REMINDER

REMINDER:

pRivacy-prEserving Machine LearnIng through secure managemeNt of Data's lifecyclE in distRibuted systems

Deliverable number: D4.1

Detailed report on vulnerabilities of existing approaches to an array of attacks

chist-era

FWF Austrian Science Fund

UKRI Engineering and Physical Sciences Research Council

uefiscdi Unitatea Executivă pentru Finanțarea Învățământului Superior, a Cercetării, Dezvoltării și Inovării

AGENCIA ESTATAL DE INVESTIGACIÓN

Project Acronym:	REMINDER
Project Full Title:	pRivacy-prEserving Machine LearnIng through secure manage-meNt of Data's lifecyclE in distRibuted systems
Call:	Security and Privacy in Decentralised and Distributed Systems (SPiDDS). 2022
Grant Number:	PCI2023-145989-2
Project URL:	https://ants.inf.um.es/en/reminder
Editor:	UWE
Deliverable nature:	Report
Dissemination level:	Public
Delivery Date:	30/11/2024
Authors:	UWE, UMU

Table 1: Project details.

Abstract

This report comprehensively analyzes the vulnerabilities associated with existing approaches to Federated Learning (FL) in the context of various security and privacy challenges. In the introduction, the report discusses the advantages of FL, such as data privacy and distributed learning, while highlighting key challenges, including exposure to multiple types of attacks. The taxonomy section categorizes common attack types, focusing on poisoning attacks, including data poisoning, model poisoning, free-riding attacks, and inference attacks, such as model inversion, membership, and reconstruction attacks. The report then explores potential countermeasures to mitigate these threats, particularly robust aggregation functions, malicious client detection mechanisms for poisoning attacks, and strategies for addressing inference and free-riding attacks. Finally, the conclusion emphasizes the importance of developing more resilient federated learning systems to safeguard against evolving attack techniques.

Table of Contents

List of Figures

List of Tables

1 Introduction

Federated Learning (FL) has emerged as a powerful technique that has captured the attention of numerous researchers [1, 2]. This approach was devised to address contemporary challenges concerning privacy and the adherence to General Data Protection Regulation (GDPR) principles [3], tackling the difficulties (in some cases the impossibility) of collecting, fusing, and using data that is in the form of isolated islands for training models [2]. FL involves multiple clients training a common Machine Learning (ML) model, each one training the model independently using its own data. These clients solely exchange the parameters derived from local training every certain training iteration with a central entity known as the server. Specifically, the server aggregates incoming updates by aggregating the parameters with a function, sending these values back to the clients for further local training. This decentralized approach allows clients to benefit from collaborative training without sharing their raw data, mitigating the need for centralized data storage. The basic workflow of FL is illustrated in Figure 1.

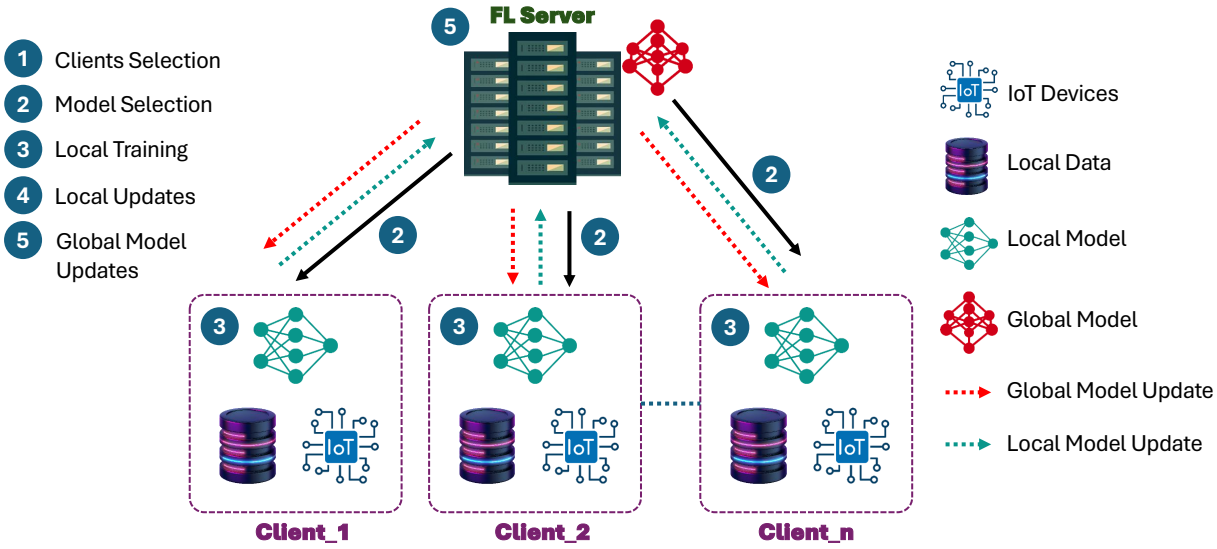

Figure 1: The basic concept of federated learning.

1.1 Advantages of Federated Learning

One of the crucial benefits of FL is its enhanced privacy and security benefits. The FL approach minimizes exposure to data breaches and protects user privacy by keeping data on each user's devices and only sharing model updates with a central server. Specifically, this keeps sensitive data on the user device, which is especially relevant, as concerns about privacy and data protection laws take precedence in the IT and Industrial domains. One evident benefit is that FL saves heavy data transfer to centralized servers. The model is trained in the local device, and only the updated model parameters will be sent to the FL server. This characteristic effectively reduced latency and maxed bandwidth utilize. This is especially effective for certain software applications that utilize big data, such as mobile networks or distributed sensor systems [4]. One additional advantage is the option to personalize AI models with local data. Every FL client can customize the model to address data features pertinent to that device, creating a more personalized user experience and accuracy. This feature is particularly beneficial in applications such as personalized recommendations, where individual user preferences can substantially impact the model's performance. Last, FL strongly aligns with data protection

policies, such as the General Data Protection Regulation (GDPR) [5]. These policies generally state that privileged, private information must stay within a certain ecosystem or users' personal devices. FL's architecture inherently supports compliance by minimizing the need to centralize sensitive information.

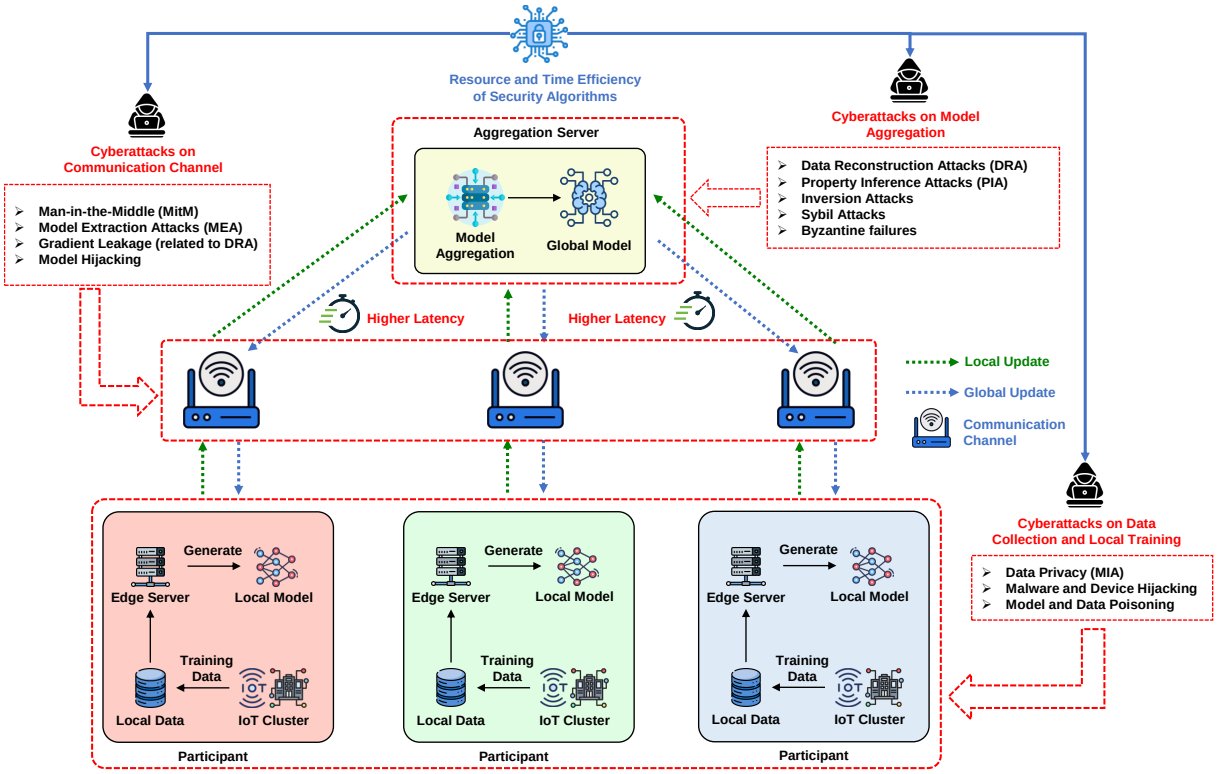

Figure 2: Security and privacy challenges in FL .

