2023 IEEE International Conference on Cluster Computing (CLUSTER) | 979-8-3503-0792-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/CLUSTER52292.2023.00014

FEDGUARD: Selective Parameter Aggregation for Poisoning Attack Mitigation in Federated Learning

Melvin Chelli*, Cédric Prigent†, René Schubotz*,
Alexandru Costan†, Gabriel Antoniu†, Loïc Cudennec‡, Philipp Slusallek*
* Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), Saarland Informatics Campus, Germany
{melvin.chelli, rene.schubotz, philipp.slusallek}@dfki.de
† University of Rennes, Inria, CNRS, IRISA - Rennes, France
{cedric.prigent, alexandru.costan, gabriel.antoniu}@irisa.fr
‡ DGA Maîtrise de l'Information, Rennes, France
loic.cudennec@intradef.gouv.fr

***Abstract*—Minimizing the attack surface of Federated Learning (FL) systems is a field of active research. FL turns out to be highly vulnerable to various threats coming from the edge of the network. Current approaches rely on robust aggregation, anomaly detection and generative models for defending against poisoning attacks. Yet, they either have limited defensive capabilities due to their underlying design or are impractical to use as they rely on constraining building blocks.**

We introduce FEDGUARD, a novel FL framework that utilizes the generative capabilities of Conditional Variational AutoEncoders (CVAE) to effectively defend against poisoning attacks with *tuneable overhead* in communication and computation. Whilst the idea of hardening a FL system using generative models is not entirely new, FEDGUARD's original contribution is in its *selective parameter aggregation operator* with parameter selection being driven by *synthetic validation data* sampled from the CVAEs trained locally by each participating party.

Experimental evaluations in a 100-client setup demonstrates FEDGUARD to be more effective than previous approaches against several types of attacks (label and sign flipping, additive noise, same value attacks). FEDGUARD successfully defends in scenarios with up to 50% malicious peers where other strategies fail. In addition, FEDGUARD does not require auxiliary datasets or centralized (pre-) training. It provides resilience against poisoning attacks from the very first round of federated training.

***Index Terms*—federated learning, malicious peer detection, robust federated learning, adversarial attacks, generative models**

I. INTRODUCTION

Federated Learning (FL) is an emerging collaborative paradigm for performing global learning tasks on datasets dispersed amongst a potentially large number of parties without any explicit data exchange. The global learning task is accomplished by training of a local model at each client (*i.e.*, worker node) and sharing only its model parameters with a central server (*i.e., master node)*. The central server is responsible for model initialization, selection of the participating clients per federated round, aggregation of all received local parameters into the global model parameters, as well as sharing back the updated global parameters with the clients. This process is illustrated in Fig. 2.

The FL paradigm is particularly relevant for learning scenarios dependent on large amounts of multi-source data with adamant privacy and confidentiality regulations, e.g. breast density or chest X-ray classification [1], [2].

Due to its distributed nature, a FL system exposes a number of exploitable vulnerabilities for data privacy breach, disruption or manipulation of the learning process through malicious internal or external actors. Considering the criticality of the data involved such as health records, financial data, and personal information, a successful attack on a FL system can result in financial losses, reputational damage, legal liabilities or other severe consequences.

In this work, we focus on adversarial attacks to the federated model and assume the following *threat model*

(TM-1) The FL server is benign.
(TM-2) The federated model is visible to all parties.
(TM-3) The adversary intends to deteriorate the FL model quality.
(TM-4) The adversary corrupts multiple clients.
(TM-5) Malicious clients may collude.
(TM-6) Malicious clients perform *poisoning* attacks.

Poisoning attacks (cf. Fig. 1) are training-time model attacks based on manipulating client's information in order to impair the performance of the federated model on its original learning task. We differentiate between *model poisoning* and *data poisoning* attacks: in the former, the adversary directly manipulates local model updates [3], [4]; in the latter, the attacker manipulates the local training data of one or more clients [5], [6]. We provide an overview and assessment of existing work on poisoning attack mitigation in Section II.

As one of this paper's main contributions, we propose FEDGUARD to effectively defend against poisoning attacks with *tuneable* overhead in communication and computation. We outline FEDGUARD's architecture, its *controllable synthesis of validation data* as well as its *selective parameter aggregation operator* in Section III.

FEDGUARD demonstrates to be more effective against poisoning attacks than previous works, does not require auxiliary datasets or any centralized (pre-) training, and provides resilience against poisoning attacks starting from the initial iteration of federated training.

Our claims are substantiated by experimental evaluation in various poisoning attack scenarios. All details regarding our GRID'5000 experimental testbed, training configuration,

2168-9253/23/$31.00 ©2023 IEEE
DOI 10.1109/CLUSTER52292.2023.00014

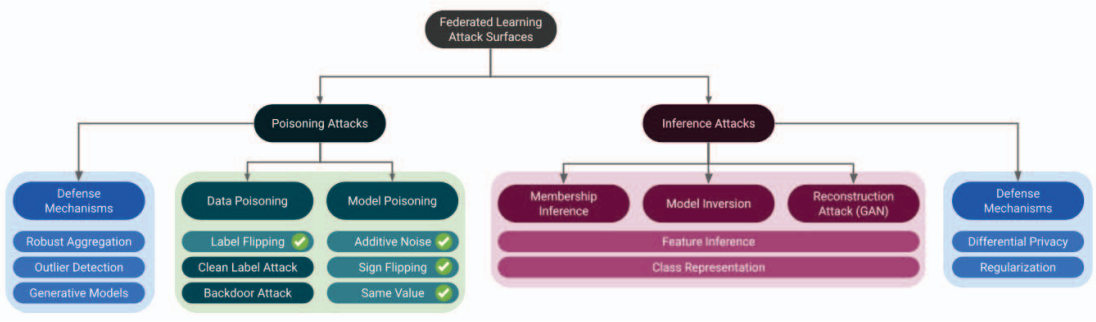

Fig. 1: Federated Learning Attack Surfaces and Defense Mechanisms

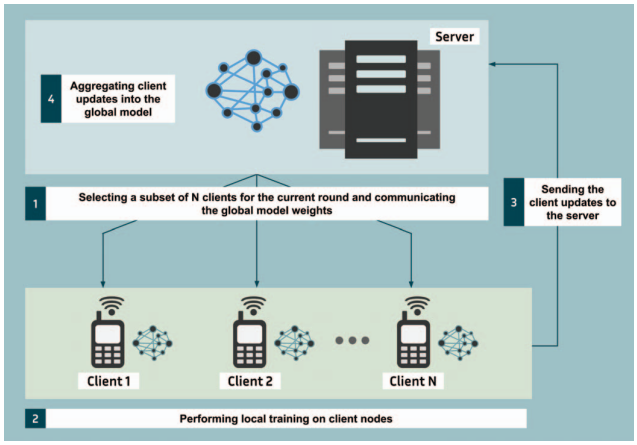

Fig. 2: Federated Learning Process

specific attack scenarios as well as baseline algorithms are provided in Section IV; we present and discussed our evaluation results (cf. Section V) and assess FEDGUARD's main benefits, its current limitations along with possible future directions (cf. Section VI). We conclude with summary in Section VII.

