

Article

Can Differential Privacy Hinder Poisoning Attack Detection in Federated Learning?

Chaitanya Aggarwal ^{1,*}, Divya G. Nair ^{2,†} , Jafar Aco Mohammadi ^{1,†} , Jyothisha J. Nair ³ and Jörg Ott ⁴

¹ Nokia, 81541 Munich, Germany; jafar.mohammadi@nokia.com

² Nokia, Bangalore 560045, India; divya.g.nair@nokia.com

³ Department of Computer Science and Engineering, Amrita School of Computing, Amrita Vishwa Vidyapeetham, Amritapuri 690525, India; jyothishaj@am.amrita.edu

⁴ Department of Computer Engineering, TUM School of CIT, Technical University of Munich, 80333 Munich, Germany; ott@in.tum.de

* Correspondence: chaitanya.aggarwal@nokia.com

† These authors contributed equally to this work.

Abstract

We consider the problem of data poisoning attack detection in a federated learning (FL) setup with differential privacy (DP). Local DP in FL ensures that privacy leakage caused by shared gradients is controlled by adding randomness to the process. We are interested in studying the effect of the Gaussian mechanism in the detection of different data poisoning attacks. As the additive noise from DP could hide poisonous data, the effectiveness of detection algorithms should be analyzed. We present two poisonous data detection algorithms and one malicious client identification algorithm. For the latter, we show that the effect of DP noise decreases as the size of the neural network increases. We further demonstrate this effect alongside the performance of these algorithms on three publicly available datasets.

Keywords: federated learning (FL); autoencoder (Ae); differential privacy (DP); network anomaly detection; data poisoning



Academic Editor: Guangjie Han

Received: 25 April 2025

Revised: 21 July 2025

Accepted: 1 August 2025

Published: 6 August 2025

Citation: Aggarwal, C.; Nair, D.G.; Mohammadi, J.; Nair, J.J.; Ott, J. Can Differential Privacy Hinder Poisoning Attack Detection in Federated Learning? *J. Sens. Actuator Netw.* **2025**, *14*, 83. <https://doi.org/10.3390/jsan14040083>

Correction Statement: This article has been republished with a minor change. The change does not affect the scientific content of the article and further details are available within the backmatter of the website version of this article.

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Federated learning (FL) addresses privacy concerns by keeping data on local devices and only exchanging model updates during the training process [1,2]. Despite their privacy-preserving design, “vanilla” FL systems are susceptible to privacy attacks. Adversaries can extract sensitive information from local data by analyzing the exchanged model updates. To counter these attacks, techniques such as differential privacy (DP) [3] have been proposed. The definition of DP is presented using a mathematical framework that enables one to account for the privacy loss of a privacy-enhancing mechanism. A DP mechanism helps protect against data leakage and membership inference attacks by incorporating randomness in a function of the dataset, e.g., by adding noise. For the FL of neural networks, one DP mechanism is the addition of carefully calibrated Gaussian noise to the gradient updates before they are aggregated [4–6], which is called differentially private stochastic gradient descent (DP-SGD).

Against this background, FL is particularly susceptible to data poisoning attacks due to its distributed construct [7–10]. The intention of an adversary is often to sabotage the model accuracy on a particular task, e.g., misclassification of one particular class while the rest of the classifier performs well. Multiple solutions have been proposed to mitigate

data poisoning attacks [7,11–13]. In [11], the authors derive bounds on the loss of model accuracy for a large set of attacks. An adaptive-regularization-based method is proposed as a defense mechanism for a large variety of attacks in [12]. The authors of [7] studied the dependency of the attack efficacy on the proportion of adversarial clients to total clients in each FL round. Furthermore, they proposed norm clipping and weak differential privacy as effective mechanisms that do not decrease the overall model accuracy. The use of differential privacy as a general defense mechanism is also further investigated in [13]. The authors show that if the adversary can manipulate a limited number of data points, then DP is able to shield the model against a data poisoning attack.

While previous work studied the efficacy of DP in building resistance against data poisoning attacks, there has not been much investigation into how DP might affect the detection of a data poisoning attack. On one hand, a DP mechanism such as DP-SGD [5] injects noise into the learning procedure, which could, consequently, bury poisonous data under the noise. On the other hand, the noise from DP has been shown to mitigate the effect of poisonous data injection by increasing the model robustness [13]. This observation motivates us to investigate the effect of DP noise, particularly DP-SGD, on the detectability of data poisoning attacks.

We consider an FL setup with a proportion of malicious clients that poison the dataset. We show that for two proposed data poisoning detection algorithms, the noise added by DP does not deteriorate the detection performance. We consider three use cases, of which two are anomaly detection problems on two different types of data, and one is the consideration of an image classification problem in a federated learning setup. For the anomaly detection problem, we apply the autoencoder neural network (AeNN) architecture, and for image classification, we use the convolutional neural network (CNN) architecture. We utilize three different publicly available datasets, CSE-CIC-IDS2018 [14] and the CIC IoT dataset 2023 [15] for anomaly detection and MNIST [16] for image classification. Moreover, we develop two data poisoning detection algorithms and implement an AE-based poisoning detection algorithm from [17]. We further implement two poisoning-based attacks, namely label flipping on the data and sign flipping on the model parameters. We mathematically show how the effect of DP noise on the poisoning detection performance scales with the size of the model and the number of clients.

1.1. Summary of Contributions

This work presents a systematic investigation into how differentially private training impacts the detectability of poisoning attacks in federated learning. Our key contributions are as follows:

- We formally pose the following novel research question: Does differential privacy hinder or help in detecting poisoning attacks in federated learning? This is a question that has gone largely unexplored in the literature.
- We analyze the effect of DP-SGD noise on detection and identification accuracy under different model sizes, numbers of clients, and poisoning intensities, supported by mathematical reasoning.
- We design and implement two lightweight poisoning detection algorithms based on model performance deviation and a third algorithm to identify malicious clients. We also adapt an existing AE-based detection method [17] for benchmarking.
- We evaluate the detection and identification performance across three different FL tasks: (i) image classification on MNIST using a CNN and (ii) anomaly detection on CSE-CIC-IDS2018 and CIC-IoT2023 using autoencoder-based neural networks.
- We evaluate the detection and identification performance of data- and model-level poisoning attacks under varying levels of DP noise, including strong ($\epsilon = 0.1$), moderate ($\epsilon = 1.0$), weak ($\epsilon = 10.0$), and no privacy ($\epsilon \rightarrow \infty$).

- Our results show that both the detection and identification of poisoned clients remain effective even under strong DP constraints, thereby validating the robustness of the proposed approach.

The overall system design is presented in Figure 1 with 10 clients, with 3 of them being malicious.

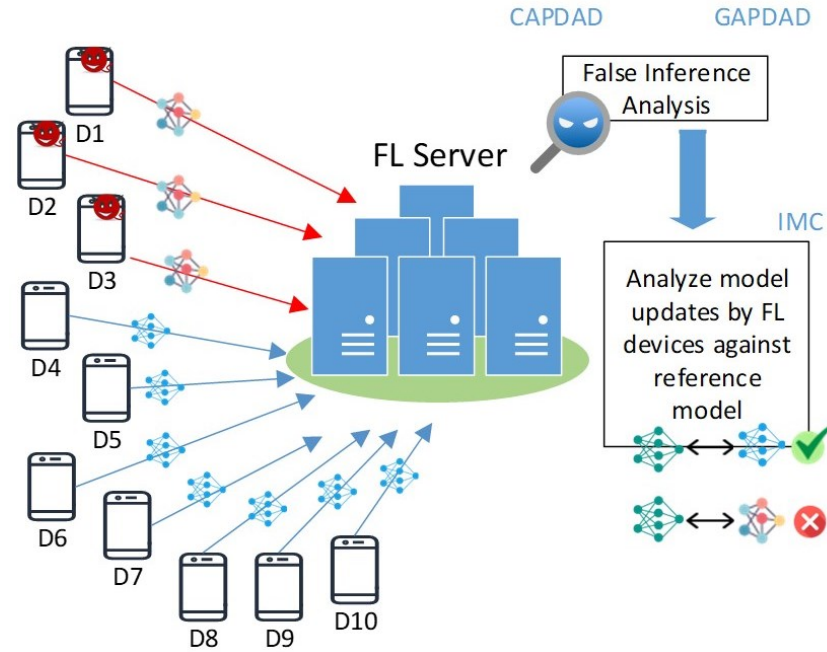


Figure 1. System framework with 30% malicious devices. On receiving the model parameters, the FL server uses Algorithm 1 (GADPAD) or Algorithm 2 (CADPAD) to detect the presence of an attack, and it uses Algorithm 3 (IMC) to identify malicious clients.

