Fig. 9: **Results for CIFAR10, Tiny-Imagenet and MNIST Without Defense**



(a). CIFAR10 Deepsight (b). CIFAR10 foolsgold (c). CIFAR10 FLAME (d). CIFAR10 FL-Detector (e). CIFAR10 RFLBAT

(f). Tiny-Imagenet Deepsight (g). Tiny-Imagenet foolsgold (h). Tiny-Imagenet FLAME (i). Tiny-Imagenet FL-Detector (j). Tiny-Imagenet RFLBAT

Fig. 10: **Results for CIFAR10 and Tiny-Imagenet against Five Backdoor Defenses**

There are two side observations from the results. First, compared to other defenses, FLAME, which applies HDBSCAN to gradient updates, has high false positive rate [33]. In our experiment, even without any attackers, HDBSCAN in FLAME still considers 48.9% of benign updates as outliers when datasets are sampled with Dirichlet distribution of hyper-parameter 0.9. It is because in each epoch, there are always some benign gradient updates with higher density than other benign gradient updates. According to [5], the density-based HDBSCAN treats the highest density one as the cluster (whose minimum size is the half of the total participants) and others as outliers. This shows that a density-based clustering method like HDBSCAN might not be suitable for FL systems where gradient updates are unevenly distributed.

Another side observation is that the baseline's main task accuracy is always lower than 3DFed in the no-defense case. For example, in Figure 9 (b), the baseline's main task accuracy for Tiny-Imagenet at the end of the attack is 3.25% lower than that of 3DFed. We attribute this to the inherent limitation of model replacement attacks. FL essentially obtains a consensus solution for all the participants with diverse data distributions. Due to such non-IID conditions, every local model trained on the local dataset is sub-optimum for the global main task [32]. Therefore, an attacker who replaces the global model with his local models inevitably deteriorate the global model from the global optimum to a sub-optimum. As a result, the model replacement attack leads to a slightly lower main task accuracy. On the other hand, in 3DFed, the global model is not replaced in one single epoch. Instead, through multiple epochs of contributions from the attacker and other benign participants, all the participants together find a solution in which the global model remains in the global optimum but exhibits backdoor behavior.

D. Ablation study

In this subsection, we vary the parameters of 3DFed to evaluate their impact on the success of the attack. Due to space limitation, we only show the results of CIFAR10 dataset. The ablation study consists of two parts. First, in Section IX-D1, we alternate some parameters in the attacker's settings and evaluate the attack performance. Second, in Section IX-D2 we disable some modules in 3DFed and restrict the attacker to use only one of the three evasion components in Sections VI,

VII-A, and VIII to evaluate each module's contribution to 3DFed.

1) Varying Attacker's Settings: We evaluate on the following parameters — number of compromised devices, β for backdoor training, η for central aggregator, and starting epoch for attack. Figure 11 plots all the results.

Number of Compromised Devices. Figure 11 (a) illustrates the backdoor accuracy with the number of compromised devices, noted as N_{adv}, varying from 20 to 2 against FLAME. Although decreasing the proportion of compromised devices makes the poisoning less efficient, the global model can still have a high backdoor accuracy at the end of the attack in most cases. For example, even when the number of compromised devices is only 4, the backdoor accuracy of the global model still reaches 73.76% in the end.

β for Backdoor Training. The hyper-parameter β controls the weight of constraining term in Section VI. The smaller the β, the more heavily the backdoor models are clipped. But meanwhile a larger β might have adverse impact on the backdoor performance. Figure 11 (b) plots the backdoor accuracy with backdoor attacks when β decreases from 0.6 to 0.1. When β is 0.6, the weight of the constraining term is too large, making it difficult for the optimizer to find a backdoor objective. When β is 0.1, the backdoor model still has a high Euclidean distance from the global model, making it less effective against norm clipping. The attack performance reaches a decent plateau when β is ranges from 0.5 to 0.1. As such, 3DFed is robust against β.

η for Central Aggregator. The central aggregator's global learning rate η has a similar effect to norm clipping in limiting backdoor attacks. The lower the η, the less impact from the attacker in a single epoch. However, a low η also limits benign gradient updates, taking the global model more epochs to converge. In Figure 11 (c), we vary η from 1 to 0.2. When η is 0.4, the global model still obtains a backdoor behavior with an accuracy of 93.12% in the end. When η is 0.2, our attack fails to achieve successful poisoning, but the global learning rate is also too low to find a model to converge for the main task.