II. Related Work

The related work on poisoning attack mitigation (cf. Fig. 1) can be categorised into three working principles. We briefly outline the key underlying ideas and refer the interested reader to representative works in each category.

Robust aggregation is focused on designing novel aggregation algorithms that are statistically more resilient against outlier and extreme value model updates than the prominent FEDAVG [7] aggregation operator, e.g., by replacing FEDAVG's arithmetic mean with the median or truncated mean [8], the geometric median [9], norm tresholding [10], a centroidality measure [11] or hierarchical averaging [12].

Anomaly detection methods aim at identifying malicious clients as abnormalities in the distribution of local model updates and to exclude their submissions from model aggregation. Popular approaches apply 2-means clustering [13], cosine similarity clustering [14], [15], dynamic clustering [16], (kernel) PCA with subsequent k-means clustering [5], [17] or prediction of expected model updates [18] to tell apart malicious from benign clients. Taking a different approach to anomaly detection, SPECTRAL [19] and FEDCVAE [20] assume the availability of a public dataset that is used for a centralised training of an auxiliary model on the global learning task. While training this auxiliary model, a (conditional) variational autoencoder is trained on the auxiliary model updates to embed to and reconstruct from a low-dimensional latent space. During federated rounds, the pre-trained autoencoder assigns a reconstruction error to all participating clients through encoding and decoding of their local model updates. Clients with high reconstruction errors are deemed malicious and excluded from aggregation.

Generative model based approaches rely on synthesized datasets to audit each client's model performance directly on the global learning task; low-performing submissions are excluded from aggregation. During its initialisation phase, PDGAN [21], [22] indiscriminately aggregates all local model updates into its federated model. Assuming the availability of an auxiliary training dataset and using the federated model as a discriminator, a generative adversarial network (GAN) [23] is trained. Only after this initialisation, each participant's local model performance is audited based on randomly synthesized data. Besides its long initialization[1] and high computational overhead incurred on the FL server, one of the major shortcomings of PDGAN is its inability to condition data synthesis on a target variable. ARMOR [24] addresses this deficiency by server-side training of an individual GAN per target class of the global learning task, however, incurring even higher computational overhead on the FL server.

III. FedGuard

FEDGUARD is a novel poisoning attack mitigation framework for FL systems. It does not require auxiliary datasets or any centralized (pre-) training and provides resilience against poisoning attacks from the very first round of federated training.

FEDGUARD relies on the generative capabilities of the Conditional Variational AutoEncoder (CVAE) framework for

[1]The experimental evaluation [21] reports of 400 and 600 federated rounds of initialization

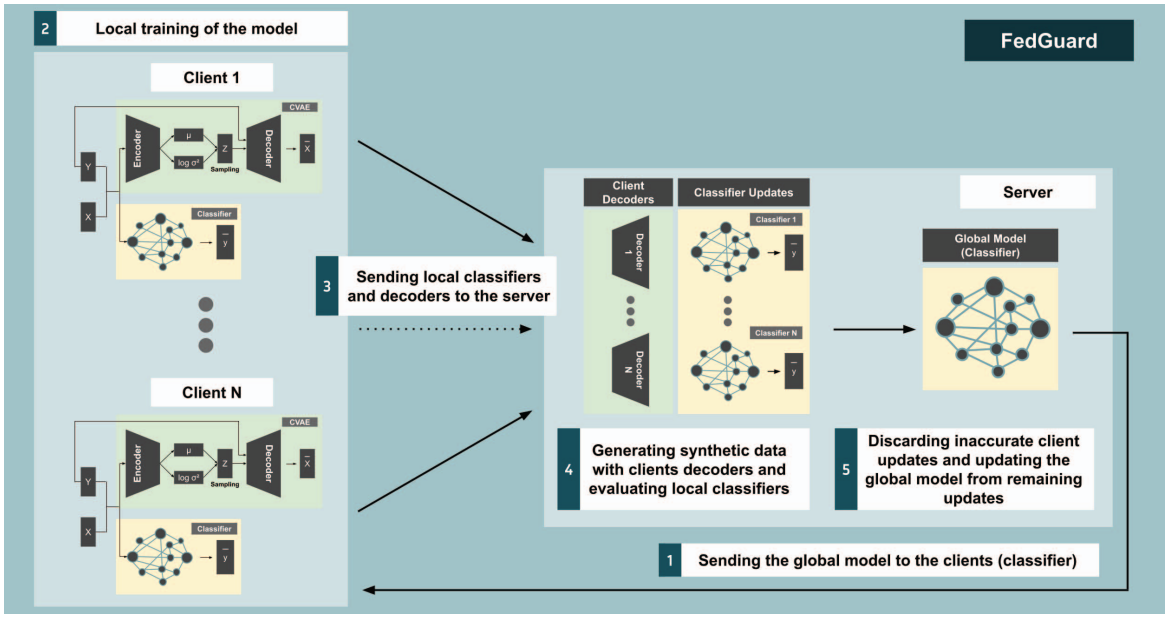

Fig. 3: FEDGUARD execution

a *controllable validation data synthesis* at the FL server. Its *selective parameter aggregation* operator uses the synthesized validation datasets to audit each client's performance on the original federated learning task. Under-performing clients are deemed malicious and excluded from aggregation. The schematic overview of FEDGUARD is shown in Fig. 3.

A. Controllable Synthesis of Validation Data

As outlined above, the core idea behind FEDGUARD is to verify client performance at the server and retain only the classifier updates that exhibit the best performance. Nevertheless, the inherent nature of FL renders it unfeasible for the server to access the data utilized by the clients during local training. Consequently, this data cannot be employed for central evaluation. Some prior works [21], [22] assume the existence of auxiliary datasets at the server. This assumption cannot always be satisfied. An interesting direction to overcome this problem is to employ a *generative model* to synthesize realistic data at the server. One of the prominent architectures belonging to the generative category is Variational AutoEncoders (VAEs) [25].

The VAEs provides a computationally efficient way for estimating the density $p(\mathbf{x})$ of observed data $\mathbf{x} \in \mathcal{X}$ by assuming a latent variable model $p(\mathbf{x}) = \mathbb{E}_{\mathbf{z}\sim p_\phi(\mathbf{z}\,|\,\mathbf{x})}[p_\theta(\mathbf{x}\,|\,\mathbf{z})]$ with a *latent variable* $\mathbf{z} \in \mathcal{Z}$, a *likelihood distribution* $p_\theta(\mathbf{x}|\mathbf{z})$ and a true but intractable *posterior distribution* $p_\phi(\mathbf{z}\,|\,\mathbf{x})$.

$$\log p(\mathbf{x}) - \mathbb{KL}[\overbrace{q_\phi(\mathbf{z}|\mathbf{x})}^{\text{approx. posterior}} || \overbrace{p_\phi(\mathbf{z}|\mathbf{x})}^{\text{true posterior}}] = \tag{1}$$

$$\underbrace{\underbrace{\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log \overbrace{p_\theta(\mathbf{x}|\mathbf{z})}^{\text{likelihood}}]}_{\text{reconstruction}} - \underbrace{\mathbb{KL}[q_\phi(\mathbf{z}\,|\,\mathbf{x})|\overbrace{p(\mathbf{z})}^{\text{prior}}]}_{\text{regularisation}}}_{\text{ELBO}} \tag{2}$$