Algorithm 1 Global Model-Assisted Poisonous Data Attack Detector (GAPDAD)

- 1: W_g : Latest global model available at server
- 2: $g_k, k \in \{1, \dots, K\}$: Local client gradients
- 3: $mal_client = 0$
- 4: **procedure** AT SERVER:
- 5: Aggregate the client models

$$W'_g = W_g + \frac{1}{K} \sum_{k=1}^K C(g_k) + v_k,$$

- 6: Test W'_g with confidential test dataset
- 7: Compute the false inference probability $P_{FI}^{W'_g}$

$$P_{FI}^{W'_g} = P_{fals}^{W'_g} + P_{miss}^{W'_g}$$

- 8: **if** $P_{FI}^{W'_g} - P_{FI}^{W_g} > th_{fi}$ **then**
 - 9: $mal_client = 1$
 - 10: **else** update $W_g \rightarrow W'_g$
 - 11: **end if**
 - 12: **if** $mal_client \neq 0$ **then**
 - 13: Trigger identification of malicious client
 - 14: **end if**
 - 15: **end procedure**
-

Algorithm 2 Client-Assisted Poisonous Data Attack Detector (CAPDAD)

```

1:  $\mathbf{g}_k$ : local gradients
2:  $dev\_client = 0$ 
3: procedure AT CLIENTS:
4:   send to server: gradient updates  $\mathbf{g}_k$ 
5:   send to server:  $P_{FI}^{W_k}$ , which is locally tested  $W_k$  on  $\mathbf{x}_k$ 
6: end procedure
7: procedure AT SERVER: ( $W_k$ )
8:   for every client  $k \in \{1, \dots, K\}$  do
9:     if  $|P_{FI}^{W'_k} - P_{FI}^{W_k}| > th_l$  then
10:      Trigger identification of malicious client
11:    end if
12:  end for
13: end procedure

```

Algorithm 3 Identification of Malicious Clients (IMC)

```

1: procedure IDENTIFY_MALICIOUS_CLIENTS( $W_k, W_g$ )
2:   for Every local gradient update  $\mathbf{g}'_k, k \in 1, \dots, N$  do
3:     Compute

$$\{d_k\} \leftarrow \frac{1}{M} \|(\mathbf{g}'_k - \mathbf{g}_g)\|_2$$

4:   end for
5:   Cluster the distances  $\{d_1, \dots, d_k\}$  into 2 groups:  $\mathcal{C}_1$  and  $\mathcal{C}_2$ 
6:   if  $|\mathcal{C}_1| > |\mathcal{C}_2|$  then

$$\{mal\_client\_set\} \leftarrow \{k | d_k \in \mathcal{C}_2\}$$

7:   else

$$\{mal\_client\_set\} \leftarrow \{k | d_k \in \mathcal{C}_1\}$$

8:   end if
9:   Ignore the  $mal\_client\_set$  in federated aggregation
10: end procedure

```

1.2. Related Work

Given its significance in various domains, FL has attracted extensive research interest. However, its decentralized nature makes it vulnerable to adversarial attacks. A growing body of research focuses on detecting and mitigating such attacks using techniques ranging from robust aggregation methods and anomaly detection to explainability-driven approaches. This section provides an overview of the evolving landscape of poisoning attack detection in FL.

The intricacy of the relationship between wireless communications, machine learning, and FL is investigated in [18], which further details certain aspects of FL applications for 6G. The work in [19] investigates a privacy-preserving framework for facial paralysis detection based on FL. The authors of [20] proposed a differentially private federated learning model against poisoning attacks for an edge computing setting. The detection mechanism uses a weight-based scheme where edge nodes assign lower weights for parameters that perform poorly in the validation datasets to decrease their contributions. The work in [21] compares the performance and security of various data privacy mechanisms adopted in blockchain-based federated learning architectures. The authors of [22] provide an overview of different approaches that integrate FL with the Internet of Things (IoT). A detailed review of FL-based intrusion detection systems can be seen in [23].

In [24], anomalous client detection is performed using an AeNN at the server side. The authors of [25] propose an intrusion detection method based on memory-augmented AeNN for the Internet of Vehicles. A memory module is added to the AeNN model to enhance its ability to store the behavior of feature patterns. A novel architecture based on clustered federated learning for network anomaly detection in large-scale heterogeneous IoT networks is proposed in [26]. The recent paper [27] utilizes additive homomorphic encryption to ensure confidentiality while minimizing computational and communication overhead.

Malicious client identification in federated settings has also recently gained attention. REC-Fed [28] introduces a robust clustered aggregation scheme that filters poisoned client submissions by dynamically forming client groups based on model similarity and excluding anomalous clusters. While effective, REC-Fed does not explicitly address differential privacy constraints. FLDetector[29] analyzes the consistency of client model updates across training rounds to flag adversaries whose behavior diverges from historical norms. Similarly, SafeFL [30] constructs synthetic validation datasets at the server to detect malicious updates. The FedDMC framework [31] adopts PCA and decision tree clustering techniques to detect outliers in client-submitted model parameters, offering robust yet communication-efficient detection. Recent works such as SecureFed [32] and VFEFL [33] demonstrate enhanced malicious client detection via dimensionality reduction and functional encryption, respectively.

Most of the existing literature considers a standard FL setup without differential privacy. Our work addresses this critical gap by evaluating whether poisoning attacks can still be effectively detected in a differentially private FL environment. We focus on the ability to observe accuracy-based deviations caused by poisoning under DP constraints. A comparative summary of recent malicious client detection approaches, highlighting their assumptions, server-side requirements, and support for differential privacy, is presented in Table 1.

Table 1. Comparison of malicious client detection approaches in federated learning. ✓ indicates that DP is considered; ✗ indicates it is not.

Method	Detection Strategy	DP	Server Requirement	Remarks
REC-Fed [28]	Cluster-based anomaly filtering	✗	Client clustering and similarity checks	Strong for edge networks but not DP-compatible
FLDetector [29]	Behavioral consistency tracking	✗	History of client updates required	May fail with unstable training dynamics
SafeFL [30]	Synthetic validation dataset testing	✗	Needs generative model at server	Effective detection; impractical where synthetic data are not feasible
FedDMC [31]	PCA and tree-based clustering	✗	Low-dimensional projections from clients	Efficient but untested under DP settings
SecureFed [32]	Dim. reduction + contribution scoring	✗	Requires client-side projections	Lightweight, trade-off between privacy and accuracy
VFEFL [33]	Functional encryption-based validation	✗	Verifiable crypto setup between clients and server	Strong privacy guarantees but computationally heavy
Ours	Accuracy trend deviation under DP noise	✓	Test accuracy evaluation only	Lightweight, privacy-preserving, and works even under strong DP noise

This article is organized as follows: Section 2 describes the preliminaries and distributed training process of AeNN. Section 3 explains the proposed novel algorithms. Section 4 demonstrates the experiments conducted and the analysis of the results. Section 5 presents a discussion and the conclusions of the study.