Starting Epoch for Attack. Backdoor attacks are more likely to occur when the global model is close to convergence. In Figure 11 (d), we vary the starting epoch, noted as E_{start}, for attack. We observe that when the attacker starts at epoch 50, the global model has not converged at this time, so it is difficult for the attacker to stabilize the global model on a fixed backdoor objective, which leads to only 73.35% backdoor accuracy at the end of attack. When the attack starts after epoch 100, a stable poisoning effect can be achieved, where the backdoor accuracy is 98.97%. To avoid early launch of backdoor attacks, one can use gradients and curvatures in Section V as natural signs of convergence. If the majority of gradients and curvatures are significantly larger than the rest, the global model is considered far from convergence.

2) Single-Layered Attack: In this subsection, we evaluate the performance of single-layered backdoor attacks, i.e., the attacker adopts one of the three camouflage components in 3DFed, namely, Constraining Loss only (CL-only), Noise Mask only (NM-only), and Decoy Model only (DM-only). The results over epochs are plotted in Figure 12. The central aggregator's defense methods and attacker's settings remain the same as in Section IX-A and Section IX-B.

Table IV summarizes the end-of-attack results, where each entry denotes the backdoor accuracy and the last column is the result of original 3DFed with all three components (The last row "Combined" will be discussed in Section X). We observe that applying single-layered camouflage component can only partially or even hardly evade defenses, especially when the detection scheme matches the camouflage component. For example, in Figure 12 (a), CL-only is successful only when the defense is based on measuring the Euclidean distance, where its backdoor accuracy is 99.94% for FLAME and 99.54% for FL-Detector. Once the defense inspects any other characteristics, CL-only completely fails with 9.58% accuracy for Deepsight, 9.65% accuracy for Foolsgold and 9.49% accuracy for RFLBAT. On the other hand, when incorporating these camouflage components altogether, 3DFed can effectively evade detection of all schemes, where the backdoor accuracy reaches 96.17% for Deepsight, 98.51% for Foolsgold, 96.18% for RFLBAT, 99.89% for FLAME and 98.21% for FL-Detector (Table IV).

	CL-only	**NM-only**	**DM-only**	**3DFed**
Deepsight	9.58%	**85.5%**	10.39%	**96.17%**
Foolsgold	9.65%	9.37%	9.73%	**98.51%**
RFLBAT	9.49%	9.69%	**99.98%**	**96.18%**
FLAME	**99.94%**	57.31%	8.97%	**99.89%**
FL-Detector	**99.54%**	9.24%	9.62%	**98.21%**
Combined	9.72%	9.61%	9.68%	**98.48%**

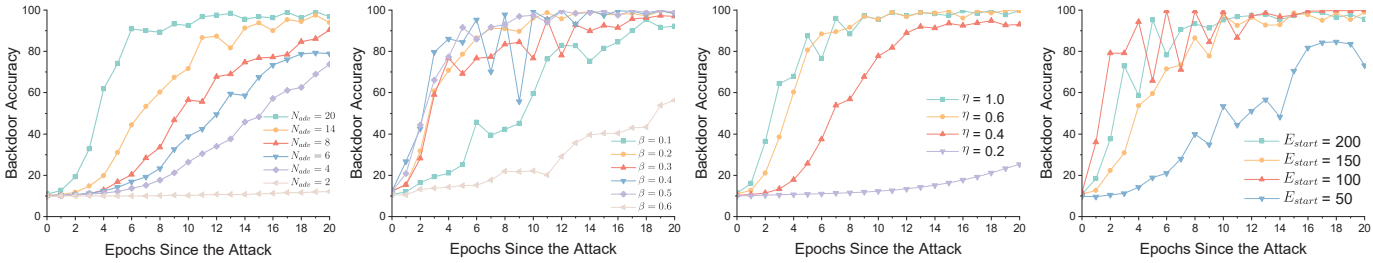
TABLE IV: **Comparison of Different Attack Modes.** Backdoor accuracy above 80% is marked in bold

We also observe that no single-layered attack can evade the detection of Foolsgold. In the NM-only case, backdoor models are imposed with a significant learning rate punishment by Foolsgold even when our adaptive tuning dynamically increases α to 0.9. Although the noise masks can scatter these backdoor models, without constraining the loss function, the gradient updates trained in NM-only have very high $l2$-norm, which cause high pairwise cosine similarities even for those noise-masked backdoor models. This in turn justifies the necessity of a multi-layered attack as 3DFed that combines all these camouflage components.