VAEs are trained by maximizing the ELBO objective (cf. Eqn. 2) which effectively allows to approximate the intractable posterior $p_\phi(\mathbf{z}\,|\,\mathbf{x})$ using an approximate posterior distribution $q_\phi(\mathbf{z}\,|\,\mathbf{x})$. In turn, the *approximate posterior* $q_\phi(\mathbf{z}\,|\,\mathbf{x})$ is regularized by a known and chosen *prior* $p(\mathbf{z})$ of the latent variable. A VAE employs a *decoder* function $D_\theta : \mathcal{Z} \to \mathcal{X}$ to model and parameterize its likelihood distribution $p_\theta(\mathbf{x}|\mathbf{z})$; the approximate posterior distribution $q_\phi(\mathbf{z}\,|\,\mathbf{x})$ is modeled and parameterised using an *encoder* function $E_\phi : \mathcal{X} \to \mathcal{Z}$. Put simply, the encoder compresses the input data to a low-dimensional latent representation, which can be sampled by the decoder to generate input-like data.

Since its inception, the VAE framework has been extended in many directions. A *Conditional Variational Autoencoder* (CVAE) [26] allows to condition its

$$\text{Conditional Decoder:} \quad D_\theta : \mathcal{Z} \times \mathcal{Y} \to \mathcal{X} \tag{3}$$

$$\text{Conditional Encoder:} \quad E_\phi : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z} \tag{4}$$

on an additional variable $\mathbf{y}$ over a discrete categorical distribution $\text{Cat}(L, \alpha)$ expressing the target labels of the observed dataset $\mathcal{D} \subset \mathcal{X} \times \mathcal{Y}$. The uniqueness of a CVAE as compared to a VAE, is its ability to generate images at the decoder by conditioning it on an additional categorical variable (Eqn.3). In other words, providing a class label (*e.g.*, MNIST '3') and a prior sample into the CVAE decoder, synthesizes a novel MNIST '3' image. This ability to generate data *controllably* allows FEDGUARD to audit client accuracy on specific inputs.

As with VAEs, a CVAE is trained by optimizing

$$\arg\min_{\phi,\theta} \mathcal{L}_{\texttt{CVAE}}(\phi, \theta) \tag{5}$$

$$= \arg\min_{\phi,\theta} [\underbrace{\mathbb{KL}[E_\phi||p(\mathbf{z})]}_{\text{regularization}} - \underbrace{\mathbb{E}_{\mathcal{D}}[\log D_\theta(E_\phi(x, y), y)]}_{\text{reconstruction}}] \tag{6}$$

where the reconstruction term is the expected log-likelihood with respect to the decoder's distribution; the regularization term is the Kullback-Leibler divergence between the encoder's distribution and the prior distribution.

After convergent training, a CVAE's conditional decoder (cf. Eqn. 3) allows for *controllable data synthesis* using a latent space sample from its prior distribution and a conditioning sample from its discrete categorical distribution. Data synthesized using a CVAE is representative for observed data in that it maximizes its expected log-likelihood (cf. Eqn. 2).

Controllable data synthesis is unique to conditional generative models, i.e. CVAEs or Conditional GANs (CGAN). CGANs are more heavy-weight and demanding to train with common problems like vanishing gradients, mode collapses and convergence failures. For this reason, we choose to employ a CVAE in our approach.

B. Selective Parameter Aggregation

As in any centralized federated learning setting, FEDGUARD aims at performing a global learning task on a dataset $\mathcal{D}$ dispersed amongst N participating clients (cf. Alg. 1: lines 10-12) without explicitly exchanging any data samples between parties. The global learning task is accomplished by training of a local model on each client's partition $\mathcal{P}_{1\leq j\leq N}$ (cf. Alg. 1: line 26) and sharing only its parameters ψ_j with a central server (cf. Alg. 1: line 18).

The central server is responsible for model initialization (cf. Alg. 1: line 15), selection of a subset J of participating clients per federated round (cf. Alg. 1: line 17), aggregation of all received local parameters into the global model parameters ψ_0 (cf. Alg. 1: line 19), as well as sharing back the updated global parameters ψ_0 with the clients (cf. Alg. 1: line 18).

(1) In addition to performing the federated learning task, FEDGUARD requires all clients to train a CVAE on their private local datasets $\mathcal{P}_{1\leq j\leq N}$ (cf. Alg. 1: line 25) and share their CVAE decoder parameters θ_j with the central server (cf. Alg. 1: line 18). The decoders enable the controllable synthesis of validation data at the server.

(2) Per federated round, the central server obtains a *synthetic validation dataset* $\mathcal{D}_{\text{syn}}$ from the active clients' decoders $D_{\theta_{j\in J}}$ using t latent space samples $[z_t]$ from the prior distribution $\mathbf{z}\sim\mathcal{N}(0,1)$ as well as t conditioning samples $[y_t]$ from a discrete categorical distribution $\mathbf{y}\sim\text{Cat}(L,\alpha)$ (cf. Alg. 1: lines 2-4). Sampling from the categorical with known or estimated parameters makes validation data statistically similar to the global dataset.

(3) The synthetic validation data is then used to evaluate and score each active client's average local model performance with respect to the federated learning task using an appropriate metric $\mathcal{L}_{\text{ACC}}$ (cf. Alg. 1: line 5).

(4) FEDGUARD will *selectively aggregate* only those local parameter updates into the global model ψ_o (cf. Alg. 1: line 7) that perform above the overall average performance $\overline{\text{ACC}}$ (cf. Alg. 1: line 6).