2. Problem Statement

2.1. Preliminaries on Differential Privacy

Differential privacy has recently been widely adopted in many practical scenarios [34–36]. DP provides strict guarantees on the privacy of an individual's presence in a dataset by introducing randomness in the process of using that dataset. It ensures the protection of an individual's privacy in a dataset by incorporating randomness into the data processing pipeline. Formally, the definition is given as the following:

Definition 1 ((ϵ, δ) -Differential Privacy). *A mechanism $\mathcal{M} : \mathbb{X} \mapsto \mathcal{R}$ is considered (ϵ, δ) -differentially private when, for every two adjacent datasets X and X' and any subset of the range $S \subseteq \mathcal{R}$, the following holds:*

$$\mathbb{P}[\mathcal{M}(X) \in S] \leq e^\epsilon \mathbb{P}[\mathcal{M}(X') \in S] + \delta, \quad (1)$$

where the neighboring datasets are defined by any two datasets X and X' , such that they differ only in one data entry; further, we have $\epsilon, \delta > 0$.

Definition 1 states that the probability of the output of mechanism $\mathcal{M}$ does not change significantly if the input database changes by one entry. This definition is also known as the bounded differential privacy (the unbounded differential privacy is defined in [37]). The parameters of privacy are represented by (ϵ, δ) , where lower (ϵ, δ) guarantees more privacy protection. The parameter $\delta > 0$ represents the probability that the privacy-enhancing mechanism fails to provide privacy risk ϵ entirely. Note that (ϵ, δ) -differential privacy is known as approximate differential privacy, which is the weaker form of ϵ -differential privacy (ϵ -DP), also known as pure differential privacy. Since its introduction, differential privacy has been realized through many different mechanisms, each designed for a specific underlying privacy problem. The Gaussian or Laplace mechanisms [3,38] are among the additive noise solutions, while the exponential mechanism [39] is designed for discrete finite-space problems. In additive noise mechanisms, we need to find the sensitivity of the function to scale the standard deviation of the noise proportionally. For the scope of this study, we require the $\mathcal{L}_2$ -sensitivity and Gaussian mechanism definitions, which are given in the following.

Definition 2 ($\mathcal{L}_2$ -sensitivity [3]). *The $\mathcal{L}_2$ -sensitivity of a vector-valued function $f(X)$ is $\Delta := \max_{X, X'} \|f(X) - f(X')\|_2$, where $X, X' \in \mathbb{X}$ are neighboring datasets.*

Definition 3 (Gaussian mechanism [38]). *Let $f : \mathbb{X} \mapsto \mathbb{R}^q$ be an arbitrary function with $\mathcal{L}_2$ -sensitivity Δ . The Gaussian mechanism with parameter σ adds noise scaled to $\mathcal{N}(0, \sigma^2)$ to each of the q entries of the output and satisfies (ϵ, δ) -differential privacy for $\epsilon \in (0, 1)$ if $\sigma \geq \frac{\Delta}{\epsilon} \sqrt{2 \log \frac{1.25}{\delta}}$.*

Note that, for any given (ϵ, δ) pair, we can calculate a noise variance σ^2 such that addition of a noise term drawn from $\mathcal{N}(0, \sigma^2)$ guarantees (ϵ, δ) -differential privacy.

2.2. Distributed Setup

We assume that K nodes (clients) intend to use the FL framework to train an ML model jointly, where the data are independent and identically distributed (iid). We further assume that each node k has a dataset of $\mathbf{x}_k \in \mathbb{R}^{F \times N}$, where F is the number of features, and N is the number of data points at client k . The loss function that all of the clients are jointly minimizing is represented by $l(W_g, \mathbf{X})$, where W_g is the global model's weights, and $\mathbf{X} := [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_K]$ is the whole dataset distributed among K clients. Each node simultaneously updates its local model weights W_k according to the same loss function

$l(W_k, \mathbf{x}_k)$ on the local dataset. A stochastic gradient descent (SGD)-based method is used to minimize the loss function locally. Federated learning then uses a server to average the local gradients and send them back to each client. To save communications overhead, and since the data are iid, each node can take a few gradient steps locally before a global averaging. The equivalent updates in terms of weights can be represented as follows:

$$W_g = \frac{1}{K} \sum_{k=1}^K W_k^T, \quad (2)$$

where W_k^T is the T^{th} epoch's updated weights using an SGD-based method at the client k , run locally on dataset $\mathbf{x}_k$.

To investigate the effects of data poisoning attacks, we considered the training of an autoencoder neural network (AeNN) and a convolutional neural network (CNN) in an FL setup. In the literature, network anomaly detection has been addressed through the use of an AeNN [24,40]. The AeNN attempts to find a lower-dimensional representation of high-dimensional data, representing network features for a specific node and instance. Furthermore, we investigate an image classification problem with a CNN trained on an FL scenario.

The training of an AI model is carried out as follows. Each client receives the pre-trained/initialized global model's weights W_g from the server. Client k runs the stochastic gradient descent (SGD) locally over mini-batches of its dataset $\mathbf{x}_k$ for T iterations. The resulting local AeNN weights will be W_k , and the necessary update of gradients will be $\mathbf{g}_k$. Then, client k enters the privacy-enhancing procedure, in which the gradients need to be clipped by a constant [5], where $C(\cdot)$ is an element-wise operator that is defined as follows:

$$C(x) = \begin{cases} x & \text{if } -c \leq x \leq c. \\ c & \text{otherwise.} \end{cases} \quad (3)$$

The clipping to the value c allows for the computation of the gradient $\mathcal{L}_2$ sensitivity. We choose c according to [5] as the median of unclipped gradients.

Then, each client performs the Gaussian mechanism on their gradients and sends them to the server.

$$C(\mathbf{g}_k) + \mathbf{v}_k, \quad (4)$$

where $\mathbf{v}_k$ is sampled from $\mathcal{N}(0, \sigma^2)$ at each client independently. The value of σ^2 is given by the required (ϵ, δ) according to the definition of the Gaussian mechanism. The server then computes the following:

$$\mathbf{g}_g = \frac{1}{K} \sum_{k=1}^K C(\mathbf{g}_k) + \mathbf{v}_k, \quad (5)$$

and broadcasts back the global consensus on the gradients to each client.

2.3. Adversary Model and Attack Assumptions

We consider a white-box adversarial setting where a subset of clients are malicious and possess full knowledge of their local data, the local training process, and the FL protocol. These malicious clients are capable of altering their local datasets or model updates before sending them to the server. The attacks occur during each communication round and are persistent throughout training. The adversaries are non-colluding; each malicious client acts independently. There have been many attack scenarios in the literature for federated learning [41], from simply adding noise to the dataset to generative adversarial neural

network (GAN)-based attacks. We consider the more accessible and likely attacks for anomaly detection problems, which are label flipping and model sign flipping. In label flipping, we flip the labels of the anomaly situation in the network to the normal data.

$$y_i^{\text{poisoned}} := f(y_i), \quad \text{such that } f(y_i) \neq y_i \quad (6)$$

These data are then used to train the local model weights, causing a dramatic increase in the misdetection probability (false negative) and rendering the anomaly detection algorithm useless. In other words, label flipping pushes our detectors' sensitivity towards recognizing the anomaly as normal. In model sign flipping, a malicious client sends the flipped signs of the weights of the updated local model to the aggregator as follows:

$$w_k^{\text{poisoned}} := -w_k. \quad (7)$$

Our defense objective is to reliably detect and, where possible, identify malicious clients using only the aggregated updates and performance trends, even under differential privacy noise. Our assumptions about the attack are the following:

- In each FL round, at least $\frac{1}{2}[K] + 1$ clients are honest. A round here refers to one complete cycle where the central server distributes the global model to clients, who then perform local training on their data and send the updated model back to the server, and the server aggregates these updates to refine the global model.
- Each client updates its model for T iterations locally before sharing it with the server.
- Each client performs (ϵ, δ) -DP locally by using a Gaussian mechanism on globally agreed-upon values for ϵ and δ , as described in Section 2.
- No zero-day attack is expected; we have some genuine prior models.
- The data distribution of a client does not change during a training round.