1.2 Key Challenges of Federated Learning

Despite multiple advantages, FL also brings several challenges. One of the most significant issues is communication overhead. Frequent data exchanges with the main server from multiple devices can utilize much bandwidth, leading to high computational costs. Data heterogeneity is also a very serious challenge. Devices often generate data that is not identically distributed (non-IID), leading to difficulties in model convergence [6]. The global model may struggle to generalize effectively when trained on data that varies widely across devices, potentially leading to reduced performance and accuracy. The devices' variability presents an extra layer of complexity. FL uses devices with varying degrees of computational power, battery life, and connectivity. This inconsistency leads to inadequate participation and slower training times because even low-intelligence devices will have difficulty performing complex computations or maintaining connectivity [6].

Although FL still reduces the direct sharing of data, it is vulnerable to several attacks. For instance, malicious participants could perform model poisoning, sending corrupted updates that can degrade the quality of the global model or introduce biases [7]. Aggregating model updates introduces additional complexity. Secure and efficient aggregation methods are required to combine updates from multiple devices while preserving privacy. Techniques such as federated averaging and differential privacy must be implemented carefully to ensure that the aggregation

process does not inadvertently expose sensitive information [8]. Some of the most significant security and privacy challenges in FL architecture are depicted in Figure 2.

1.3 Potential Solutions to Address the Security and Privacy Challenges in FL

To address security and privacy challenges in Federated Learning (FL), a comprehensive security solution must incorporate robust defense mechanisms across all stages of the FL pipeline. For safeguarding data collection and local training, techniques such as Differential Privacy (DP) and Secure Multi-Party Computation (SMPC) can effectively protect against Membership Inference Attacks (MIA) and prevent data leakage. Implementing Trusted Execution Environments (TEEs) and robust device authentication protocols can mitigate risks of malware, device hijacking, and model or data poisoning. To secure communication channels, end-to-end encryption, homomorphic encryption, and digital signatures can defend against Man-in-the-Middle (MitM) attacks, Model Extraction Attacks (MEA), Gradient Leakage, and Model Hijacking. Additionally, communication protocols should incorporate secure aggregation techniques to preserve data confidentiality. During model aggregation, employing Byzantine-resilient aggregation algorithms, such as Krum and Bulyan, can counteract Sybil attacks, Byzantine Failures, and model poisoning. Furthermore, Differential Privacy and Homomorphic Encryption can protect against Data Reconstruction Attacks (DRA), Property Inference Attacks (PIA), and Inversion Attacks by limiting the exposure of sensitive information. Combining these strategies within a layered security framework enhances the resilience of FL systems against evolving cyber threats, ensuring data privacy and model integrity.

The key objective of this document is to provide a detailed analysis of security and privacy attacks in FL architectures and promising solutions to overcome these issues. The next section elaborates on the taxonomy of attacks in FL.

2 Taxonomy

This section presents the taxonomy of attacks in Federated Learning, distinguishing the two main types: inference attacks and poisoning attacks. Inference attacks aim to extract sensitive information from the model updates, while poisoning attacks compromise the training process by intentionally injecting malicious data or altering model parameters. Figure. 3 shows the classification of the different attacks.

2.1 Poisoning attacks

Poisoning attacks in FL are intended to harm the training phase by introducing maliciously modified data or weights to undermine the model performance [9]. These attacks are usually represented in two forms: data poisoning attacks and local model poisoning attacks. In Figure. 4, we show the different types of poisoning attacks, and the specific attacks considered.

2.1.1 Data poisoning attacks

Data poisoning attacks occur when an adversary deliberately modifies or manipulates the training data in order to degrade a model's performance or direct its outcomes toward specific, malicious goals. Data poisoning attacks can by applied in two different forms [10]: Sybil and backdoor. The **Sybil attack** involves a malicious participant attempting to amplify the impact of data poisoning by generating multiple false identities of legitimate participants, all containing poisoned data. **Backdoor attack** is a threat assessment framework that exploits the distributed nature of FL to manipulate a subset of training data by injecting adversarial triggers in a distributed manner. Data poisoning attacks typically fall into two main categories [11]. On one hand, **clean-label** attacks subtly alter the input samples without changing their original labels, making them harder to detect. On the other hand, **dirty-label** attacks involve modifying both the training samples and their labels, creating more obvious discrepancies but potentially causing significant damage if not quickly identified. By corrupting the data used to train ML models, data poisoning attacks pose a substantial threat to the integrity and reliability of AI systems, underscoring the need for robust preventive and detection strategies.

Among data poisoning attacks, dirty-label methods tend to be more effective, as they involve not only altering the input samples but also changing their labels, thereby creating significant discrepancies that rapidly undermine model performance. In our future work, we plan to employ label flipping [12], a common form of dirty-label attack—to further investigate and test defense mechanisms against these types of adversarial threats. **Label flipping** is a type of data poisoning attack in which the labels of a client are changed randomly from one specific class to another specific class, remaining the training samples untouched. A visual example of this is shown in Figure. 5.

2.1.2 Model poisoning attacks

Local model poisoning attacks target modifying the weights generated through the local training process, so that clients transmit corrupted weights to the aggregator. They encompass two classes: untargeted and targeted poisoning attacks [9]. **Untargeted attacks** constitute the basic form, seeking to minimally modify the global model to provide incorrect predictions for different samples. In contrast, **targeted attacks** are intricate, introducing a multi-task problem