Algorithm 1: `FedGuard`

1 **Function** `FedGuard(`$[\theta_{j\in J}]$, $[\psi_{j\in J}]$, L, α, t`)`**:**
2 $\quad\mathbf{z}\sim\mathcal{N}(0,1)$
3 $\quad\mathbf{y}\sim\text{Cat}(L,\alpha)$
4 $\quad\mathcal{D}_{\text{syn}}\leftarrow$ `map(` $D_{\theta_j}([z_t],[y_t])$, $[\theta_{j\in J}]$`)`
5 $\quad\overline{\text{ACC}}_j\leftarrow$ `map(` $\mathcal{L}_{ACC}(f_{\psi_j}(\mathcal{D}_{syn},[y_t]))$, $[\psi_{j\in J}]$`)`
6 $\quad\overline{\text{ACC}}\leftarrow\frac{1}{m}\sum_{j\in J}\overline{\text{ACC}}_j$
7 $\quad$**return** `FedAvg(filter(`$[\psi_{j\in J}]$, $\overline{ACC}_{j\in J}\geq\overline{ACC}$`))`
8
9 **Procedure** `Federation(`$\mathcal{D}$, N, m, R, L, α, t`)`**:**
10 $\quad[\mathcal{P}_{1\leq i\leq N}]\leftarrow$ `split(`N, $\mathcal{D}$`)`
11 $\quad[\mathcal{C}_{1\leq i\leq N}]\leftarrow$ `map(partial(Client),` $[\mathcal{P}_{1\leq i\leq N}]$`)`
12 $\quad$`Server(`$[\mathcal{C}_{1\leq i\leq N}], N, m, R, L, \alpha, t$`)`
13
14 **Function** `Server(`$[\mathcal{C}_{1\leq i\leq N}], N, m, R, L, \alpha, t$`)`**:**
15 $\quad\psi_0\leftarrow$ `init()`
16 $\quad$**foreach** $r\in$ `range(`$1, R$`)` **do**
17 $\quad\quad J\leftarrow$ `sample((range(`$1, N$`),` m`)`
18 $\quad\quad[\theta_{j\in J}],[\psi_{j\in J}]\leftarrow$ `unzip(`$[\mathcal{C}_{j\in J}(\psi_0)]$`)`
19 $\quad\quad\psi_0\leftarrow$ `FedGuard(`$[\theta_{j\in J}]$,$[\psi_{j\in J}]$, L, α, t`)`
20 $\quad$**end foreach**
21
22 **Function** `Client(`$\mathcal{P}$, ψ^*`)`**:**
23 $\quad\mathcal{X},\mathcal{Y}\leftarrow\mathcal{P}$
24 $\quad\psi\leftarrow\psi^*$
25 $\quad\phi^*,\theta^*\leftarrow\arg\min_{\phi,\theta}\mathcal{L}_{\text{CVAE}}(\phi,\theta)$
26 $\quad\psi^*\leftarrow\arg\min_{\psi}\mathbb{E}_{\mathcal{X}}\left[\mathbb{E}_{\mathcal{Y}\mid\mathcal{X}}\left[\mathcal{L}_{\text{MNIST}}(\mathbf{x},\mathbf{y},\psi)\right]\right)$
27 $\quad$**return** θ^*, ψ^*

TABLE I: Definitions of notations in Algorithm 1

Notation	Definition
Indices:	
N	Number of clients ($i\in\{1,...,N\}$)
R	Number of rounds ($r\in\{1,...,R\}$)
m	Number of clients sampled per round
Parameters:	
$\mathcal{D}$	Dataset
ψ	Classifier parameters
ϕ	Encoder parameters
θ	Decoder parameters
α	Assumed probability per class
J	Subset of clients
L	Number of classes within the dataset
t	Number of synthetic samples per decoder

IV. EXPERIMENTAL DETAILS

We now detail on the Federated Learning objective and setup, the considered poisoning attack scenarios, our evaluation baselines and specific training configurations as well as our experimental testbed for evaluating the effectiveness of FEDGUARD against the state of the art. All relevant source code can be found online[2].

A. FL objective and setup

MNIST digits classification [27] is the global learning task for our evaluation experiments. Throughout all experiments, the *very same* MNIST classifier architecture (cf. Table II) is used. It is composed of two ReLU-activated 5x5 convolution layers with 32 and 64 channels (each followed with 2x2 max

[2]https://github.com/Melvin-chelli/FedGuard

TABLE II: Classifier architecture

Layer	Type	Kernel	Input Shape	Output Shape	Activation	Parameters
1	Conv2d	(5, 5)	(28, 28)	(32, 26, 26)	ReLu	800
2	MaxPool2d	(2, 2)	(32, 26, 26)	(32, 14, 14)	-	-
3	Conv2d	(5, 5)	(32, 14, 14)	(64, 12, 12)	ReLu	51,200
4	MaxPool2d	(2, 2)	(64, 12, 12)	(64, 7, 7)	-	-
5	Flatten	-	(64, 7, 7)	(3136)	-	-
6	Linear	-	(3136)	(512)	ReLu	1,605,632
7	Linear	-	(512)	(10)	Softmax	5,120
Total Parameters: 1,662,752				*Total Size (MB): 6.65*		

TABLE III: CVAE architecture

Layer	Type	Kernel	Input Shape	Output Shape	Activation	Parameters
Encoder Network						
1	Linear	-	(794)	(400)	ReLu	318,000
2a	Linear	-	(400)	(20)	ReLu	8,020
2b	Linear	-	(400)	(20)	ReLu	8,020
Decoder Network						
3	Linear	-	(30)	(400)	ReLu	12,400
4	Linear	-	(400)	(794)	Sigmoid	318,394
Size of the Encoder (MB): 1.34				*Size of the Decoder (MB): 1.32*		
Total Parameters: 664,834				*Total Size (MB): 2.66*		

pooling), a ReLu-activated fully connected layer (FCL) with 512 units, and a softmax-activated 10-units output FCL.

Each client will train the MNIST classifier for 5 epochs on its private local dataset. For that purpose, the entire MNIST dataset is partitioned between $N = 100$ clients as per the Dirichlet distribution [28, $\alpha = 10$] to simulate real-world data distribution. In each federated round, the FL server uniformly samples $m = 50$ participating clients.

B. Attack scenarios

We investigate the following scenarios based on four different types of poisoning attack.

A *same-value attack* [3] sets all the weights of the local model update to the same value such as $w_k = c * \vec{1}$ where w_k is the local model update, c is a constant and $\vec{1}$ is an all-one vector. For our experiments, $c = 1$, which corresponds to setting all the weights of the local model updates to 1. In this scenario, we set the number of malicious peers to 50%.

A *sign-flipping attack* [3] reverses the sign of all the weights of the local model update such as $w_k = -1 * w_k$ where w_k is the local model update. The magnitude of model update weights stays the same and therefore can be a point of failure for defenses based on norm thresholding [10]. In this scenario, we set the number of malicious peers to 50%.

A *label-flipping attack* [29] is a data poisoning attack that flips the labels of the input data before training the local model on it. In related scenarios, malicious clients flip the labels from data representing the MNIST digits 5 and 7, as well as digits 4 and 2. This attack is a targeted attack which aims at making the model misclassify a subset of classes. The overall performance of the resulting model is less affected than in untargeted attack scenarios which makes it more challenging to detect. As a result, we evaluate the performance of the different strategies in two distinct scenarios in which we set the number of malicious peers to 30% and 40%.

An *additive noise attack* [30] adds a Gaussian noise to the local model update such as $w_k = w_k + \epsilon$ where w_k is the local model update and ϵ is the Gaussian noise. In this scenario, malicious clients performing this attack all agree on the same Gaussian noise to add to the local model updates. We set the number of malicious peers to 50%.

C. Evaluation baselines[3]

FEDAVG [7] is the standard approach for FL. It uses weighted averaging of the client updates to build the global model update. It does not come with any defense mechanism.

GEOMED [12] updates the global model by using the geometric median of local model updates. With this approach, outlier updates which are likely to be far from benign updates should not be used to build the global model update.

KRUM [11] computes a score for each local update given its proximity to the others and selects the one with the best score. Benign updates target a same objective, therefore should be close to each other and get selected instead of outlier updates.

SPECTRAL[4] [19] uses a pre-trained variational auto-encoder to reconstruct local model updates and discard updates with a reconstruction error above a dynamic threshold set to the mean of all reconstruction errors.