3. Detection of Poisonous Data in the Presence of DP

In this section, we present two algorithms, each with different practical assumptions about how to detect the existence of an attack. Finally, we present an algorithm for identifying which user is injecting poisoned data into the system.

3.1. Attack Detection at the Server

We propose two different algorithms for detecting the presence of an anomaly among the clients' updates in each round by monitoring the weight updates.

In the first approach, the *Global Model-Assisted Poisonous Data Attack Detector (GAPDAD)*, the details of which are given in Algorithm 1, we assume that the global model and a confidential test dataset $\mathbf{x}_{\text{sec}}$ are available at the server. Since the dataset is iid and the data distribution does not evolve much over time, this assumption can be extremely helpful in detecting the anomaly. In other words, we assume that the whole training procedure is a fine-tuning procedure and that we have no zero-day attack. The details are given in the following.

After the gradient aggregation at the server, from (5), we have

$$W'_g = W_g - \eta \frac{1}{K} \sum_{k=1}^K C(\mathbf{g}_k) + \mathbf{v}_k, \quad (8)$$

where η is the learning rate, and W'_g is the updated global model, which is achieved by aggregating gradients from each client. If we test this model with the secret test data $\mathbf{x}_{\text{sec}}$, we can compute two main detection metrics, namely the probability of misdetection (false negative) P_{miss} and the probability of a false positive P_{fals} ; these are defined as follows:

$$P_{miss} = P(x \in \mathcal{O} | x \in \mathcal{A}) \quad (9)$$

$$P_{fals} = P(x \in \mathcal{A} | x \in \mathcal{O}) \quad (10)$$

where $\mathcal{O}$ and $\mathcal{A}$ denote the set of ordinary (normal) events and the set of anomalous events.

3.1.1. Global Model-Assisted Poisonous Data Attack Detection

In addition to our general assumptions, we further consider the existence of the global model at the server along with a confidential test dataset. Since the presence of the poisonous data degrades the performance of our global model W_g , we can detect an attack by monitoring the change in the probability of false inference for the model W_g , which is defined as $P_{FI}^{W_g} := P_{fals}^{W_g} + P_{miss}^{W_g}$. This has been given in detail in Algorithm 1. In Algorithm 1, th_{fi} is the threshold, which is determined in practice based on the dataset.

3.1.2. Client-Assisted Poisonous Data Attack Detection

In certain scenarios, assuming the existence of a confidential test dataset and the fact that the distribution of the dataset remains constant over time is not very practical. We propose that each client test the updated model before uploading it to the server. We use P_{FI} as a performance indicator of a model. If P_{FI} for a client changes much from its previous update, then we claim that there might have been poisonous data in the previous round. Note that, here, we cannot pinpoint the exact attacker client. We only have the assumption that the number of possible malicious clients will be strictly smaller than the honest clients. Similarly to Algorithm 1, th_l depends on the data and model sensitivity and can be determined in practice. The details of this approach are given in Algorithm 2, the *Client-Assisted Poisonous Data Attack Detector* (CAPDAD).

3.2. Identification of the Adversarial Clients

We are interested in identifying the attacker to prevent poisonous data from damaging our FL model. A meaningful sabotage of the model occurs when each update of a malicious client $\mathbf{g}'_k$ deviates significantly from the previous global model $\mathbf{g}_g$. It is difficult to propose a threshold to compare the model deviation due to the dynamic nature of gradient updates via any optimizer, dataset, and loss function variations. Thus, we propose the use of the assumption that the majority of the clients are honest. Therefore, we measure the l_2 -norm of the difference in the latest global gradients $\mathbf{g}_g$ with each client's current update $\mathbf{g}'_k$. This will map high-dimensional data $\in \mathbb{R}^{M \times 1}$ into just one scalar. Since the majority of the clients are always honest, if we simply use a two-class clustering algorithm, we can identify the set of anomalous clients. Any client belonging to the set with a smaller cardinality is considered to be malicious. The detailed algorithm is presented in Algorithm 3. Note that this algorithm does not require any thresholds, as it only relies on the assumption that "the majority of the clients are honest", thus making it very flexible for practical use. In Algorithm 3, $|\mathcal{C}_1|$ indicates the cardinality of the set $\mathcal{C}_1$.

3.3. The Effect of DP Noise on the Detection Performance

The detection and identification algorithms are not impacted dramatically by the additive noise ν_k as a result of DP. This claim is observed through the extensive numerical analysis of multiple different NNs and datasets. In this section, we demonstrate why DP noise does not hinder the detection performance of the proposed algorithms. Intuitively, one explanation could be as follows. Since the additive noise has zero mean $\mathcal{N}(0, \sigma^2)$ and, by design, has to be drawn at random and independently for all of the entries of $\mathbf{g}_k$ and for all of the clients, we are essentially taking an empirical expectation over the additive noise values, i.e., $\frac{1}{K} \sum_k \nu_k$, which approaches zero with larger model parameters and more

clients. To illustrate this effect better, we derive the first- and second-order statistics of the difference between the identification of a client with or without DP in the following. In Algorithm 3, d_k can be written as follows:

$$d_k(DP) = \frac{1}{M} \|C(\mathbf{g}'_k) + \mathbf{v}'_k - \frac{1}{K} \sum_{k=1}^K (C(\mathbf{g}_k) + \mathbf{v}_k)\|_2 \quad (11)$$

$$\begin{aligned} &\leq \frac{1}{M} \|C(\mathbf{g}'_k) - \frac{1}{K} \sum_{k=1}^K C(\mathbf{g}_k)\|_2 \\ &\quad + \left\| \frac{1}{M} \mathbf{v}'_k - \frac{1}{MK} \sum_{k=1}^K \mathbf{v}_k \right\|_2 \end{aligned} \quad (12)$$

$$= d_k(noDP) + \left\| \frac{1}{M} \mathbf{v}'_k - \frac{1}{MK} \sum_{k=1}^K \mathbf{v}_k \right\|_2, \quad (13)$$

where $d_k(DP)$ and $d_k(noDP)$ are values of the parameter d_k defined in Algorithm 3 for the cases of the presence of DP noise and no DP noise, respectively. Note that the $C(\cdot)$ operation does not introduce much distortion due to the normalized values of the gradients [5]. We define

$$\boldsymbol{\zeta} := \frac{1}{M} \mathbf{v}'_k - \frac{1}{MK} \sum_{k=1}^K \mathbf{v}_k,$$

where the norm value $\|\boldsymbol{\zeta}\| > 0$ follows by definition. In the following, we show that both the expected value and the second-order statistics of $\|\boldsymbol{\zeta}\|_2$ decrease as the size of the gradient vector M increases.

We further compute $\mathbb{E}(\|\boldsymbol{\zeta}\|_2^2)$ in the following:

$$\mathbb{E}(\|\boldsymbol{\zeta}\|_2^2) = \mathbb{E}\left(\frac{1}{(M)^2} \|\mathbf{v}'_k - 1/K \sum_{k=1}^K \mathbf{v}_k\|_2^2\right) \quad (14)$$

$$= \frac{1}{(KM)^2} \mathbb{E}\left(\sum_{i=1}^M \left(K\mathbf{v}'_k - \sum_{k=1}^K \mathbf{v}_{k,i}\right)^2\right) \quad (15)$$

$$= \frac{1}{(KM)^2} \mathbb{E}\left(\sum_{i=1}^M \left((K\mathbf{v}')_k^2 + \sum_{k=1}^K \mathbf{v}_{k,i}^2\right)\right) \quad (16)$$

$$= \frac{K+1}{KM} \sigma^2 \quad (17)$$

where, from (15) to (16), we used the independence of the noise values. The values of M and K will drastically reduce the effect of the additive noise of differential privacy.