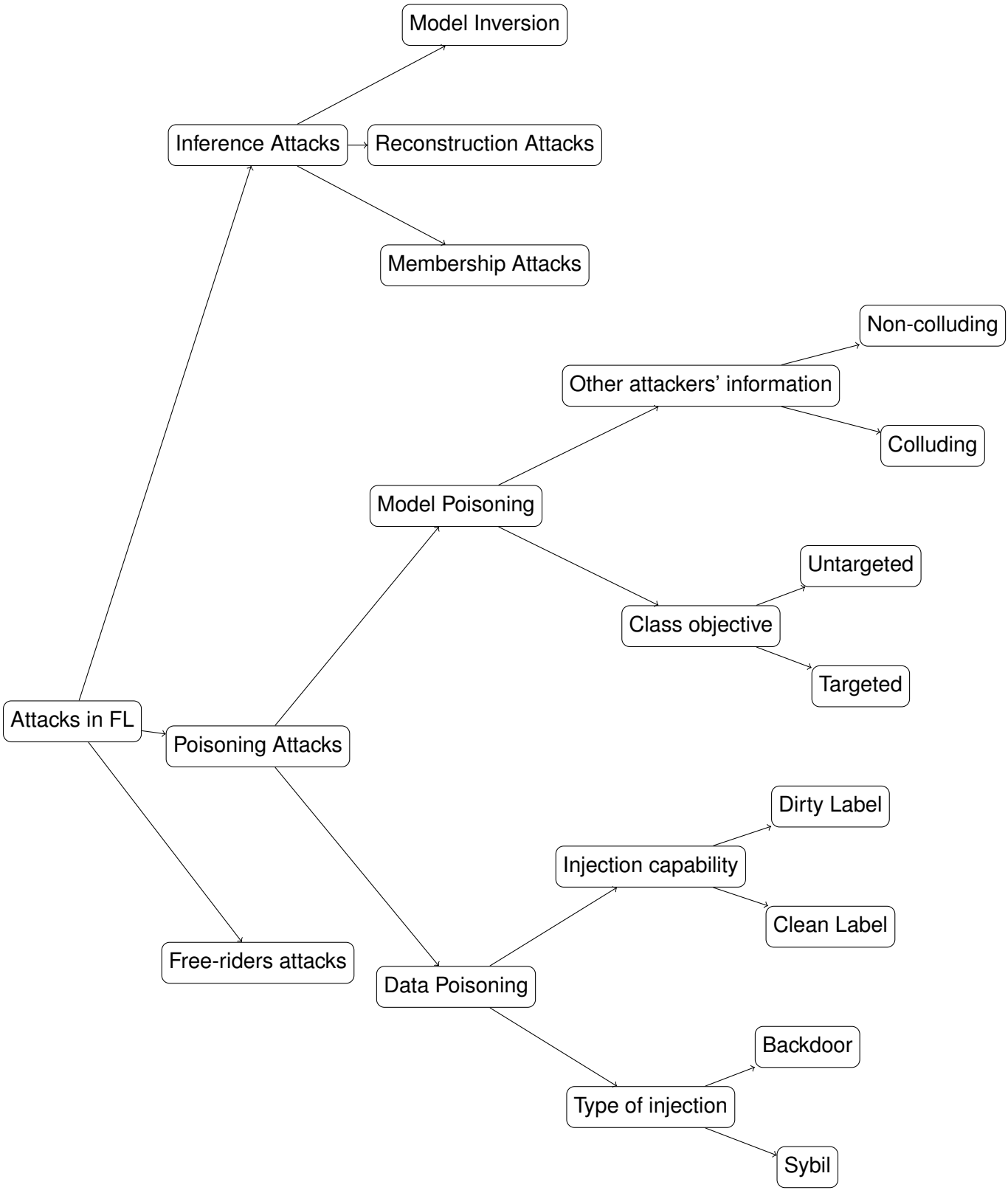

Figure 3: Taxonomy of the attacks in FL

to manipulate the victim's predictions on specific samples while maintaining functionality on benign samples (unlike random alterations in untargeted attacks).

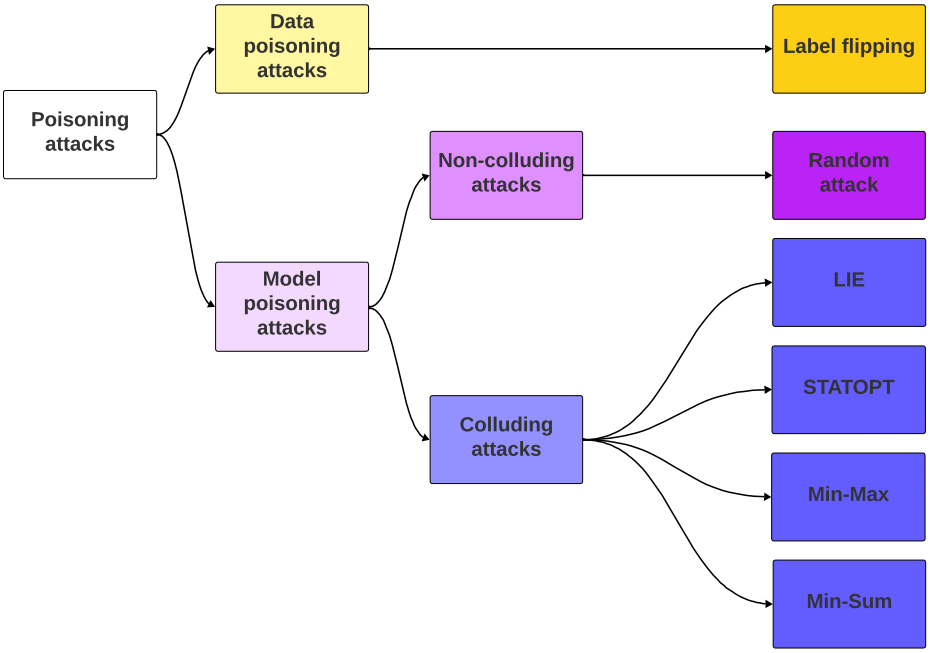

Figure 4: Description of considered attacks.

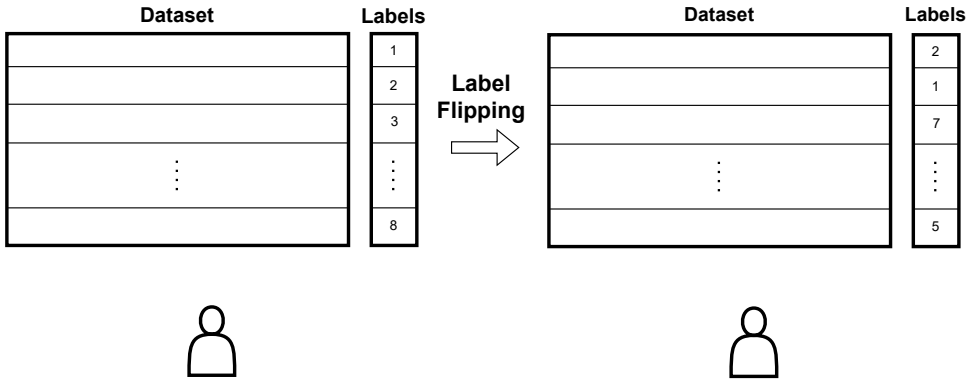

Figure 5: Description of label flipping attack.

Recent works such as [13] argue that data poisoning attacks can essentially be perceived as local model poisoning by computing their corresponding weight modifications. Moreover, empirical evidence suggests that local model poisoning attacks yield a more significant impact than dataset alterations [13]. Additionally, the research in [14] underscores the challenge in detecting untargeted poisoning attacks compared to their targeted counterparts. In this direction, we will focus on untargeted attacks. A considered attack in our future works will be the random attack. The **random attack** consists of substituting the weights by random values from a specific distribution.

Untargeted poisoning attacks can be further categorized based on whether malicious clients conspire with each other or not, as described in [15]. In these cases, the malicious clients first proceed with standard model training independently, as usual. However, after training, they combine their weight updates by averaging them and add a penalty function to intentionally harm the aggregated model. In Figure. 6 it is shown a scheme of how these colluding attacks work. The attacks considered in this case are:

1. **LIE:** LIE (Little Is Enough) [16] is a type of attack where it is added small amounts of noise to each dimension of the average of their updates. Specifically, the malicious clients calculate the mean (∇^M) and standard deviation (σ^M) of their weights after the training of that round. Then, the malicious clients compute a coefficient z based on the number of benign and compromised clients. Finally, the new weights which will be sent to the server will be the average ∇^M plus $z\sigma^m$
2. **STATOPT:** STATOPT (static Optimization) [17] computes of the average of the malign clients weights (∇^M) and computes a static malicious direction $\omega = -sign(\nabla^M)$. Hence, the final malign weights send to the server are $-\gamma\omega$, where γ is a variable chosen by the attackers.
3. **Min-max:** The min-max attack was proposed in [15]. In essence, it consists of maximising the distance between the malign weights and benign weights, without the robust aggregation function noticing them. In particular, the malign clients will calculate the average of their weights (∇^M). Additionally, the malicious clients calculate the direction (θ) and magnitude (γ) and add them to ∇^M to generate the malign weights which will be sent to the server. θ is any malicious direction in the space of gradients that the adversary can use to perturb ∇^M and find the malicious gradients W_m. The distance perturbed by γ is calculated as an optimization problem, where the objective is to find the γ that satisfies that the maximum distance between this perturbation and the weights of the malicious clients cannot be higher than the maximum distance between the weights of the malicious clients.
4. **Min-sum:** The min-sum attack is also proposed in [15]. It is similar to min-max in the sense that the same parameters are considered, the average ∇^M, the direction θ, and the distance γ. However, in this case, the optimization function instead of taking the maximum distance, is to take the γ that maximizes the sum of the distance between the point $\nabla^M + \gamma\theta$ and the weights of the malicious clients, subject to the total sum of the distance between the weights of the malicious clients.

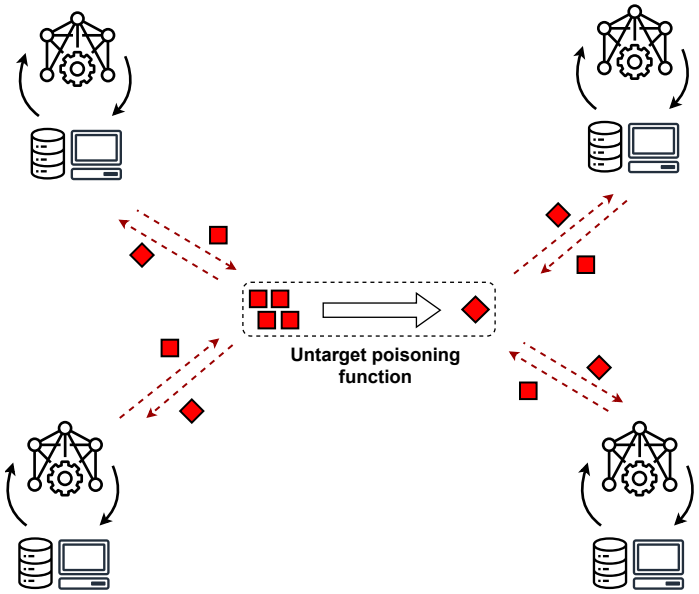

Figure 6: Description of an untargeted attack where the attackers collude.