D. FEDGUARD's configuration

We implement FEDGUARD's conditional encoder (cf. Eqn. 4) using three ReLu-activated FCLs (400, 20 and 20 units); its decoder (cf. Eqn. 3) is composed of a ReLu-activated

[3]Unfortunately, we could not find any open implementation of FedC-VAE [20] or PDGAN [21].

[4]We use the implementation found at https://github.com/Suyi32/Learning-to-Detect-Malicious-Clients-for-Robust-FL

400-units FCL followed by a sigmoid-activated 794-units FCL (cf. Table III). Each FEDGUARD client will train the CVAE for 30 epochs on its private local dataset[5]. In each federated round, we generate $t = 2 * m = 100$ samples from the latent variable $\mathbf{z} \sim \mathcal{N}(0, 1)$ as well as from the conditioning variable $\mathbf{y} \sim \text{Cat}(L = 10, [\alpha_{1 \leq i \leq L} = \frac{1}{L}])$ resulting in a class-balanced validation dataset of 100 synthetic MNIST digits.

E. Experimental Platform

We rely on E2CLAB [31] to automatically deploy our experiments to the GRID'5000 experimental testbed. We use a total of 5 nodes equiped with 2 Nvidia Tesla P100 (16GB) and 2 Intel Xeon Gold 6126 (12 cores/CPU) each. The server runs on a single node, while 100 clients are deployed on the 4 remaining nodes (25 clients per node). Each node is given access to a ethernet configuration with a 10GBps rate.

V. RESULTS AND COMPARISONS

The results of the experiments conducted as described in the previous section are presented in Fig. 4. The average accuracy achieved by each strategy over the last 40 rounds of training is reported in table IV (we do not average the 10 first rounds of training because the model has not converge yet which would lead to a inaccurate comparison). Table V reports communication and computation overhead of the strategies. In the remainder of this section, we provide discussions on the robustness and performance of each strategy with regard to the several evaluation scenarios.

A. Robustness in malicious scenarios

Additive Noise - 50% Malicious peers. As illustrated in Fig. 4, both FEDGUARD and SPECTRAL successfuly defend against the additive noise attack and achieve similar convergence as in a scenario with no attack. Results from table IV show that SPECTRAL achieves the same accuracy as in a scenario with no attack by reaching an average accuracy of 98.97%, and FEDGUARD achieves almost similar accuracy with an average of 98.72%. FEDAVG, GEOMED and KRUM, on the other hand, are unable to defend in this scenario by achieving a very low average accuracy of less than 8%.

Label Flipping - 30% Malicious peers. In this scenario, as explained in section IV, malicious peers are performing a targeted attack which is more challenging to detect, therefore we investigate the performance of each strategy in a coordinated attack involving 30% malicious peers. In this scenario, FEDGUARD is the strategy achieving the best performance and stability with an average accuracy of 98.96% and standard deviation of 0.17%. GEOMED, in contrast which achieves the second best performance with an average accuracy of 98.13% has a standard deviation of 1.63%. FEDAVG and SPECTRAL are very unstable with standard deviations of more than 6%. KRUM achieves the worst accuracy with an average of 96.51%.

Sign Flipping - 50% Malicious peers. Regarding the sign flipping attack, FEDGUARD is again defending very well achieving very stable training and high average accuracy (98.97%). All other approaches are very unstable with at least 14% standard deviations and cannot converge to an optimum.

Same value - 50% Malicious peers. In the same value scenario, FEDGUARD and SPECTRAL both defend very well and achieve same accuracy as in a scenario with no attack. FEDAVG, GEOMED and KRUM , on the other hand, all fail in defending this attack and achieves average accuracy of 10%.

GEOMED and KRUM, which use distance-based defenses are unable to defend in situations involving a majority of malicious peers. In fact, if a majority of malicious clients are able to run a coordinated attack, they will have a great probability of not being detected, and therefore poison the global model. Previously, SPECTRAL has been shown to defend very well in various scenarios including sign-flipping attack [19]. However, it seems that with regard to the model used in our experiments, the corresponding surrogate vectors are not accurate enough. Using higher dimensional surrogate vectors would not be a suitable solution as it would highly increase the computational cost. Moreover, the bigger the classifier to train, the bigger the surrogate vector would have to be. FEDGUARD, on the other hand, could defend in all scenarios by providing a solution that evaluates the quality of each local update and selects them according to the description in section III-B. Our approach can defend very well against untargeted attacks which introduce no logic into the model updates and therefore achieve low evaluation accuracy on the synthetic data. Targeted attacks, on the other hand that introduce biased logic is more challenging to defend against. Still, our strategy given its architecture should be able to defend up to an upper limit of 50% malicious peers selected for a given round. With our experiments, we show that it defends particularly well in a scenario with a total of 30% malicious peers performing label flipping.

Testing FEDGUARD limits: We go further and test the limits of FEDGUARD by running experiments with 40% malicious peers performing label flipping.

Label Flipping - 40% Malicious peers. In this scenario, FEDGUARD starts to suffer from some instabilities that can arise occasionally when a great number of malicious clients is sampled as represented in Fig. 5 (FedGuard-lr-1). As the instabilities seems to happen only occasionally, a rather straightforward solution is to introduce a learning rate at the server while updating the global model with incoming local updates. While slowing down the convergence rate of the model, adopting a learning rate at the server should mitigate the impact of occasional failures from the strategy. We set the server learning rate to 0.3 (instead of 1 in the standard case) and run the same experiment. As shown in Fig. 5 (FedGuard-lr-0.3), by adopting a server learning rate, we can achieve a better stability. The model takes more time to converge while benefiting from a better robustness against occasional failures of the defensive mechanism.

B. System overhead

Since our defensive approach relies on the transmission of an additional element between the clients and the server (*i.e*,

[5]In our experiments, the MNIST partitioning is static. Therefore, a FEDGUARD client trains its CVAE only once.