Computing $\mathbb{E}(\|\boldsymbol{\zeta}\|_2)$ requires some results from the literature [42]; the norm of a vector of Gaussian random variables ($\boldsymbol{\zeta}$ is Gaussian, as it is the sum of independent Gaussian random variables) is distributed according to the Chi distribution. Thus, the mean is given by the following:

$$\mathbb{E}(\|\boldsymbol{\zeta}\|_2) = \frac{\sqrt{2} \Gamma(\frac{M+1}{2})}{\Gamma(\frac{M}{2})} \sigma_{\boldsymbol{\zeta}} \quad (18)$$

where $\sigma_{\boldsymbol{\zeta}}$ is the standard deviation of the random variable entries of $\boldsymbol{\zeta}$, and Euler's gamma function is denoted by $\Gamma(\cdot)$. For large numbers of M , the above can be further approximated using

$$\mathbb{E}(\|\boldsymbol{\zeta}\|_2) \approx \sqrt{(M - \frac{1}{2})} \sigma_{\boldsymbol{\zeta}}. \quad (19)$$

We can substitute this with the values for $\sigma_{\boldsymbol{\zeta}}$, as follows:

$$\mathbb{E}(\|\zeta\|_2) \approx \frac{(K+1)\sqrt{(M-\frac{1}{2})}}{MK}\sigma. \quad (20)$$

which is a decreasing function of M .

This result demonstrates that the difference $\|\zeta\|_2$ in terms of both $\mathbb{E}(\|\zeta\|_2)$ and $\mathbb{E}(\|\zeta\|_2^2)$ decreases as the training model becomes larger (larger gradient vector). In other words, the effect of DP noise in the detection could be negligible when the NN model's size is large. We have further shown that the difference between having differential privacy $d_k(DP)$ and not having differential privacy $d_k(noDP)$ is always positive, i.e., $d_k(DP) - d_k(noDP) > 0$. This could be helpful in the clustering scenario, i.e., all the distances will be shifted more or less with larger numbers.

In this section, we present an extensive numerical analysis of different datasets and scenarios to show the effectiveness of our algorithms in the presence of DP noise.

4. Experiments and Results

4.1. Simulation Setup and Datasets

We conducted our experiments using three datasets: the CSE-CIC-IDS2018 dataset [14], the CIC IoT dataset 2023 [15], and the MNIST image dataset [16]. The first two datasets consist of classes of attacks in network and IoT devices, respectively, and they are utilized for anomaly detection tasks. The base anomaly detector is developed using an AeNN consisting of four encoder layers and four decoder layers. The MNIST dataset is employed for an image classification task for 10 digits and is implemented with SimpleNet in this work. SimpleNet is a convolutional neural network with 13 layers. The network employs a homogeneous design utilizing 3×3 kernels for the convolutional layer and 2×2 kernels for pooling operations; it was introduced in [43].

In order to simulate FL, we used the FLOWER framework [44]. Our simulation setup consists of 10 clients, wherein we considered varying numbers of devices to be malicious, with and without the presence of DP. Our training data for CIC-IDS 2018, CIC-IOT, and MNIST consist of 1 M, 1 M, and 60 K training samples, respectively, distributed across 10 clients. The samples assigned to each client are drawn randomly from the entire dataset, preserving the overall class proportions and statistical properties.

To simulate a data poisoning attack, the labels of the samples are flipped. For a model poisoning attack, the sign of the model parameters is flipped and scaled. These malicious model updates are then sent to the federated learning (FL) server for global aggregation. The experiments are conducted with varying numbers of malicious devices in the presence of strong DP noise with $\epsilon = 0.1$. Additionally, experiments are performed to analyze the impact when only a portion of a malicious device's complete training data are poisoned. To understand the effect of the poisoning attack and the performance of detection algorithms, we consider the changes in overall P_{FI} . Finally, to evaluate the detection performance in the presence of strong DP noise, independent of the detection technique used, we additionally conducted experiments with the state-of-the-art autoencoder-based poisoning attack detection mechanism, as described below.

4.2. Experiment I: Impact of Poisoning Attacks with FL and DP

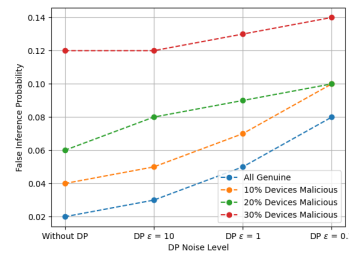
Table 2 presents the performance metrics of our model, including the precision, recall, and accuracy. Precision is the ratio of true positive predictions to all positive predictions, providing insight into the model's correctness when predicting the positive class. Recall, the ratio of true positives to all actual positives, measures the model's ability to capture all relevant instances. Accuracy indicates the overall correctness of the model's predictions.

Together, these metrics offer a comprehensive evaluation of the model's performance. The results are shown for normal FL, FL with DP without an attack, and that with model and data poisoning attacks. The addition of DP noise increased P_{FI} , and other metrics were also affected; with poisoning attacks, the impacts were even more pronounced.

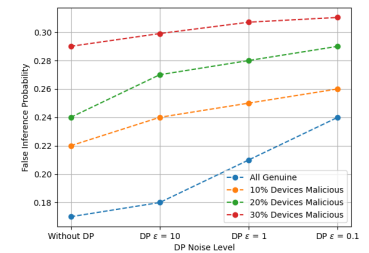
Table 2. Performance metrics across different experiments and datasets.

Dataset	Experiment	Accuracy	False Inference	Precision	Recall
CICIDS	FL	98.60	0.01	0.95	1.0
	FL with DP	91.41	0.08	0.99	0.74
	Data Poisoning	88.30	0.12	0.93	0.69
	Model Poisoning	63.20	0.36	0.47	0.94
IoT	FL	85.88	0.14	0.67	0.98
	FL with DP	72.64	0.27	0.52	0.54
	Data Poisoning	68.94	0.31	0.45	0.49
	Model Poisoning	38.73	0.61	0.26	0.66
MNIST	FL	92.05	0.07	0.92	0.92
	FL with DP	91.76	0.08	0.91	0.92
	Data Poisoning	85.53	0.14	0.88	0.85
	Model Poisoning	69.81	0.31	0.70	0.70

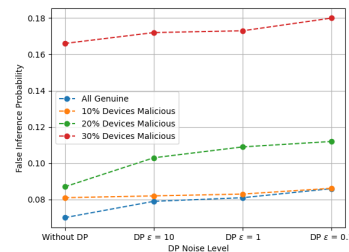
Figure 2 depicts the impact of DP noise levels ($\epsilon = 0.1, 1$, and 10) on the data poisoning attacks experienced by genuine devices when 10–30% of the clients are malicious. In all three figures (Figure 2a–c), as the DP noise level increases (ϵ decreases), P_{FI} also increases, which is consistent across all scenarios. The plots indicate that when more devices are malicious (e.g., 30%), P_{FI} is generally higher compared with cases with fewer malicious devices or all genuine devices. Furthermore, the impact of the attack may also vary with the dataset selected. For instance, the attack impact is visible in the IDS and IoT datasets with 10% malicious devices, and the largest amount of DP noise is visible with 20% malicious clients in the MNIST dataset under the same settings.



(a) CSE-CIC-IDS2018 dataset



(b) IoT dataset



(c) MNIST dataset

Figure 2. Impact of data poisoning attacks on P_{FI} under varying DP noise levels ($\epsilon \in \{0.1, 1, 10, \infty\}$) with different proportions of malicious devices ($\in \{10\%, 20\%, 30\%\}$).

4.3. Experiment II: Detection of Poisoning Attacks Using the GAPDAD and CAPDAD Algorithms 1 and 2

To statistically validate our method, we apply the principles of Monte Carlo simulations; we ran the entire FL training 100 times. At each new FL, we sampled uniformly at random over the total dataset and selected 30% of the data points. The selected 30% were turned into malicious data through label flipping.

The resulting plots are illustrated in Figure 3. The red dashed line in the figure represents the threshold established based on the P_{FI} values from genuine cases. As depicted in the figures, the P_{FI} values in most abnormal cases exceed this threshold, demonstrating the effectiveness of our detection algorithm. The overlapping values, where some false positives and false negatives occur, determine the overall performance of the algorithm in identifying malicious activities.