2.1.3 Free-riding attacks

In FL, free-riding is a deceptive attack in which a participant attempts to take advantage of the benefits of the global model without investing adequate resources in the training process. Essentially, a free-rider selects only a small subset of their actual dataset for training or, in some cases, uses random noise instead. By doing so, they conserve their computational resources while still reaping the rewards of the collaborative training process. This behavior unfairly shifts the burden onto honest participants, who are then required to contribute more resources to compensate for the free-rider's lack of effort. As a result, the overall performance of the model is compromised due to the poor-quality data contributed by the free-rider, ultimately diminishing the effectiveness and reliability of the global model. This attack's impact would be more in a smaller landscape FL environment as the absence of client participation can negatively impact global model training. As the probability of this attack is lower, the severity of this attack is medium.

2.2 Inference attacks

Inference attacks in FL aim to extract sensitive information from the shared updates or the global model without direct access to the underlying data. They exploit the gradients or model parameters exchanged during the training process to infer private data characteristics. The following provides a detailed breakdown of different categories of inference attacks.

2.2.1 Model inversion

Model inversion attacks aim to reconstruct input data from model outputs or gradients by leveraging knowledge of the model architecture and its parameters [18]. The fundamental idea is to invert the model's output to approximate the corresponding input. This is particularly concerning in scenarios where the model is over-parameterized or overfitted, as it may inadvertently memorize training data.

Consider a neural network model $f(\theta, x)$, where θ represents the model parameters, and x is the input. Given the model output $y = f(\theta, x)$, the attack seeks to reconstruct x by solving an optimization problem:

$$\hat{x} = \arg\min_{x} \mathcal{L}(f(\theta, x), y) + \lambda R(x), \tag{1}$$

where:

- $\mathcal{L}$ is the loss function quantifying the discrepancy between the model's output and the observed output y. Mean Squared Error (MSE) is commonly used for regression tasks, while cross-entropy loss is used for classification.
- $R(x)$ is a regularization term that enforces plausibility in the reconstructed input. For instance, it can be a Total Variation (TV) norm to promote smoothness or a norm constraint to restrict the input space.
- λ is a hyperparameter balancing the trade-off between the loss and the regularization term.

The reconstruction process is typically carried out using gradient descent. Starting from a random initialization x', the input is iteratively updated as follows:

$$x' \leftarrow x' - \eta \nabla_{x'} \left[\mathcal{L} \left(f \left(\theta, x' \right), y \right) + \lambda R \left(x' \right) \right] \tag{2}$$

where η is the learning rate. The gradient of the loss with respect to the input propagates through the layers of the neural network, effectively reversing the computation path. This enables the adversary to approximate the training sample corresponding to the observed output.

Model inversion attacks pose significant privacy risks, particularly in scenarios involving facial recognition systems or medical image classifiers, where sensitive information can be reconstructed from model outputs [19]. For instance, given the class label of a facial recognition model, an attacker can approximate the facial features corresponding to that label, thereby compromising user privacy.

2.2.2 Memberships attacks

Membership inference attacks aim to determine whether a specific data point was part of the model's training set. In FL, where individual clients contribute updates to the global model, this attack exploits the model's tendency to behave differently on training versus non-training samples, especially when the model is overfitted [20, 21].

An adversary queries the model with a target input x and observes the output $s = f(\theta, x)$. By analyzing the confidence scores or loss values, the attacker infers the membership status of the input. The attack leverages the fact that models often yield higher confidence and lower loss for training samples compared to unseen data.

$$M(x, y, s) = \begin{cases} 1, & \text{if } x \text{ is a member} \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

The classifier is trained on the following features:

- Confidence scores: $s = f(\theta, x)$
- Loss values: $\mathcal{L}(f(\theta, x), y)$
- Gradient norms: $|0 \nabla_\theta \mathcal{L}| \, 0$

A common approach is to compare the confidence score against a threshold:

$$M(x, y, s) = \begin{cases} 1, & s > \tau \\ 0, & s \leq \tau \end{cases} \tag{4}$$

where τ is a threshold determined empirically. This approach is particularly effective in overfitted models, where the output distribution for training samples differs from that of non-training samples.

Membership inference attacks reveal whether an individual's data was used in model training, posing severe privacy risks in sensitive domains such as healthcare and finance. For instance, confirming that a patient's data was used to train a medical diagnosis model indirectly reveals their medical condition.

2.2.3 Reconstruction attacks

Reconstruction attacks aim to reconstruct the entire input data from model gradients or updates shared during FL. Unlike model inversion attacks, which focus on reconstructing a single input, reconstruction attacks target the full dataset or its statistical properties [22]. This is particularly relevant in federated settings where clients share gradients with the central server.

In FL, each client computes gradients based on their local data. For a client i, the gradients are calculated as:

$$g_i = \nabla_\theta \mathcal{L}\left(f\left(\theta, x_i\right), y_i\right) \tag{5}$$

An adversary observing these gradients seeks to reconstruct the corresponding inputs x_i. The reconstruction problem is formulated as:

$$\hat{x} = \arg\min_x \left|0g_i - \nabla_\theta \mathcal{L}(f(\theta, x), y)\right| 0^2, \tag{6}$$

where the objective is to find inputs whose gradients closely match the observed gradients.

The reconstruction is achieved using a gradient matching technique, where a dummy input x' is iteratively updated as follows:

$$x' \leftarrow x' - \eta \nabla_x \left|0g_i - \nabla_\theta \mathcal{L}\left(f\left(\theta, x'\right), y\right)\right| 0^2 \tag{7}$$

where η is the learning rate. This approach effectively reverses the gradient computation process to approximate the original input data.

Reconstruction attacks pose a significant privacy threat in collaborative learning environments, where gradients are shared among participants. For example, in a federated learning system for medical diagnosis, an attacker could reconstruct sensitive patient records from shared gradients, leading to data breaches and privacy violations.

3 Countermeasures

In this section, we describe the current solutions to tackle the attacks described above. In Figure. 7, we describe different countermeasures for each type of attack.