X. Adaptive Defenses

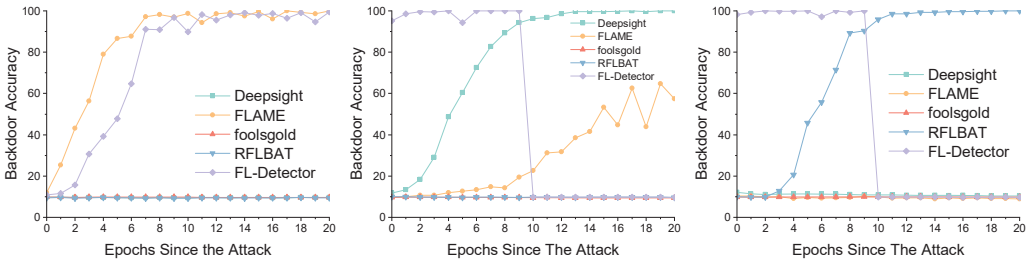
The central aggregator might be aware of 3DFed and take countermeasures adaptively. In this section, we further discuss several potential research directions for adaptive defense against 3DFed and how they may be further circumvented.

Clip the model with a stricter clipping bound. One may apply a stricter clipping bound to further mitigate the effect of constraining loss in Section VI from the 3DFed. However, as pointed out by FLAME [31], this will significantly slow down the convergence. In addition, 3DFed can adaptively



(a). Varying Numbers of Compromised Devices (b). Varying β For Backdoor Training (c). Varying η For Central Aggregator (d). Varying Starting Epoch for Attack

Fig. 11: **Varying Attacker's Settings**



(a). CL-only (b). NM-only (c). DM-only

Fig. 12: **Varying Attacker's Parameters Settings**

change β to have a smaller $l2$-norm, which makes this defense ineffective.

Detect noise-masked models with excessive high-UPs neurons. A future defense might consider models having excessive high-UPs neurons similar to noise-masked backdoor models in Section VII. Nonetheless, this defense may not be effective for two reasons. First, as described in Section VII-A, 3DFed randomly selects the number of neurons to perturb. Second, the attacker can always use her benign model as a reference to avoid extreme numbers.

PCA with a dynamic number of components. Another adaptive defense using PCA might randomly decide the number of components to mitigate 3DFed's decoy model in Section VIII. In addition to the side-effect that inappropriate component numbers may lead to garbage dimensions and degrade clustering accuracy [43], such defense has two issues to address. First, 3DFed can simply decrease the sensitivity of adaptive tuning as a countermeasure. Second, even if decoy models are detected by this dynamic-number PCA, it is still challenging to identify which cluster is malicious due to the reinforcement of other two camouflage modules, constraining loss and noise masks, which hide malicious features in backdoor models.

Add DP noise to redundant neurons to undermine indicators in Section V. 3DFed can apply a larger κ and leverage multiple redundant neurons as the indicator so that their noises can be canceled.

Combine all defenses together. First, there are severe conflicts among the five defenses. Therefore, we can only combine four defenses that conflict less with each other, namely, Deepsight, FL-Detector, Foolsgold, and RFLBAT. In the last row named "Combined" of Table IV, we observe that 3DFed can still achieve 98.48% backdoor accuracy against them.

Defenses using new techniques. 3DFed is designed to address the flaws in backdoor attacks that these defenses can exploit. It will remain working if future defenses, especially those based on outlier detection, depend on the same flaws. In case of new flaws, thanks to its high extensibility, we can still add corresponding new camouflage modules to the 3DFed framework. Nonetheless, there are some promising defenses beyond outlier detections, such as model pruning [8], [39] and federated unlearning [10], [44], [50], that may undermine the effectiveness of 3DFed.