Figure 4 compares GAPDAD and CAPDAD in a scenario where 30% of the clients are malicious, with the strongest DP noise being applied at $\epsilon = 0.1$. In this setup, Devices 1, 2, and 3 are assumed to be malicious, and we expect them to send falsified P_{FI} values to the FL server during Rounds 3 and 4 after initiating the attack to hide their poisonous data injection. The majority of clients (D4–D10) are honest; thus, they will still send correct P_{FI} values in Round 4. Therefore, the server, after analyzing the deviation of P_{FI} received from each client, is able to detect the attack (since the majority of the devices are showing an anomalous increase in deviation compared with previous FL rounds). Furthermore, we observe that GAPDAD is able to detect the attack in Round 3 itself (i.e., before even sending the poisonous aggregated model to the other clients), whereas CAPDAD is able to detect the attack in Round 4. Therefore, even though there is a trade-off between the availability of the test dataset at the FL server and the proactive detection of the poisoning attack, both algorithms are successful in attack detection.

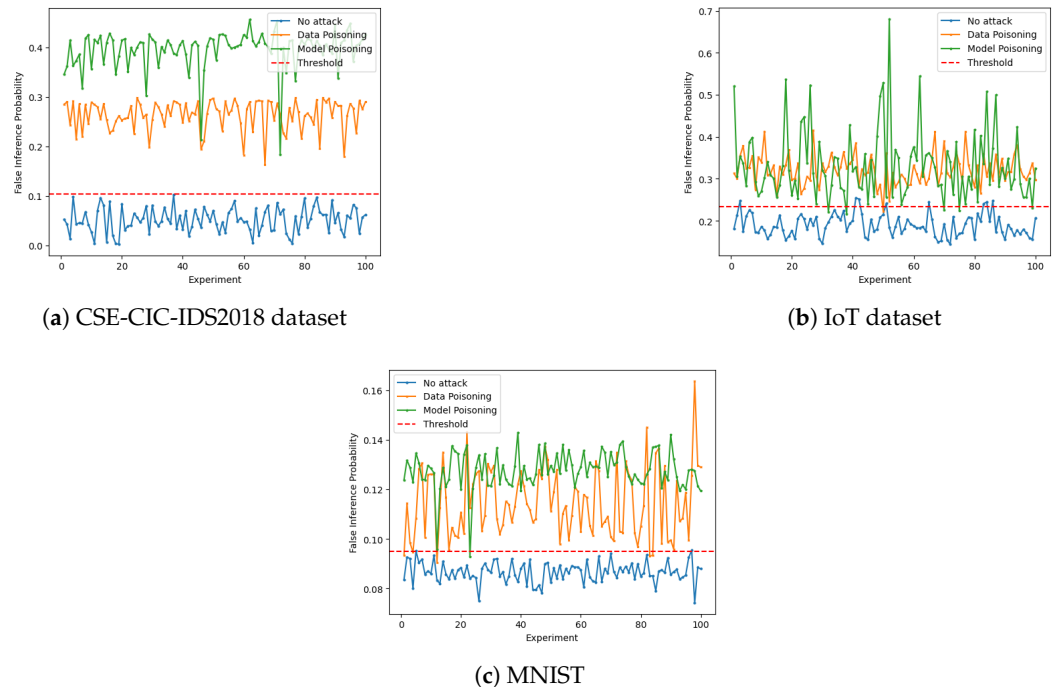


Figure 3. Impact of data poisoning and model poisoning attacks on P_{FI} in various experimental scenarios across different datasets. The figure illustrates P_{FI} for 100 rounds across no-attack, data poisoning, and model poisoning scenarios, with the red dashed line indicating the threshold for acceptable values of P_{FI} .

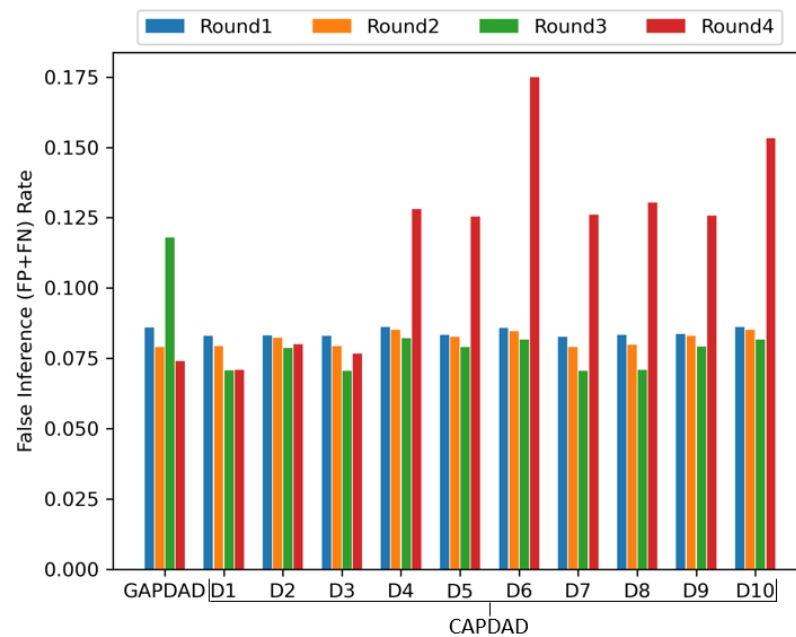


Figure 4. Comparison of the GAPDAD and CAPDAD attack detection algorithms. GAPDAD can detect the attack earlier than CAPDAD in Round 3, immediately after the attack's occurrence.

Table 3 provides comprehensive insights with respect to the performance of the proposed detection algorithm in 100 different FL experiments. The detection algorithm demonstrates strong and reliable performance against both data and model poisoning attacks across all evaluated datasets. For MNIST, it achieves a precision between 0.979 and 0.980, a recall of up to 0.99, an accuracy of around 97%, and robustness (measured using the Matthews correlation coefficient, MCC [45]) of close to 0.97. The IDS dataset records a precision of 0.970, accuracy of 98%, and robustness of 0.970, reflecting stable and consistent detection performance. For the IoT dataset, the algorithm attains a precision between 0.83 and 0.84, an accuracy of around 88%, and robustness values ranging from 0.76 to 0.82. The MCC is used to measure robustness because it incorporates all elements of the confusion matrix (TP, TN, FP, and FN), offering a balanced evaluation of performance even when class distributions are uneven. Overall, these results show that while the detection algorithm performs very well against both data and model poisoning attacks in structured datasets such as MNIST and IDS, it still provides good accuracy for the IoT dataset, but it highlights the need for additional tuning to improve the robustness under more complex conditions.

Table 3. Algorithm performance evaluation metrics grouped by attack type.

Poisoning Type	Dataset	Precision	Recall	Accuracy	Robustness
Data Poisoning	CSE-CIC-IDS2018	0.970	1	0.985	0.97
	CIC-IoT-2023	0.832	0.940	0.875	0.76
	MNIST	0.971	0.95	0.965	0.93
Model Poisoning	CSE-CIC-IDS2018	0.970	1	0.985	0.97
	CIC-IoT-2023	0.840	1.000	0.905	0.83
	MNIST	0.98	0.99	0.985	0.97

Figure 5 illustrates the impact of varying differential privacy noise levels on the false inference probability difference for both the IDS and IoT datasets. As the DP noise increases to a very high level (lower ϵ values), the difference between malicious and genuine cases decreases. In the case of the IDS dataset, a 33% reduction in deviation difference is observed

under the strongest noise setting ($\epsilon = 0.1$) compared with the lowest-noise scenario ($\epsilon = 10$), whereas for the IoT dataset, the reduction is approximately 41%. This consistent trend across both datasets highlights how very strong privacy mechanisms may “blur” the separability between malicious and genuine outcomes, making accurate detection more challenging and underscoring the inherent privacy–utility balance of DP-based methods, at least in the case of very strong DP noise.