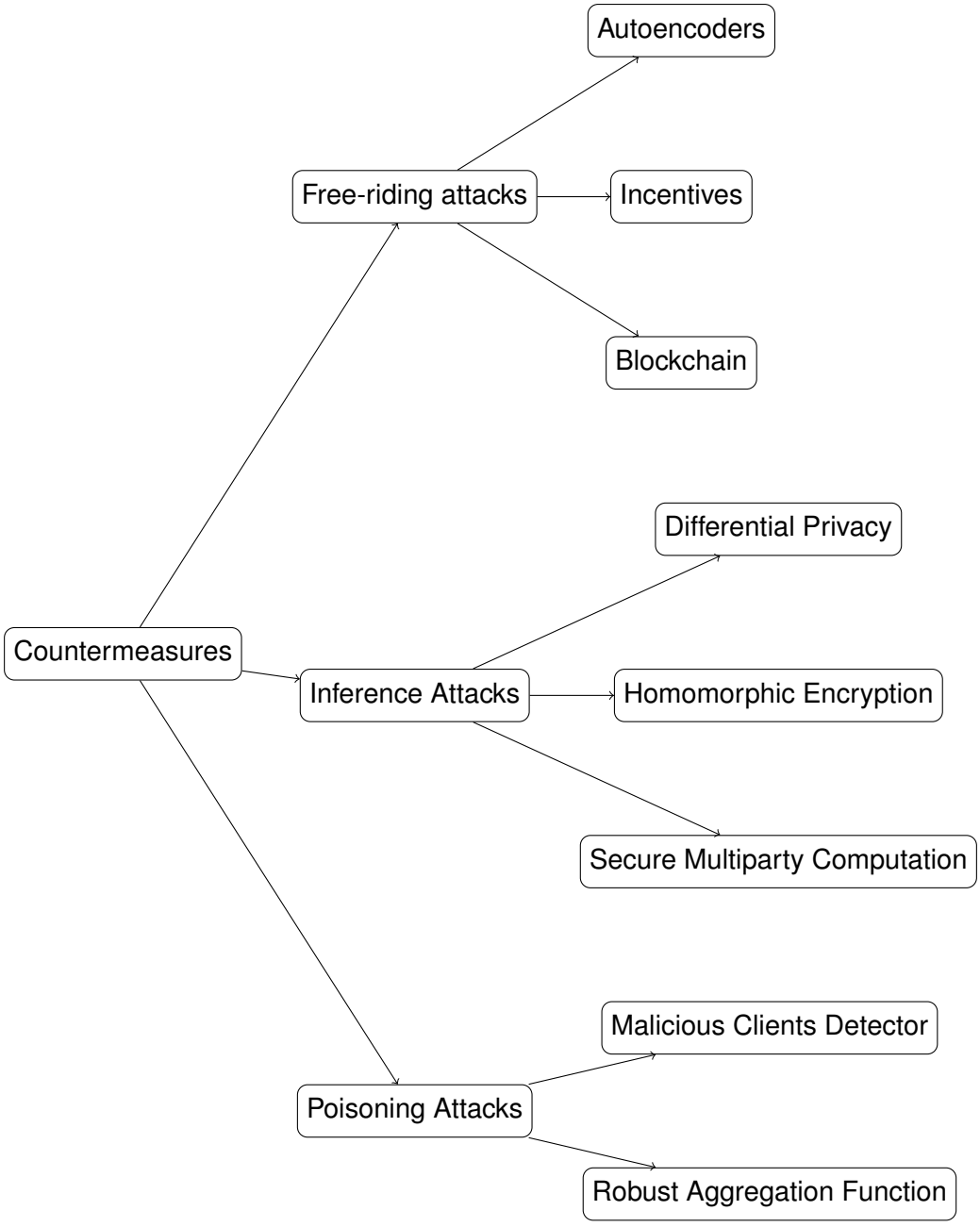

Figure 7: Description of the different countermeasures in FL

3.1 Poisoning attacks

In the first part of this section, we describe the assumptions made and the elements involved in our FL scenario. Then, we detail the methods and algorithms created to defend against these attacks.

We provide a visual explanation of the considered FL scenario, shown in Figure. 8. Regarding our FL scenario, it is composed of K clients and one aggregator. Although the proposed scenario only considers a single aggregator, it is important to note that our approach can be used in decentralized or hierarchical settings where multiple aggregators are deployed [23]. Our threat model assumes the existence of a number M of compromised clients or byzantine attackers. In this case, M will be less than half of the total number of clients, i.e., $M < K/2$. Following the scheme of Figure. 8, these malicious clients, instead of sending benign weights (represented by light green circles), will send malicious weights (represented by red diamonds), produced by poisoning attacks, in order to corrupt the aggregated weights made by the aggregation function, and produce non fully benign weights, i.e., to produce a deviation in the convergence of the benign weights. For producing the corrupted weights, the byzantine clients will use either a data or a model poisoning attack.

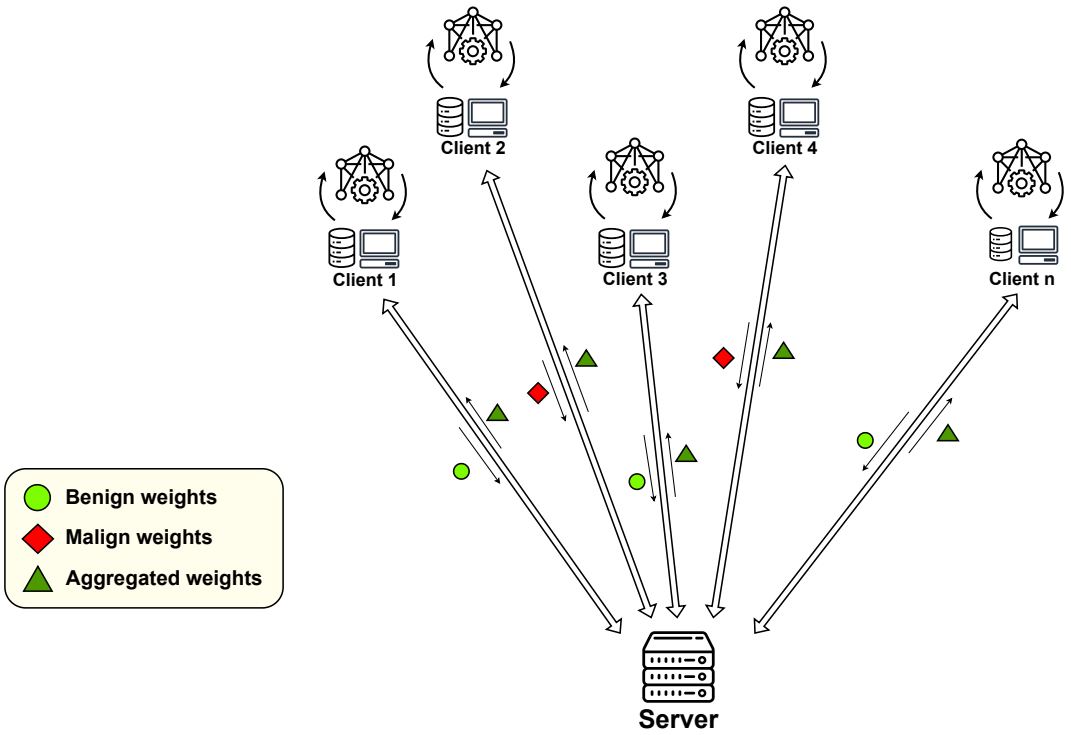

Figure 8: Description of our scenario considering byzantine clients.

The assumptions made in our threat model, we consider the systematization of threat models for poisoning attacks in FL proposed by [24]. This system describes different aspects related to the characteristics of the attackers, including their objectives, knowledge, and capabilities:

- **Objective of the adversary**: for byzantine clients, the goal is to reduce the model's accuracy either by manipulating the weights generated from local training in each FL round or the dataset used. Therefore, our considered attacks, either data poisoning or local model poisoning, are *indiscriminate* [16], which means their aim is to cause a general misclassification of the data used, without targeting a specific class or set of samples.
- **Knowledge of the adversary**: in the case of the local model poisoning attacks, the attackers only have access to the model updates produced by the client they control, and the other malign clients' models, but not weights of the benign clients. For the data poisoning attack, the byzantine clients have no information about other clients, only the

dataset of the client. Therefore, following the categorization described by [24], we assume *whitebox* knowledge regarding the models produced by the clients in each training round. Moreover, although it could be considered in future work, we assume that the attackers do not have information related to the aggregation function used.

- **Capabilities of the adversary**: for the data poisoning attack, the malign clients are capable of changing the local data themselves at the beginning of the federated training. The rest of the considered attacks are based on poisoning the models produced by the clients in each training round. Access can be triggered by direct access to the device itself, or through attacks such as Man-in-the-middle where the attacker receives the updates and modifies them before they are sent to the aggregator in each training round. Furthermore, these attacks follow an *online* mode, i.e., these attacks are launched in every training round. In addition, the attackers have the ability to collude with each other by exchanging their local updates to achieve a greater impact.

Now we describe the way to defend against these attacks are either robust aggregation functions or creating algorithms for identifying malicious clients.

3.1.1 Robust aggregation functions

Robust aggregation functions are meant to resist the impact of corrupted weights by identifying and discarding such data. For example, [25] employs median and trimmed mean to protect their federated process using an intrusion detection dataset for IoT, although authors do not consider sophisticated colluding attacks, like the min-max. The description of median and trimmed mean is provided by [26], but the authors do not offer experimental results on the resilience of these functions to Byzantine attacks. Additionally, [27] implements a variant of Krum called multi-Krum, in which the update generated by the aggregator in each round is based on averaging several updates provided by the clients. While collusion attacks are considered, these are related to privacy aspects [28].

Moreover, other studies consider variations of existing aggregation functions or define new approaches to address different poisoning attacks. In this direction, [29] uses the geometric median instead of the median as employed in FedMedian. Specifically, the authors define RFA, a robust aggregation algorithm based on the weighted geometric median to defend the system against clients' corrupted weights considering aspects of privacy and efficiency. Moreover, the authors of [30] extend this technique of the geometric median by incorporating additional weights for those that are close to each other based on their formula for measuring skewness. In both works, they adapted their method to fit non-IID scenarios by adding personalization techniques [31]. Next, FoolsGold [32] and FLTrust [33], instead of applying a function which does not take into account outlier values, they change the coefficients of the different clients in the aggregation. In particular, FoolsGold measures the similarity of the historical vectors of the clients, and then, it apply a technique called pardoning, based on the maximum value of each round, to change the values of the similarity matrix, and then it calculates the coefficients of the clients. On the other hand, FLTrust calculates each client's trust score by applying ReLU to the cosine similarity of the client's model with the server's model. It then normalizes these scores to mitigate malicious influence, and finally aggregates the normalized weights into the global model.