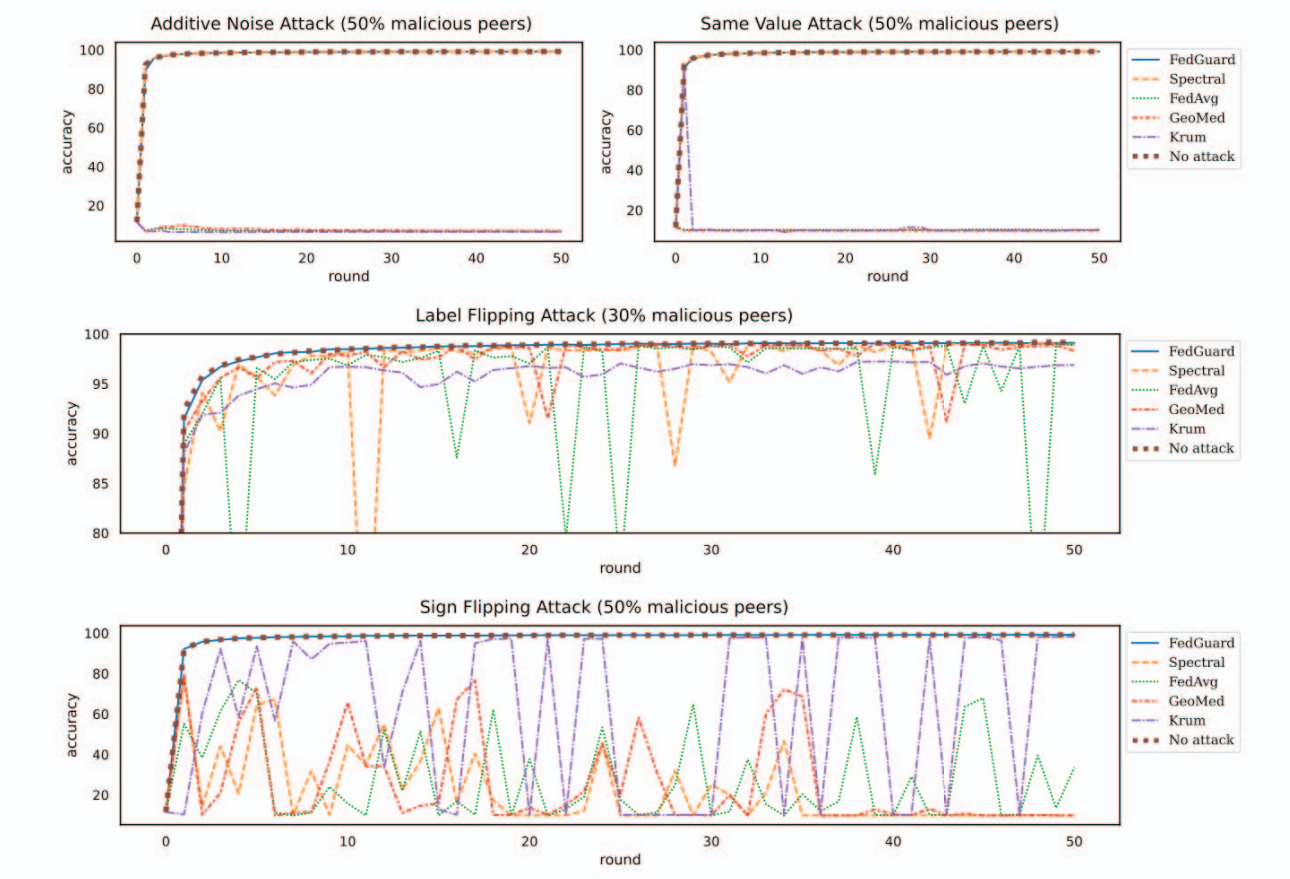

Fig. 4: Accuracies of various strategies in different attack scenarios

TABLE IV: Average accuracy and standard deviation over the last 40 rounds of training

Strategy	Additive Noise Attack 50% malicious peers	Label Flipping Attack 30% malicious peers	Sign Flipping Attack 50% malicious peers	Same Value Attack 50% malicious peers
FEDAVG [7]	$6.87\% \pm 0.12\%$	$95.80\% \pm 6.66\%$	$24.21\% \pm 18.74\%$	$10.16\% \pm 0.09\%$
GEOMED [12]	$7.26\% \pm 0.31\%$	$98.13\% \pm 1.63\%$	$23.66\% \pm 21.56\%$	$9.78\% \pm 0.0\%$
KRUM [11]	$6.52\% \pm 0.46\%$	$96.51\% \pm 0.59\%$	$62.48\% \pm 41.96\%$	$9.93\% \pm 0.45\%$
SPECTRAL [19]	$\mathbf{98.97\% \pm 0.18\%}$	$96.91\% \pm 6.12\%$	$18.95\% \pm 14.81\%$	$98.97\% \pm 0.17\%$
FEDGUARD (Ours)	$98.72\% \pm 0.60\%$	$\mathbf{98.96\% \pm 0.17\%}$	$\mathbf{98.97\% \pm 0.22\%}$	$\mathbf{98.99\% \pm 0.19\%}$
No attack	$98.97\% \pm 0.17\%$	$98.97\% \pm 0.17\%$	$98.97\% \pm 0.17\%$	$98.97\% \pm 0.17\%$

the decoder of the CVAE), it results in some communication overhead. Similarly, the local training of the CVAE for the clients and the data generation at the server result in computational and training time overheads. In table V, we provide communication and time metrics of the several strategies obtained during our experiments. We compare against FEDAVG, which involves no additional mechanism for defense.

Regarding bandwidth usage, GEOMED, KRUM and SPECTRAL do not involve additional transmissions compared to FEDAVG. FEDGUARD, on the other hand, involves an additional communication of the decoders from the client to the server. During our experiments, we measure an overhead of 20% for server downloads, which results in a total bandwidth overhead of 10% (considering uploads and downloads).

Regarding training time, GEOMED provides the most time efficient approach by achieving a 24% overhead in comparison to FEDAVG. FEDGUARD and SPECTRAL provides similar training time overhead with 82% and 84% respectively, which is the consequence of using an additional auto-encoder for generating evaluation data or reconstructing model updates. Finally, KRUM is the most time consuming strategy with 95% overhead, resulting from the need to compute distances between all local updates which can become computationally expensive with the growing number of clients sampled in each round.

The overhead of our defensive mechanism is highly related to the size of the CVAE which impacts the training and communication overhead. As a consequence, the final overhead

TABLE V: System overhead of the defensive strategies

Strategy	Server uploads / round	Server downloads / round	Server total communication / round	Training time / round
FEDAVG [7]	348.3 MB	348.3 MB	696.6 MB	3.76 s
GEOMED [12]	348.3 MB	348.3 MB	696.6 MB	4.66 s (+24%)
KRUM [11]	348.3 MB	348.3 MB	696.6 MB	7.32 s (+95%)
SPECTRAL [19]	348.3 MB	348.3 MB	696.6 MB	6.94 s (+84%)
FEDGUARD (Ours)	349.3 MB	417.4 MB (+20%)	766.7 MB (+10%)	6.86 s (+82%)

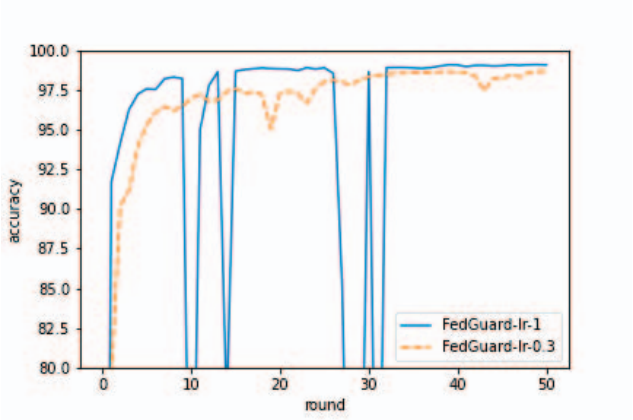

Fig. 5: Impact of learning rate on FEDGUARD stability (40% malicious peers, label flipping attack)

of our strategy can be seen as a ratio between the size of the classifier to train and the size of the autoencoder. If the size of the autoencoder is relatively small compared to the targeted model, then the overhead of our strategy will be relatively small. On the other hand, if the size of the autoencoder is close to, or higher than the size of the targeted model, the resulting overhead will be high. In critical applications requiring a highly accurate model, it might be worth paying such an overhead to ensure the security of the system. In less critical applications, finding a trade-off between the size of the targeted model and the size of the autoencoder can be a solution to combine system performance and security of the resulting model.