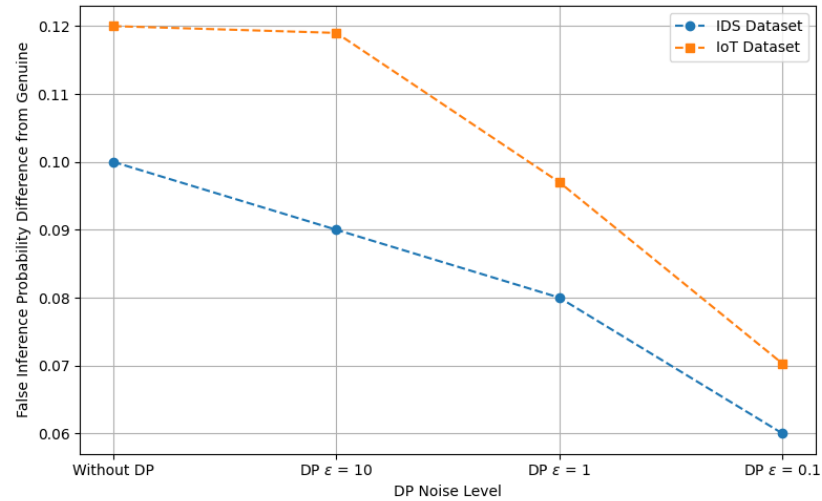


Figure 5. Impact of DP noise levels on the false inference probability difference for the IDS and IoT use cases.

4.4. Experiment III: Identification of Malicious Clients

Once the malicious client detection is triggered by Algorithms 1 or 2, identification of the malicious devices is performed by analyzing the deviation of local model parameters sent by the clients in each round from the reference global model. Figure 6 shows how the three malicious devices (D1, D2, and D3) and their poisonous model updates diverge from the reference model at the FL server. Here, the Y-axis represents the deviation of each device model from the global reference model and is computed as in Algorithm 3, Equation (3). We observe that for the AeNN, there is a difference in deviation magnitudes with respect to the encoder layer. The last encoder layer (i.e., in our case, Layer 6) provides the best results, while we could not detect malicious devices if we considered the parameters from Layer 2 of the encoder. The last layer captures the most relevant information, making it a useful representation for tasks such as dimensionality reduction or feature extraction. Thus, the last encoded layer in an AeNN is often used for transfer learning. The mean value of Layer 6 is used to plot Figure 6a for the CICIDS dataset and Figure 6b for the IoT dataset. Further, for the MNIST dataset, the results are shown in Figure 6c, where the complete model parameters’ mean is used to plot the graphs.

We observe that even in the presence of strong DP noise, identification of malicious devices is still possible. To examine the performance limits of the proposed Algorithm 3, we plot the accuracy of our identification algorithm by conducting 100 experiments with random clients that are poisonous at any instance. These experiments are conducted while keeping the amount of noise at a minimum so that we can observe the impact on false inference. The results are plotted in Figure 7a–c for the three datasets.

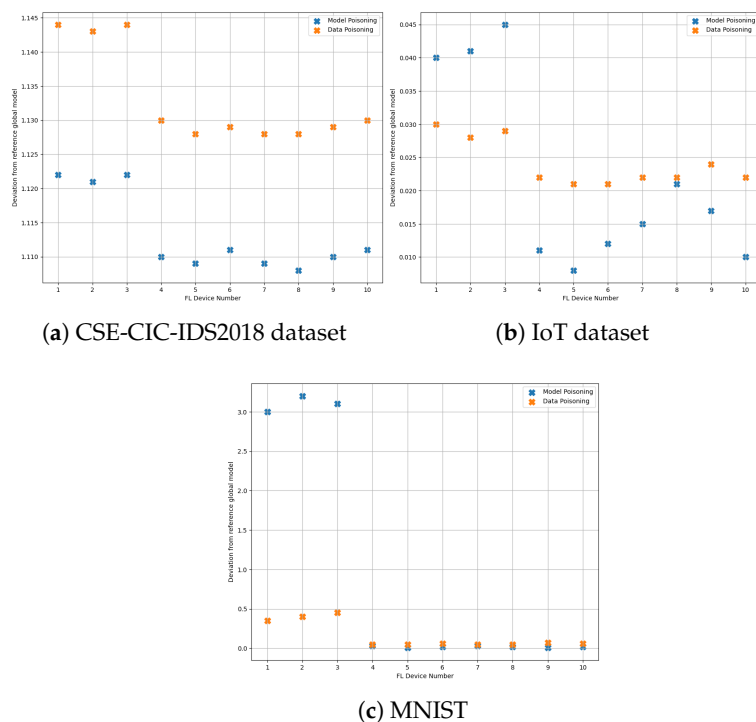


Figure 6. Identification of malicious nodes in poisoning attacks. The Y-axis represents the deviation of each device model from the global reference model, calculated using the Euclidean distance between the weight vectors.

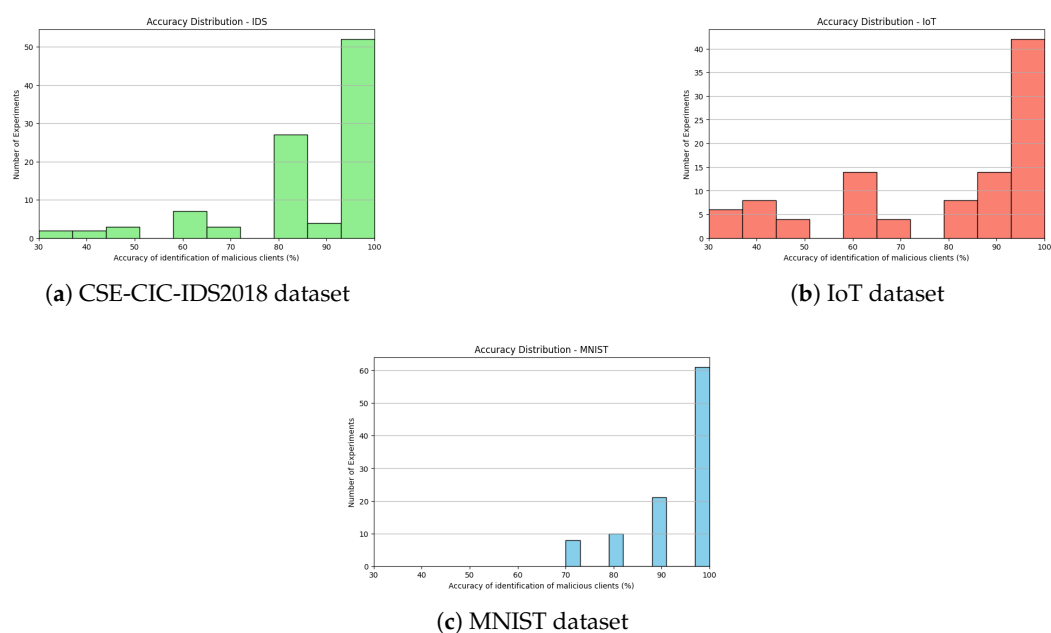


Figure 7. Accuracy histogram plots for identifying malicious clients across 100 experiments, with random clients being poisoned in each instance. Despite occasional dips, the algorithm consistently maintains high accuracy, demonstrating its robustness even in challenging scenarios.

The graphs highlight the promising potential of the proposed identification algorithm, even under challenging conditions with minimal DP noise. Across 100 experiments, where the selection of clients and the presence of malicious behavior were randomized, the algorithm consistently demonstrates its ability to accurately identify malicious devices. While the results show some variability in accuracy, with occasional dips, these instances are outliers in an overall strong performance.

Finally, to investigate the sensitivity of our approach, we decrease the percentage of poisonous data in a single malicious device. Figure 8a,b show the decrease in deviation from the reference global model as the amount of poisonous data in a given training round decreases. When the poisonous data are just 5% of the complete training data, we observe that the algorithm is barely able to identify the malicious device. Thus, if we further reduce the poisonous data percentage, there exists a point where multiple malicious devices remain undetected while still being able to poison the global model.