Next, we describe in detail the most important robust aggregation functions:

1. **FedMedian:** It represents a different approach compared with FedAvg by employing a median computation rather than averaging weights. This variant computes the median in

a coordinate-wise manner, evaluating weight values individually.

$$W = Median(W^1, \dots, W^K) \tag{8}$$

2. **The trimmed mean:** It constitutes a distinct approach to mean computation. This method involves selecting a value $n < \frac{K}{2}$, where K represents the number of clients. Subsequently, in a coordinate-wise manner, the server eliminates the lowest n values and the highest n values, computing the mean from the remaining $K - 2n$ values. The selection of n aims to eliminate outliers and enhance the system's capability against potential adversarial behavior from n malicious clients, thereby enhancing the system's robustness against up to 50% of malicious entities. Denoting this resulting set after removing the 2n elements as I:

$$W = \sum_{n}^{K-n} \frac{W^i}{|I|}, \tag{9}$$

3. **Krum:** It derives from the Krum function (Equation 10). This function operates on the clients' weights W^k, selecting the weight with the lowest sum among their $K - f - 2$ nearest neighbors. Here, K denotes the total number of clients, f represents a server-determined constant indicative of the count of malicious clients, and Γ^i_{K-f-2} denotes the set of nearest neighbors for client k. Subsequently, the server transmits this selected value to the remaining clients.

$$Krum(W^1, W^2, \dots, W^K) = Min\{(\sum_{j \in \Gamma^i_{K-f-2}} ||W^i - W^j||)_{i=1}^K\} \tag{10}$$

4. **FoolsGold:** The method involves obtaining the historical vector Δ_i for each client and calculating a similarity matrix using cosine similarity, with the consideration of weighted features. For each row i of this matrix, representing a client i, the maximum value v_i is calculated. Subsequently, for each pair of clients i and j, if v_j surpasses v_i, the similarity between clients i and j is adjusted by multiplying it by v_i/v_j, a process they referred to as "pardoning". This modification reweights the cosine similarity, enhancing the distinction between honest updates that might erroneously appear malicious due to the stochastic gradient descent's variance. Following this, α_i values are calculated to weight the average among the clients' weights, which will be sent to the clients. To calculate α_i for each client i, 1 is subtracted from the maximum value in row i and the result is divided by the maximum value among all αs. Finally, to amplify the significance of benign clients, a logit function centred on 0.5 and neperian based is employed. Consequently, the server computes the average of clients, taking into account these α_i, i.e., the weights at round r are uploaded by the equation:

$$W_r = W_{r-1} + \sum_{1}^{K} \alpha_i * \Delta_i \tag{11}$$

5. **FLTrust:** FLTrust first calculates the trust score TS_i of each client i by calculating the ReLU function to the cosine similarity between the clients' models W^i and the server model W^0. Once this vector has been calculated, the client weights are normalised to

reduce the impact of malicious client attacks into $\bar{W}^i$. Finally, the weights are aggregated by the following equation:

$$W = \frac{\sum_1^n TS_i \bar{W}^i}{\sum_1^n TS_i} \tag{12}$$

6. **FedRDF:** FedRDF is a robust aggregation function created in [34] that can adapt to any situation. First, FedRDF uses the Kolmogorov-Smirnov Test [35] to measure if there is malicious presence. In particular, from the distribution of all clients, FedRDF takes a subset of them and with the Kolmogorov-Smirnov Test checks whether the distribution of this subset and the remaining set are the same. This process is repeated several times and if the number of times the test has failed is greater than a threshold, FedRDF assumes that there are malicious clients. Depending on the result of this test, if the test has failed, FedRDF employs the Fast Fourier Transform (FFT) to aggregate the weights, and otherwise, it employs FedAvg. In particular, the FFT projects the weights to the frequency domain, and we select those weights that achieve the highest frequency.

3.1.2 Malicious clients detector

Several studies aim to identify and isolate malign clients for subsequent aggregation of benign updates using standard functions. Notably, these works employ diverse techniques to detect malicious clients to protect the global model. For instance, [36] utilizes a Variational Autoencoder to classify benign and malign clients. Nevertheless, this approach requires the server to access a clean validation dataset, operates effectively only when benign updates are distinguishable from malign ones, and substantially increases training time due to the need for ML model pre-training before FL training. A common technique across the majority of studies involves a clustering phase utilizing various clustering methodologies to classify benign and malign clients. For example, [37] introduces **FLDetector**, which leverages the Hessian matrix of weights to predict subsequent client weights. By measuring the disparity between the real and the predicted weights of each client, FLDetector utilizes gap statistics to determine the number of client clusters. Subsequently, if the count exceeds one, they employ k-means to partition clients into two clusters: benign and malign ones. However, the cumulative computational cost of these steps may make the implementation in restricted devices unfeasible.

Another prevalent approach in current works involves measuring model similarity using cosine similarity. In [38], the authors introduce **Sentinel**, a defense strategy against poisoning attacks in FL. Utilizing cosine similarity, Sentinel filters out clients exhibiting lesser similarity compared to others. Subsequently, the authors calculate the loss of remaining clients, eliminating those with higher loss. Finally, they protect the system from those clients which evade the previous filters by normalizing their gradients. Nevertheless, this technique protects the system against gradients with high norm, not for gradients with opposite directions. Moreover, Sentinels's efficacy against sophisticated attacks remains unverified in their work. Studies like [39,40] suggest that direct application of cosine similarity may yield similar results between benign and malign clients. Hence, [40] modifies the input for cosine similarity in the method they proposed, **MAB-RFL**. This approach filters out potential attackers based on past attacks in previous rounds. Additionally, they filter between Sybil and non-Sybil attacks [41]. For Sybil attack they create a graph based on the cosine similarity to detect Sybil clients (when an attacker generates multiple fake data points or models to manipulate the training process or outcomes), and for non-Sybil attacks, they cluster the clients depending on the cosine similarity of the gradients. However, the assumption of attackers attacking every round undermines the effectiveness of this initial filter. Additionally, normalization techniques in [38] and [40] face convergence delays, disrupting

the value of benign clients.

Lastly, in [39], the authors introduce **SignGuard**, a framework devised to detect malign clients. They employ a two-filter approach: initially, by utilizing gradient norms, they eliminate those exceeding the median value. Subsequently, they employ mean statistics to cluster clients into malign and benign categories. Additionally, they extend their methodology by incorporating cosine similarity into their analysis. The work [42] presents **LoMar**, an algorithm created to defend the FL process against poisoning attacks. Their algorithm is divided into two parts, first, they divide the model weights using the k nearest neighbors and then use their density using kernel density estimation (KDE) to calculate what they call the client's malicious factor. In the second part, they set a threshold to assess which clients are malicious. Finally, **FedDMC** [43] reduces the dimension of the clients' weights to increase the dissimilarity between malign and benign clients. Next, they cluster the clients using a binary tree classification to separate between benign and malign. Finally, FedDMC calculates a trust score based on the times the clients were chosen benign or malign to make the final decision.

3.2 Inference attacks

The growing concerns surrounding inference attacks in FL have prompted various researchers to explore and propose countermeasures to enhance the security and privacy of this decentralized learning paradigm A number of state-of-the-art contributions focus on addressing different types of inference attacks, including category inference, source inference, property inference, and label inference, as well as general threats posed by malicious participants. The work in [44] identifies category privacy as a critical vulnerability, especially in the context of FL with non-IID data. Secure aggregation protocols often fail to protect category privacy, exposing sensitive information about data categories. The authors propose a novel mitigation approach involving a differential selection strategy coupled with de-noising techniques to address this issue. Additionally, they explore counter-detection methods to make inference attacks less conspicuous, thereby strengthening privacy-preserving mechanisms in FL. This work underscores the need for enhanced protection against inference attacks targeting category privacy. In [45], the focus shifts to source privacy, a less-discussed vulnerability in FL. The study introduces the concept of a source inference attack (SIA), where an adversarial server can deduce which client contributed specific training data. This type of attack is particularly insidious, as it can occur without deviating from standard FL protocols. The authors employ Bayesian inference techniques to identify the origin of the training data, demonstrating that various FL frameworks, whether sharing gradients, model parameters, or predictions, inadvertently expose source information. Their study highlights the connection between attack success and local model overfitting, calling for developing more robust privacy-preserving mechanisms to counteract the risks associated with source inference. Property inference attacks, which involve inferring sensitive properties of training data unrelated to the learning objective, are explored in [46]. The authors introduce a novel poisoning-assisted property inference attack that exploits periodic model updates to detect shifts in data distribution. This approach distorts the global model's decision boundary, forcing benign participants to inadvertently reveal more information about sensitive data properties. The findings emphasize the vulnerability of FL to property inference, particularly in dynamic environments. To mitigate this risk, the authors advocate for developing stronger defenses to safeguard against such attacks, ensuring the integrity and confidentiality of sensitive data properties.