VI. DISCUSSIONS

In this section we outline the main benefits of FEDGUARD, its limitations as well as its generality. We also outline few directions for future works.

A. Main benefits of FEDGUARD

Works out of the box. SPECTRAL [19] and FEDCVAE [20] both require the training of an auto-encoder prior to the actual FL training. FEDGUARD in comparison can directly be used without the need to pre-train specific models which makes it handy to use.

No need for preparation phase. The defensive mechanism proposed in PDGAN [21] requires a preparation phase during which the GAN is trained. This phase can last for many rounds (*e.g.*, 400 rounds), which can make the system vulnerable for a long period. FEDGUARD, in contrast, does not need any preparation phase to work and is effective from the first round of training.

Ability to generate specific evaluation data. PDGAN uses a GAN for data generation at the server to evaluate local client updates. While providing evaluation data, it cannot condition it on specific labels which leads to unpredictable generation. The class of generated data is unknown, and some classes might even not be generated. This can be a point of failure in specific attack scenarios. FEDGUARD, by using CVAEs, has the ability to generate evaluation data conditioned on specific labels, therefore provides better evaluation capability. Moreover, with FEDGUARD the true labels of evaluation data is known.

No need for an auxiliary dataset. PDGAN, SPECTRAL and FEDCVAE, all three require the existence of an auxiliary dataset on the server to work. PDGAN needs it to train its GAN during the FL process, while SPECTRAL and FEDCVAE use it to pre-train their auto-encoder. The existence of such dataset cannot be satisfied in all scenarios. The server may have very limited access or no given access to auxiliary samples. FEDGUARD, on the other hand, does not require such dataset to work.

Tuneable system. By selecting the number of decoders to use for generating evaluation data or by changing the total number of evaluation data to generate, one can balance between evaluation data diversity (ensuring a higher security) and resource usage at the server level. The quantity of data to generate can be selected for each class, and therefore could be increased for specific class of data which is considered more critical. The system can be configured with respect to the computing resources available at the server level and the criticality of the application.

B. Constraints and Generality

Limiting factors. FEDGUARD was evaluated in a setup where the entire dataset is partitioned between 100 clients which lets clients sufficient data to train local CVAEs for generation of synthetic data. A highly heterogeneous setting where clients only have very limited data representing very few classes to train locally could be a limiting factor for our strategy. While having few clients in this state should not hurt the effectiveness of the strategy, having a majority of clients with very few data could be a real problem for data synthesis. One solution could be to share, in addition to the client decoders, the classes each decoder was trained on for data

generation. Using this information, the server could generate specific data with regard to what each decoder was trained on. FEDGUARD can reach another limit regarding the number of malicious decoders. If the decoders sent from malicious peers are trained with regard to a malicious objective (*e.g.*, label flipping) and are in a majority position, the evaluation process at the server will be highly impacted and risks to fail in its defense.

Learning rate as a defensive mechanism. In FL, a subset of clients is sampled from a possibly large number of devices. While the probability of sampling a majority of malicious clients is low, a single failure from the defensive mechanism that lets an attack succeed can be very harmful and have dramatic consequences. The use of a learning rate at the server appears to be an interesting additional defensive mechanism to mitigate the impact of a failure from the main defensive mechanism.

C. Future Works

Internal aggregation operator. Currently FEDGUARD uses the FEDAVG internal aggregation operator to update the global model. However, FEDGUARD's design makes it easy to switch from one internal aggregation operator to another. In future works, we could investigate the use of other operators such as GEOMED or FEDPROX [32] aggregation operators which could further improve the robustness of FEDGUARD.

Imbalanced datasets. In this paper, we validated the effectiveness of FEDGUARD in a 100-client setup where data is partitioned according to a Dirichlet distribution with $\alpha = 10$ which simulates a real-world data partitioning. Investigating the robustness of FEDGUARD under different levels of dataset imbalance, including highly imbalanced dataset with very few samples per client, is one direction for future works.

Dynamic datasets. FEDGUARD was evaluated in a scenario where FL clients have access to static datasets. Evaluating it in a setup where clients get access to a stream of incoming data is more realistic. Therefore, we would like to investigate how FEDGUARD and other strategies perform in such setup. We would like to further investigate how the number of local epochs and how often performing the local training of the CVAE impacts FEDGUARD's performance in such scenario.

VII. CONCLUSION

Federated Learning has now reached a state that enables it to be applied in various domains. Unfortunately, with its increased adoption in safety-critical applications, also comes several threats linked to its decentralized nature. In this paper, we are interested in finding an efficient mechanism for detection of malicious peers injecting corrupted updates into the distributed system. We propose FEDGUARD, a novel FL strategy which takes advantage of Conditional Variational AutoEncoders trained locally by the clients to synthesis validation data at the server. FEDGUARD relies on a selective aggregation operator which flags updates as malicious given their performance on the synthesised validation dataset. We evaluate and compare our approach with several baseline algorithms in multiple challenging scenarios involving up to 50% malicious peers performing model and data poisoning attacks. FEDGUARD is shown to be very robust by achieving the best defense in almost all scenarios, while resulting in negligible system overhead compared to other approaches. In short, FEDGUARD provides an effective and tuneable framework for robust FL in poisoning scenarios, yet its underlying mechanism could further be used in many other applications including detection of defective sensors in volatile environments or for enabling a better sampling of quality candidates in FL systems.

AUTHOR CONTRIBUTION STATEMENT

Conceptualization: M.C., C.P., R.S.; Methodology: M.C., C.P, R.S.; Software: M.C., C.P.; Validation: M.C., C.P.; Formal analysis: M.C., C.P., R.S.; Investigation: M.C., C.P.; Resources: M.C., C.P., A.C., G.A., R.S. Data curation: M.C., C.P., Writing (original draft): M.C., C.P., R.S.; Writing (review & editing): M.C., C.P., R.S., A.C., G.A.; Visualization: C.P.; Supervision: A.C., G.A., L.C., R.S., P.S.; Project administration: R.S., A.C., G.A., P.S.; Funding acquisition: R.S., G.A., P.S.

ACKNOWLEDGMENT

This publication resulted (in part) from ENGAGE, a French-German Inria-DFKI collaborative project. On the German side, it is supported by the German **Federal Ministry of Education and Research** (BMBF) under grant number 01IS21106. Experiments presented in this paper were carried out using the French Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr).

REFERENCES

[1] H. R. Roth, K. Chang, P. Singh *et al.*, "Federated learning for breast density classification: A real-world implementation," in *Domain Adaptation and Representation Transfer, and Distributed and Collaborative Learning: Second MICCAI Workshop, DART 2020, and First MICCAI Workshop, DCL 2020, Held in Conjunction with MICCAI 2020, Lima, Peru, October 4–8, 2020, Proceedings*. Berlin, Heidelberg: Springer-Verlag, 2020, p. 181–191. [Online]. Available: https://doi.org/10.1007/978-3-030-60548-3%5F18

[2] I. Feki, S. Ammar, Y. Kessentini *et al.*, "Federated learning for covid-19 screening from chest x-ray images," *Applied Soft Computing*, vol. 106, p. 107330, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494621002532

[3] L. Li, W. Xu, T. Chen *et al.*, "RSA: byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets," *CoRR*, vol. abs/1811.03761, 2018. [Online]. Available: http://arxiv.org/abs/1811.03761

[4] W. Chen, Z. Zhang, X. Hu *et al.*, "Boosting decision-based black-box adversarial attacks with random sign flip," in *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Cham: Springer International Publishing, 2020, pp. 276–293.