XI. Conclusion

In this paper, we address the two limitations of existing FL backdoor attacks, namely, requiring information beyond black-box setting, and unilateral objectives. The proposed 3DFed framework is adaptive and extensible to launch covert backdoor attacks against in-depth FL defenses. 3DFed is multi-layered and sports three orthogonal evasion modules that camouflage a backdoor model to evade the detection of multi-layered defense. By implanting indicators into a backdoor model, 3DFed obtains attack feedback in a black-box scenario and then leverages adaptive tuning to adjust the backdoor evasion modules dynamically. Through extensive experiments, we demonstrate the stealthiness and robustness of 3DFed against multiple state-of-the-art FL backdoor defenses.

As for future work, we plan to extend 3DFed to support other FL algorithms beyond FedAvg, such as FedAMP [13] and FedBN [25]. We will also incorporate new evasion modules to 3DFed as it is an extensible framework.

ACKNOWLEDGEMENT

This work was supported by the National Natural Science Foundation of China (Grant No: 92270123, 62072390, and 62102334), the National Natural Science Foundation of China for Joint Fund Project (No. U1936218), and the Research Grants Council, Hong Kong SAR, China (Grant No: 15222118, 15218919, 15203120, 15226221, 15225921, 15209922 and C2004-21GF). This work was partially supported by Huawei International. We appreciate anonymous reviewers' constructive comments on the manuscript of this paper.

REFERENCES

[1] Eugene Bagdasaryan and Vitaly Shmatikov. Blind backdoors in deep learning models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1505–1521, 2021.

[2] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*, pages 2938–2948. PMLR, 2020.

[3] Gilad Baruch, Moran Baruch, and Yoav Goldberg. A little is enough: Circumventing defenses for distributed learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[4] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens. In *International Conference on Machine Learning*, pages 634–643. PMLR, 2019.

[5] Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 160–172. Springer, 2013.

[6] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. Local model poisoning attacks to {Byzantine-Robust} federated learning. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 1605–1622, 2020.

[7] Ferdinando Fioretto, Pascal Van Hentenryck, Terrence WK Mak, Cuong Tran, Federico Baldo, and Michele Lombardi. Lagrangian duality for constrained deep learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 118–135. Springer, 2020.

[8] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[9] Clement Fung, Chris J. M. Yoon, and Ivan Beschastnikh. The Limitations of Federated Learning in Sybil Settings. In *Symposium on Research in Attacks, Intrusion, and Defenses*, RAID, 2020.

[10] Xiangshan Gao, Xingjun Ma, Jingyi Wang, Youcheng Sun, Bo Li, Shouling Ji, Peng Cheng, and Jiming Chen. Verifi: Towards verifiable federated unlearning. *CoRR*, abs/2205.12709, 2022.

[11] Xueluan Gong, Yanjiao Chen, Huayang Huang, Yuqing Liao, Shuai Wang, and Qian Wang. Coordinated backdoor attacks against federated learning with model-dependent triggers. *IEEE network*, 36(1):84–90, 2022.

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

[13] Yutao Huang, Lingyang Chu, Zirui Zhou, Lanjun Wang, Jiangchuan Liu, Jian Pei, and Yong Zhang. Personalized cross-silo federated learning on non-iid data. In *AAAI*, pages 7865–7873, 2021.

[14] Hyejun Jeong, Joonyong Hwang, and Tai Myung Chung. Abc-fl: Anomalous and benign client classification in federated learning. *arXiv preprint arXiv:2108.04551*, 2021.

[15] Malhar S Jere, Tyler Farnan, and Farinaz Koushanfar. A taxonomy of attacks on federated learning. *IEEE Security & Privacy*, 19(2):20–28, 2020.

[16] Muah Kim, Onur Günlü, and Rafael F. Schaefer. Federated learning with local differential privacy: Trade-offs between privacy, utility, and communication. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2021, Toronto, ON, Canada, June 6-11, 2021*, pages 2650–2654. IEEE, 2021.

[17] Yein Kim, Huili Chen, and Farinaz Koushanfar. Backdoor defense in federated learning using differential testing and outlier detection. *arXiv preprint arXiv:2202.11196*, 2022.

[18] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[19] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

[20] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[21] Li Li, Yuxi Fan, Mike Tse, and Kuo-Yi Lin. A review of applications in federated learning. *Computers & Industrial Engineering*, 149:106854, 2020.