To address the inefficiencies of existing secure aggregation protocols, the work in [47] presents SAFELearn, a generic framework designed to mitigate inference attacks, particularly those

perpetrated by malicious aggregators. SAFELearn is efficient and scalable, requiring only two communication rounds per training iteration and avoiding expensive cryptographic operations on clients. Additionally, it operates without a trusted third party and is resilient to client dropouts. By incorporating multi-party computation (MPC) or fully homomorphic encryption (FHE), SAFELearn significantly improves the efficiency of secure aggregation while enhancing privacy protection in FL. The challenge of malicious updates, which can compromise the accuracy of the global model, is addressed in [48] with the introduction of FedXPro, a Byzantine client detection algorithm for FL in wireless IoT systems. FedXPro utilizes a combination of predictive coding/biased competition-divisive input modulation (PC/BC-DIM) neural networks and geometric median (GM) techniques to detect malicious clients. By leveraging Bayesian inference to distinguish legitimate clients from adversaries, FedXPro improves detection accuracy and ensures faster convergence, thereby maintaining the integrity of the global model and strengthening FL security while minimizing performance trade-offs.

In the context of vertical federated learning (VFL), the study in [49] addresses label inference attacks, a growing security threat in this decentralized learning paradigm. To defend against passive label inference attacks, the authors propose the FL Similar Gradients (FLSG) scheme, which replaces original gradients with randomly generated ones that preserve similar dimensions and control cosine distance. Unlike differential privacy, FLSG offers a more efficient defense with lower computational overhead while maintaining model utility. This approach strengthens VFL security by preventing adversaries from extracting private label information and ensuring confidentiality in collaborative learning environments. [50] evaluates the privacy vulnerabilities of federated learning-based network intrusion detection systems (NIDSs). The authors introduce two novel privacy evaluation metrics—privacy score and evasion rate—to assess the effectiveness of privacy defenses. In response to these vulnerabilities, they propose FedDef, an optimization-based input perturbation defense strategy that balances privacy protection with model utility. By minimizing gradient distance while maximizing input distance, FedDef significantly improves the robustness of FL-based NIDSs against privacy attacks, outperforming existing defenses and enhancing security without substantial loss in detection accuracy.

3.3 Free-riding attacks

Several strategies have been suggested to address the issue of free-riding in FL. A widely proposed solution involves leveraging blockchain technology to monitor participant updates and verify their contributions. In [51], an FL framework is introduced that incorporates blockchain to facilitate the exchange and verification of local model updates, effectively addressing the free-riding problem. Within this system, each participant trains their local model and submits it to a corresponding blockchain miner, receiving rewards proportional to the amount of data they process. This structure discourages free-riding and promotes active contribution to the learning process. A similar blockchain-based model is explored in [52], aiming to enhance data confidentiality, computational auditability, and participant incentives in FL. However, implementing blockchain technology requires maintaining miners to sustain the network. Additionally, blockchain consensus mechanisms like Proof-of-Work (PoW) can result in significant delays in information exchange, making them less suitable for dynamic FL models.

Another method to mitigate free-riding relies on incentive mechanisms that compensate participants based on their contributions to collaborative training [53]. The incentive system ensures that participants are rewarded for their efforts, with Richardson et al. defining participant influence as the impact of their contributions on the FL model's loss function. Incentives are

distributed in proportion to this influence. Furthermore, research presented in [54] investigates free-riding attacks in FL settings and introduces an improved anomaly detection technique using autoencoders to identify free-riders effectively.

4 Conclusions

This report presents a comprehensive exploration of the vulnerabilities in federated systems, focusing particularly on poisoning attacks. FL, despite its distributed and privacy-preserving nature, remains susceptible to attacks that compromise model integrity by either poisoning data or manipulating local model updates. These types of attacks have the potential to degrade model performance, introduce biases, or cause targeted misclassifications, which highlights the critical need for robust defense mechanisms. Our analysis classifies these poisoning attacks into two categories: data poisoning, which can take the form of clean-label or dirty-label attacks, and model poisoning, which can be untargeted or targeted. The decentralized architecture of FL exacerbates these vulnerabilities, as the server responsible for aggregating model updates lacks direct visibility into the local training processes of the clients. This lack of oversight creates opportunities for malicious clients to interfere with the training phase and influence the final aggregated model updates.

To mitigate these vulnerabilities, several countermeasures have been categorized into two main approaches. The first approach involves the use of robust aggregation functions, including methods such as FedMedian, Trimmed Mean, Krum, and FedRDF. These functions aim to filter or reduce the impact of malicious client updates on the global model. Their effectiveness varies depending on the complexity of the attack, particularly when attackers coordinate to evade these defenses. The second approach focuses on the detection and isolation of malicious clients before the aggregation process. Techniques in this category employ clustering methods, gradient similarity analysis, and anomaly detection frameworks such as FLDetector, Sentinel, and SignGuard. These methods enhance model robustness but may face challenges regarding computational efficiency, especially in environments with resource-constrained devices.

Our findings emphasize the necessity of hybrid defense strategies that integrate robust aggregation methods with effective malicious client screening. This dual approach helps mitigate the risks posed by poisoning attacks more comprehensively. Additionally, future research should address evolving challenges, including adaptive strategies employed by attackers, the scalability of defense mechanisms, and their compatibility with real-world FL implementations. Protecting FL systems from adversarial threats is crucial to ensuring the reliability and security of distributed machine learning applications, particularly those operating under stringent privacy and data protection requirements.

References

[1] 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.

[2] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.

[3] Michelle Goddard. The eu general data protection regulation (gdpr): European regulation that has a global impact. *International Journal of Market Research*, 59(6):703–705, 2017.

[4] Tuo Zhang, Lei Gao, Chaoyang He, Mi Zhang, Bhaskar Krishnamachari, and A Salman Avestimehr. Federated learning for the internet of things: Applications, challenges, and opportunities. *IEEE Internet of Things Magazine*, 5(1):24–29, 2022.

[5] Zahra Abbas, Sunila Fatima Ahmad, Madiha Haider Syed, Adeel Anjum, and Semeen Rehman. Exploring deep federated learning for the internet of things: A gdpr-compliant architecture. *IEEE Access*, 2023.

[6] Hangyu Zhu, Jinjin Xu, Shiqing Liu, and Yaochu Jin. Federated learning on non-iid data: A survey. *Neurocomputing*, 465:371–390, 2021.

[7] Alberto Blanco-Justicia, Josep Domingo-Ferrer, Sergio Martínez, David Sánchez, Adrian Flanagan, and Kuan Eeik Tan. Achieving security and privacy in federated learning systems: Survey, research challenges and future directions. *Engineering Applications of Artificial Intelligence*, 106:104468, 2021.

[8] Qi Xia, Winson Ye, Zeyi Tao, Jindi Wu, and Qun Li. A survey of federated learning for edge computing: Research problems and solutions. *High-Confidence Computing*, 1(1):100008, 2021.

[9] Geming Xia, Jian Chen, Chaodong Yu, and Jun Ma. Poisoning attacks in federated learning: A survey. *IEEE Access*, 11:10708–10722, 2023.

[10] Helio N Cunha Neto, Jernej Hribar, Ivana Dusparic, Diogo Menezes Ferrazani Mattos, and Natalia C Fernandes. A survey on securing federated learning: Analysis of applications, attacks, challenges, and trends. *IEEE Access*, 11:41928–41953, 2023.

[11] Zhiyi Tian, Lei Cui, Jie Liang, and Shui Yu. A comprehensive survey on poisoning attacks and countermeasures in machine learning. *ACM Computing Surveys*, 55(8):1–35, 2022.

[12] Lingjuan Lyu, Han Yu, and Qiang Yang. Threats to federated learning: A survey. *arXiv preprint arXiv:2003.02133*, 2020.

[13] 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.

[14] Panagiota Kiourti, Kacper Wardega, Susmit Jha, and Wenchao Li. Trojdrl: evaluation of backdoor attacks on deep reinforcement learning. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020.