[5] V. Tolpegin, S. Truex, M. E. Gursoy *et al.*, "Data poisoning attacks against federated learning systems," in *Computer Security – ESORICS 2020*, L. Chen, N. Li, K. Liang, and S. Schneider, Eds. Cham: Springer International Publishing, 2020, pp. 480–501.

[6] G. Sun, Y. Cong, J. Dong *et al.*, "Data poisoning attacks on federated machine learning," *IEEE Internet of Things Journal*, vol. 9, no. 13, pp. 11 365–11 375, 2021.

[7] H. B. McMahan, E. Moore, D. Ramage *et al.*, "Federated learning of deep networks using model averaging," *CoRR*, vol. abs/1602.05629, 2016. [Online]. Available: http://arxiv.org/abs/1602.05629

[8] D. Yin, Y. Chen, R. Kannan *et al.*, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. Pmlr, 10–15 Jul 2018, pp. 5650–5659. [Online]. Available: https://proceedings.mlr.press/v80/yin18a.html

[9] Z. Wu, Q. Ling, T. Chen *et al.*, "Federated variance-reduced stochastic gradient descent with robustness to byzantine attacks," *IEEE Transactions on Signal Processing*, vol. 68, pp. 4583–4596, 2020.

[10] Z. Sun, P. Kairouz, A. T. Suresh *et al.*, "Can you really backdoor federated learning?" *arXiv preprint arXiv:1911.07963*, 2019.

[11] P. Blanchard, E. M. El Mhamdi, R. Guerraoui *et al.*, "Machine learning with adversaries: Byzantine tolerant gradient descent," *Advances in neural information processing systems*, vol. 30, 2017.

[12] Y. Chen, L. Su, and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," in *Abstracts of the 2018 ACM International Conference on Measurement and Modeling of Computer Systems*, 2018, pp. 96–96.

[13] S. Shen, S. Tople, and P. Saxena, "AUROR: Defending against poisoning attacks in collaborative deep learning systems," in *Proceedings of the 32nd Annual Conference on Computer Security Applications*, ser. Acsac '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 508–519. [Online]. Available: https://doi.org/10.1145/2991079.2991125

[14] F. Sattler, K.-R. Müller, T. Wiegand *et al.*, "On the byzantine robustness of clustered federated learning," *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8861–8865, 2020.

[15] S. Awan, B. Luo, and F. Li, "CONTRA: Defending against poisoning attacks in federated learning," in *Computer Security–ESORICS 2021: 26th European Symposium on Research in Computer Security, Darmstadt, Germany, October 4–8, 2021, Proceedings, Part I 26*. Springer, 2021, pp. 455–475.

[16] T. D. Nguyen, P. Rieger, M. H. Yalame *et al.*, "Flguard: Secure and private federated learning," *Crytography and Security*, no. Preprint, 2021.

[17] D. Li, W. E. Wong, W. Wang *et al.*, "Detection and mitigation of label-flipping attacks in federated learning systems with kpca and k-means," in *2021 8th International Conference on Dependable Systems and Their Applications (DSA)*. IEEE, 2021, pp. 551–559.

[18] Z. Zhang, X. Cao, J. Jia *et al.*, "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*, ser. Kdd '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 2545–2555. [Online]. Available: https://doi.org/10.1145/3534678.3539231

[19] S. Li, Y. Cheng, W. Wang *et al.*, "Learning to detect malicious clients for robust federated learning," *CoRR*, vol. abs/2002.00211, 2020. [Online]. Available: https://arxiv.org/abs/2002.00211

[20] Z. Gu and Y. Yang, "Detecting malicious model updates from federated learning on conditional variational autoencoder," in *2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2021, pp. 671–680.

[21] Y. Zhao, J. Chen, J. Zhang *et al.*, "PDGAN: A novel poisoning defense method in federated learning using generative adversarial network," in *Algorithms and Architectures for Parallel Processing*. Berlin, Heidelberg: Springer-Verlag, 2019, p. 595–609.

[22] Y. Zhao, J. Chen, J. Zhang *et al.*, "Detecting and mitigating poisoning attacks in federated learning using generative adversarial networks," *Concurrency and Computation: Practice and Experience*, vol. 34, no. 7, 2022.

[23] I. Goodfellow, J. Pouget-Abadie, M. Mirza *et al.*, "Generative adversarial networks," *Commun. ACM*, vol. 63, no. 11, p. 139–144, Oct. 2020. [Online]. Available: https://doi.org/10.1145/3422622

[24] E. Fatima, R. Talbi, S. Bouchenak *et al.*, "Towards mitigating poisoning attacks in federated learning," in *ComPAS'2021 : Parallélisme/ Architecture/ Système MILC-Lyon*, Lyon, France, Jul. 2021. [Online]. Available: https://hal.science/hal-03815244

[25] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," 2013. [Online]. Available: http://arxiv.org/abs/1312.6114

[26] K. Sohn, H. Lee, and X. Yan, "Learning Structured Output Representation using Deep Conditional Generative Models," in *Advances in Neural Information Processing Systems*, vol. 28. Curran Associates, Inc., 2015. [Online]. Available: https://bit.ly/3BfyaaX

[27] Y. Lecun, L. Bottou, Y. Bengio *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[28] T. H. Hsu, H. Qi, and M. Brown, "Measuring the effects of non-identical data distribution for federated visual classification," *CoRR*, vol. abs/1909.06335, 2019. [Online]. Available: http://arxiv.org/abs/1909.06335

[29] M. Fang, X. Cao, J. Jia *et al.*, "Local model poisoning attacks to byzantine-robust federated learning," *CoRR*, vol. abs/1911.11815, 2019. [Online]. Available: http://arxiv.org/abs/1911.11815

[30] Z. Wu, Q. Ling, T. Chen *et al.*, "Federated variance-reduced stochastic gradient descent with robustness to byzantine attacks," *CoRR*, vol. abs/1912.12716, 2019. [Online]. Available: http://arxiv.org/abs/1912.12716

[31] D. Rosendo, P. Silva, M. Simonin *et al.*, "E2clab: Exploring the computing continuum through repeatable, replicable and reproducible edge-to-cloud experiments," in *Cluster 2020 - IEEE International Conference on Cluster Computing*, Kobe, Japan, Sep. 2020, pp. 1–11. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02916032

[32] A. K. Sahu, T. Li, M. Sanjabi *et al.*, "On the convergence of federated optimization in heterogeneous networks," *CoRR*, vol. abs/1812.06127, 2018. [Online]. Available: http://arxiv.org/abs/1812.06127