[22] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-iid data silos: An experimental study. In *38th IEEE International Conference on Data Engineering, ICDE 2022, Kuala Lumpur, Malaysia, May 9-12, 2022*, pages 965–978. IEEE, 2022.

[23] Suyi Li, Yong Cheng, Wei Wang, Yang Liu, and Tianjian Chen. Learning to detect malicious clients for robust federated learning. *arXiv preprint arXiv:2002.00211*, 2020.

[24] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

[25] Xiaoxiao Li, Meirui Jiang, Xiaofei Zhang, Michael Kamp, and Qi Dou. Fedbn: Federated learning on non-iid features via local batch normalization. *arXiv preprint arXiv:2102.07623*, 2021.

[26] Lingjuan Lyu, Han Yu, Xingjun Ma, Lichao Sun, Jun Zhao, Qiang Yang, and Philip S Yu. Privacy and robustness in federated learning: Attacks and defenses. *arXiv preprint arXiv:2012.06337*, 2020.

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

[28] Yuxi Mi, Jihong Guan, and Shuigeng Zhou. Ariba: Towards accurate and robust identification of backdoor attacks in federated learning. *arXiv preprint arXiv:2202.04311*, 2022.

[29] Mohammad Naseri, Jamie Hayes, and Emiliano De Cristofaro. Local and central differential privacy for robustness and privacy in federated learning. *arXiv preprint arXiv:2009.03561*, 2020.

[30] Kai Wang Ng, Guo-Liang Tian, and Man-Lai Tang. Dirichlet and related distributions: Theory, methods and applications. 2011.

[31] Thien Duc Nguyen, Phillip Rieger, Huili Chen, Hossein Yalame, Helen Möllering, Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Shaza Zeitouni, et al. {FLAME}: Taming backdoors in federated learning. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 1415–1432, 2022.

[32] Joaquin Quinonero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. *Dataset shift in machine learning*. Mit Press, 2008.

[33] Phillip Rieger, Thien Duc Nguyen, Markus Miettinen, and Ahmad-Reza Sadeghi. Deepsight: Mitigating backdoor attacks in federated learning through deep model inspection. *arXiv preprint arXiv:2201.00763*, 2022.

[34] W. Ruan, M. Xu, W. Fnag, L. Wang, L. Wang, and W. Han. Private, efficient, and accurate: Protecting models trained by multi-party learning with differential privacy. In *2023 2023 IEEE Symposium on Security and Privacy (SP) (SP)*, pages 76–93, Los Alamitos, CA, USA, may 2023. IEEE Computer Society.

[35] Virat Shejwalkar and Amir Houmansadr. Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning. In *NDSS*, 2021.

[36] Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H Brendan McMahan. Can you really backdoor federated learning? *arXiv preprint arXiv:1911.07963*, 2019.

[37] Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal*

Statistical Society: Series B (Statistical Methodology), 63(2):411–423, 2001.

[38] Vale Tolpegin, Stacey Truex, Mehmet Emre Gursoy, and Ling Liu. Data poisoning attacks against federated learning systems. In *European Symposium on Research in Computer Security*, pages 480–501. Springer, 2020.

[39] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y. Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*, pages 707–723. IEEE, 2019.

[40] Hongyi Wang, Kartik Sreenivasan, Shashank Rajput, Harit Vishwakarma, Saurabh Agarwal, Jy-yong Sohn, Kangwook Lee, and Dimitris Papailiopoulos. Attack of the tails: Yes, you really can backdoor federated learning. *Advances in Neural Information Processing Systems*, 33:16070–16084, 2020.

[41] Yongkang Wang, Dihua Zhai, Yufeng Zhan, and Yuanqing Xia. Rflbat: A robust federated learning algorithm against backdoor attack. *arXiv preprint arXiv:2201.03772*, 2022.

[42] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H. Yang, Farhad Farokhi, Shi Jin, Tony Q. S. Quek, and H. Vincent Poor. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Trans. Inf. Forensics Secur.*, 15:3454–3469, 2020.

[43] Svante Wold. Cross-validatory estimation of the number of components in factor and principal components models. *Technometrics*, 20(4):397–405, 1978.

[44] Chen Wu, Sencun Zhu, and Prasenjit Mitra. Federated unlearning with knowledge distillation. *CoRR*, abs/2201.09441, 2022.