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

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

[17] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Local model poisoning attacks to byzantine-robust federated learning. In *Proceedings of the 29th USENIX Conference on Security Symposium*, pages 1623–1640, 2020.

[18] Junzhe Song and Dmitry Namiot. A survey of the implementations of model inversion attacks. In *International Conference on Distributed Computer and Communication Networks*, pages 3–16. Springer, 2022.

[19] Xuejun Zhao, Wencan Zhang, Xiaokui Xiao, and Brian Lim. Exploiting explanations for model inversion attacks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 682–692, 2021.

[20] Jun Niu, Peng Liu, Xiaoyan Zhu, Kuo Shen, Yuecong Wang, Haotian Chi, Yulong Shen, Xiaohong Jiang, Jianfeng Ma, and Yuqing Zhang. A survey on membership inference attacks and defenses in machine learning. *Journal of Information and Intelligence*, 2024.

[21] Mengkai Song, Zhibo Wang, Zhifei Zhang, Yang Song, Qian Wang, Ju Ren, and Hairong Qi. Analyzing user-level privacy attack against federated learning. *IEEE Journal on Selected Areas in Communications*, 38(10):2430–2444, 2020.

[22] Hongfu Liu, Bin Li, Changlong Gao, Pei Xie, and Chenglin Zhao. Privacy-encoded federated learning against gradient-based data reconstruction attacks. *IEEE Transactions on Information Forensics and Security*, 18:5860–5875, 2023.

[23] Anusha Lalitha, Shubhanshu Shekhar, Tara Javidi, and Farinaz Koushanfar. Fully decentralized federated learning. In *Third workshop on bayesian deep learning (NeurIPS)*, volume 2, 2018.

[24] Virat Shejwalkar, Amir Houmansadr, Peter Kairouz, and Daniel Ramage. Back to the drawing board: A critical evaluation of poisoning attacks on production federated learning. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1354–1371. IEEE, 2022.

[25] Valerian Rey, Pedro Miguel Sánchez Sánchez, Alberto Huertas Celdrán, and Gérôme Bovet. Federated learning for malware detection in iot devices. *Computer Networks*, 204:108693, 2022.

[26] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learning*, pages 5650–5659. PMLR, 2018.

[27] Jinhyun So, Başak Güler, and A Salman Avestimehr. Byzantine-resilient secure federated learning. *IEEE Journal on Selected Areas in Communications*, 39(7):2168–2181, 2020.

[28] Pedro Ruzafa-Alcázar, Pablo Fernández-Saura, Enrique Mármol-Campos, Aurora González-Vidal, José L Hernández-Ramos, Jorge Bernal-Bernabe, and Antonio F Skarmeta. Intrusion detection based on privacy-preserving federated learning for the industrial iot. *IEEE Transactions on Industrial Informatics*, 19(2):1145–1154, 2021.

[29] Krishna Pillutla, Sham M Kakade, and Zaid Harchaoui. Robust aggregation for federated learning. *IEEE Transactions on Signal Processing*, 70:1142–1154, 2022.

[30] Shenghui Li, Edith Ngai, and Thiemo Voigt. Byzantine-robust aggregation in federated learning empowered industrial iot. *IEEE Transactions on Industrial Informatics*, 19(2):1165–1175, 2021.

[31] Viraj Kulkarni, Milind Kulkarni, and Aniruddha Pant. Survey of personalization techniques for federated learning. In *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, pages 794–797. IEEE, 2020.

[32] Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. Mitigating sybils in federated learning poisoning. *arXiv preprint arXiv:1808.04866*, 2018.

[33] Xiaoyu Cao, Minghong Fang, Jia Liu, and Neil Zhenqiang Gong. Fltrust: Byzantine-robust federated learning via trust bootstrapping. *arXiv preprint arXiv:2012.13995*, 2020.

[34] Enrique Mármol Campos, Aurora Gonzalez-Vidal, José L Hernández-Ramos, and Antonio Skarmeta. Fedrdf: A robust and dynamic aggregation function against poisoning attacks in federated learning. *IEEE Transactions on Emerging Topics in Computing*, 2024.

[35] Vance W Berger and YanYan Zhou. Kolmogorov–smirnov test: Overview. *Wiley statsref: Statistics reference online*, 2014.

[36] 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.

[37] 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.

[38] Chao Feng, Alberto Huertas Celdran, Janosch Baltensperger, Enrique Tomas Matınez Bertran, Gerome Bovet, and Burkhard Stiller. Sentinel: An aggregation function to secure decentralized federated learning. *arXiv preprint arXiv:2310.08097*, 2023.

[39] Jian Xu, Shao-Lun Huang, Linqi Song, and Tian Lan. Byzantine-robust federated learning through collaborative malicious gradient filtering. In *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*, pages 1223–1235. IEEE, 2022.

[40] Wei Wan, Shengshan Hu, Jianrong Lu, Leo Yu Zhang, Hai Jin, and Yuanyuan He. Shielding federated learning: Robust aggregation with adaptive client selection. *arXiv preprint arXiv:2204.13256*, 2022.

[41] John R Douceur. The sybil attack. In *International workshop on peer-to-peer systems*, pages 251–260. Springer, 2002.

[42] Xingyu Li, Zhe Qu, Shangqing Zhao, Bo Tang, Zhuo Lu, and Yao Liu. Lomar: A local defense against poisoning attack on federated learning. *IEEE Transactions on Dependable and Secure Computing*, 2021.

[43] Xutong Mu, Ke Cheng, Yulong Shen, Xiaoxiao Li, Zhao Chang, Tao Zhang, and Xindi Ma. Feddmc: Efficient and robust federated learning via detecting malicious clients. *IEEE Transactions on Dependable and Secure Computing*, 2024.

[44] Jiqiang Gao, Boyu Hou, Xiaojie Guo, Zheli Liu, Ying Zhang, Kai Chen, and Jin Li. Secure aggregation is insecure: Category inference attack on federated learning. *IEEE Transactions on Dependable and Secure Computing*, 20(1):147–160, 2021.

[45] Hongsheng Hu, Xuyun Zhang, Zoran Salcic, Lichao Sun, Kim-Kwang Raymond Choo, and Gillian Dobbie. Source inference attacks: Beyond membership inference attacks in federated learning. *IEEE Transactions on Dependable and Secure Computing*, 21(4):3012–3029, 2023.

[46] Zhibo Wang, Yuting Huang, Mengkai Song, Libing Wu, Feng Xue, and Kui Ren. Poisoning-assisted property inference attack against federated learning. *IEEE Transactions on Dependable and Secure Computing*, 20(4):3328–3340, 2022.

[47] Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Helen Möllering, Thien Duc Nguyen, Phillip Rieger, Ahmad-Reza Sadeghi, Thomas Schneider, Hossein Yalame, et al. Safelearn: Secure aggregation for private federated learning. In *2021 IEEE Security and Privacy Workshops (SPW)*, pages 56–62. IEEE, 2021.

[48] Pubudu L Indrasiri, Dinh C Nguyen, Bipasha Kashyap, Pubudu N Pathirana, and Yonina C Eldar. Fedxpro: Bayesian inference for mitigating poisoning attacks in iot federated learning. *IEEE Internet of Things Journal*, 11(7):12115–12131, 2023.

[49] Kai Fan, Jingtao Hong, Wenjie Li, Xingwen Zhao, Hui Li, and Yintang Yang. Flsg: A novel defense strategy against inference attacks in vertical federated learning. *IEEE Internet of Things Journal*, 11(2):1816–1826, 2023.

[50] Jiahui Chen, Yi Zhao, Qi Li, Xuewei Feng, and Ke Xu. Feddef: defense against gradient leakage in federated learning-based network intrusion detection systems. *IEEE Transactions on Information Forensics and Security*, 18:4561–4576, 2023.

[51] Hyesung Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. On-device federated learning via blockchain and its latency analysis. *arXiv preprint arXiv:1808.03949*, 2018.

[52] Jiasi Weng, Jian Weng, Jilian Zhang, Ming Li, Yue Zhang, and Weiqi Luo. Deepchain: Auditable and privacy-preserving deep learning with blockchain-based incentive. *IEEE Transactions on Dependable and Secure Computing*, 18(5):2438–2455, 2019.

[53] Adam Richardson, Aris Filos-Ratsikas, and Boi Faltings. Budget-bounded incentives for federated learning. *Federated Learning: Privacy and Incentive*, pages 176–188, 2020.

[54] Jierui Lin, Min Du, and Jian Liu. Free-riders in federated learning: Attacks and defenses. *arXiv preprint arXiv:1911.12560*, 2019.