[45] Yaxin Xiao, Qingqing Ye, Haibo Hu, Huadi Zheng, Chengfang Fang, and Jie Shi. Mexmi: Pool-based active model extraction crossover membership inference. In *Advances in Neural Information Processing Systems*.

[46] Chulin Xie, Minghao Chen, Pin-Yu Chen, and Bo Li. Crfl: Certifiably robust federated learning against backdoor attacks. In *International Conference on Machine Learning*, pages 11372–11382. PMLR, 2021.

[47] Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. Dba: Distributed backdoor attacks against federated learning. In *International Conference on Learning Representations*, 2019.

[48] Qiang Yang, Yang Liu, Yong Cheng, Yan Kang, Tianjian Chen, and Han Yu. Federated learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 13(3):1–207, 2019.

[49] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pages 3987–3995. PMLR, 2017.

[50] Lin Zhang, Li Shen, Liang Ding, Dacheng Tao, and Ling-Yu Duan. Fine-tuning global model via data-free knowledge distillation for non-iid federated learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 10164–10173. IEEE, 2022.

[51] Zaixi Zhang, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Fldetector: Defending federated learning against model poisoning attacks via detecting malicious clients. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2545–2555, 2022.

[52] Zhengming Zhang, Ashwinee Panda, Linyue Song, Yaoqing Yang, Michael Mahoney, Prateek Mittal, Ramchandran Kannan, and Joseph Gonzalez. Neurotoxin: Durable backdoors in federated learning. In *International Conference on Machine Learning*, pages 26429–26446. PMLR, 2022.

[53] Chen Zhao, Yu Wen, Shuailou Li, Fucheng Liu, and Dan Meng. Federatedreverse: A detection and defense method against backdoor attacks in federated learning. In *Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security*, pages 51–62, 2021.

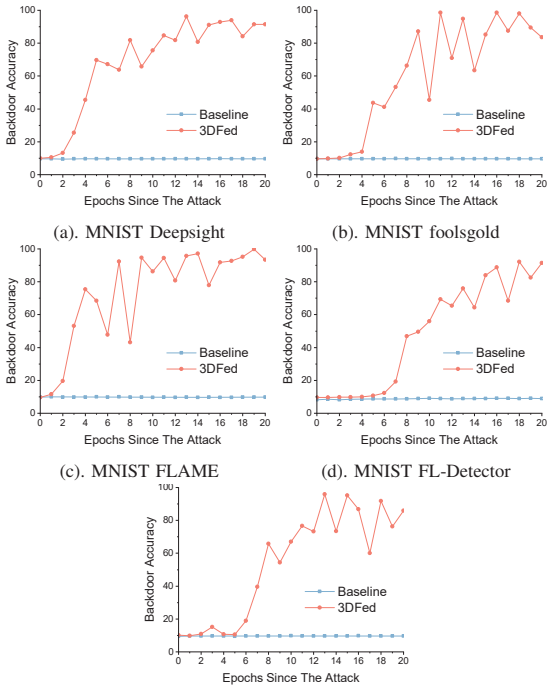
[54] Huimin Zhao, Jianjie Zheng, Junjie Xu, and Wu Deng. Fault diagnosis method based on principal component analysis and broad learning system. *IEEE Access*, 7:99263–99272, 2019.

[55] Huadi Zheng, Haibo Hu, and Ziyang Han. Preserving user privacy for machine learning: local differential privacy or federated machine learning? *IEEE Intelligent Systems*, 35(4):5–14, 2020.

[56] Huadi Zheng, Qingqing Ye, Haibo Hu, Chengfang Fang, and Jie Shi. Protecting decision boundary of machine learning model with differentially private perturbation. *IEEE Trans. Dependable Secur. Comput.*, 19(3):2007–2022, 2022.

[57] Xingchen Zhou, Ming Xu, Yiming Wu, and Ning Zheng. Deep model poisoning attack on federated learning. *Future Internet*, 13(3):73, 2021.

APPENDIX



(a). MNIST Deepsight (b). MNIST foolsgold

(c). MNIST FLAME (d). MNIST FL-Detector

(e). MNIST RFLBAT

Fig. 13: **Results for MNIST against Five Backdoor Defenses**