We further analyze the extent to which differential privacy noise impairs malicious client identification capabilities by varying the privacy parameter ϵ across the datasets. Figure 9 illustrates how the identification performance is degraded as the DP noise level increases (i.e., as ϵ decreases). Our results demonstrate that the threshold at which noise begins to hinder malicious client detection is not fixed but depends on the dataset and model architecture. For instance, with $\epsilon = 0.1$, for the CSE-CIC-IDS2018 dataset, the deviation drops by 48%, while the CIC-IoT-2023 dataset exhibits an 81% decrease in deviation difference between malicious and genuine clients. This suggests that datasets with richer feature representations or more stable behavior patterns are more resilient to noise. Moreover, deeper model architectures exhibit better robustness under high DP noise, benefiting from their capacity to generalize across noisy updates.

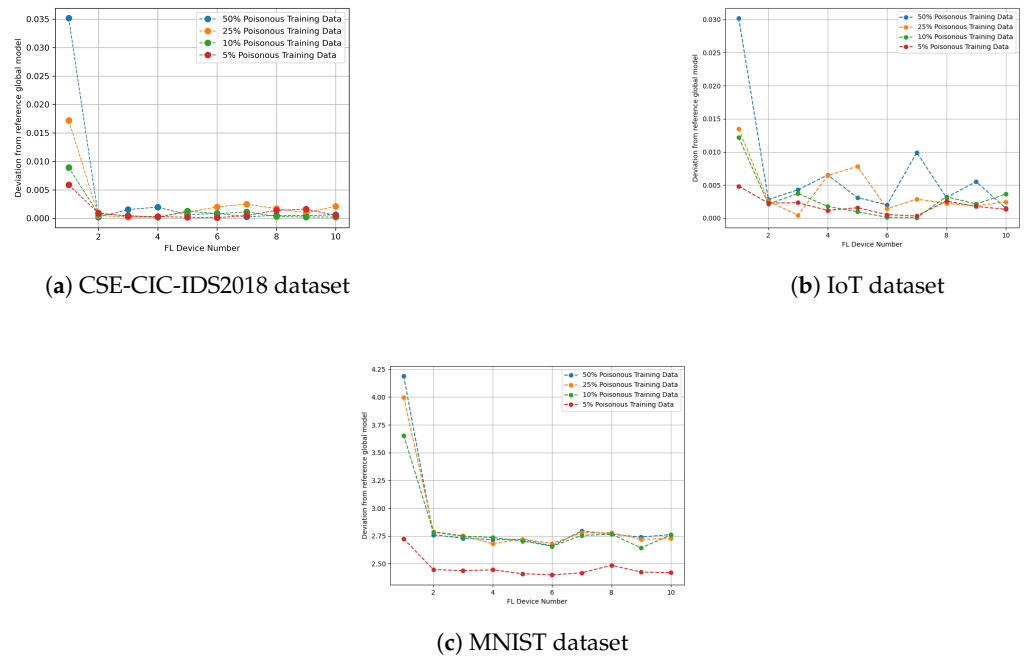


Figure 8. Impact on the identification of malicious clients over varying percentages of poisonous data on a single client. When poisonous data constitute around 5% of the training data, the algorithm struggles to identify the malicious device.

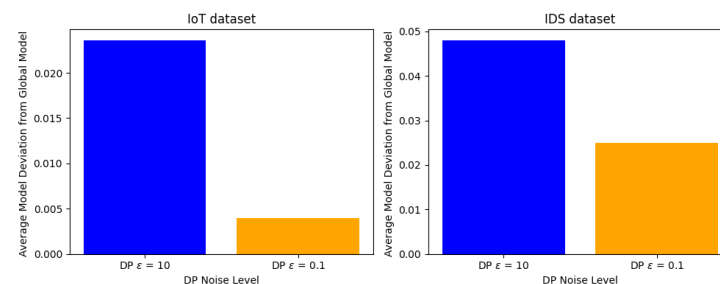


Figure 9. Impact of varying DP noise levels on detection accuracy. A significant performance drop is observed as ϵ decreases by a factor of 100 times, which is equivalent to 100 times more added DP noise in this case.

4.5. Experiment IV: AeNN-Based Identification of Malicious Clients

In this experiment, we explore an alternative approach for identifying malicious devices based on the reconstruction error using an AeNN. Specifically, three distinct AeNNs are trained on genuine model data for each dataset. These AeNNs learn to reconstruct the input models accurately. When a malicious model is introduced to the corresponding AeNN, the reconstruction error is used as an indicator of anomalies. A significant reconstruction error suggests that the model is likely malicious, as the AeNN trained on genuine data struggles to accurately reconstruct an anomalous input.

This method leverages the AeNN's ability to compress and then reconstruct the data, with the assumption that it will perform poorly on data that deviate significantly from what it was trained on. Thus, a high reconstruction error serves as a signal for the presence of potentially malicious activity in the FL setting. The results are plotted in Figure 10, where both model and data poisoning attacks are detected and identified.

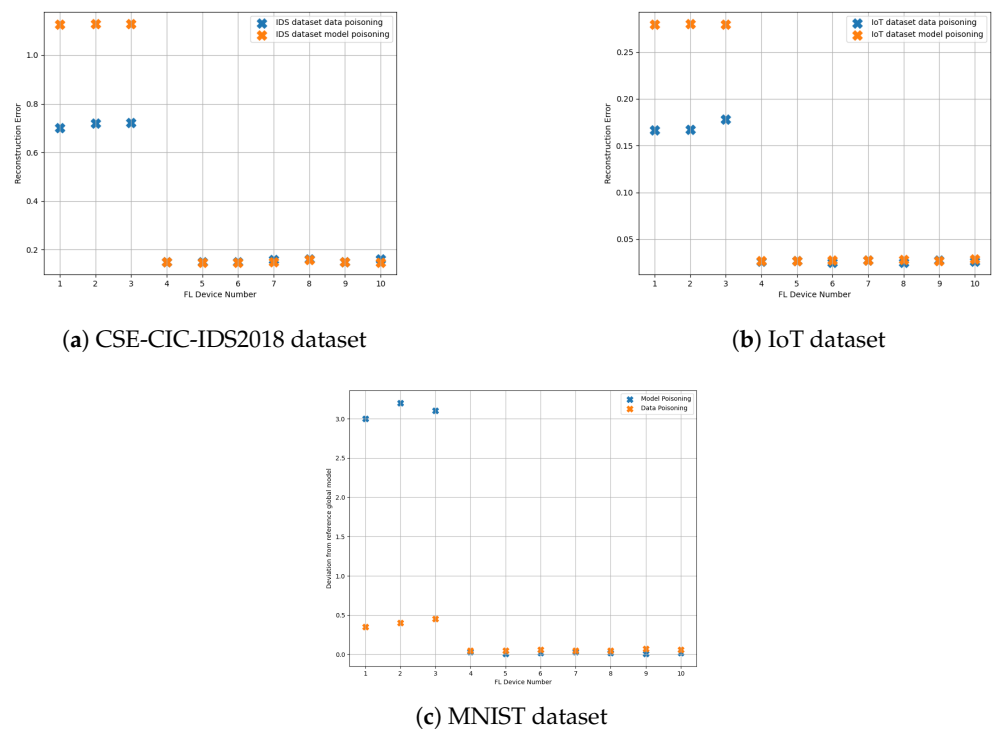


Figure 10. Reconstruction-error-based detection and identification of malicious clients. The first three devices are designated as malicious, and their elevated reconstruction errors clearly distinguish them from the non-malicious devices.

4.6. Experiment V: Analysis of Detection and Identification Trends Across Datasets

In this experimental setting, the percentage of poisonous data remains the same, but the number of malicious devices varies. Figure 11a–c represent the detection and identification trends in different datasets, with each indicating how the FI rate and mean model deviation change as the percentage of malicious devices increases. These plots illustrate the detection and identification trends across different datasets when varying percentages of malicious devices are present. A critical observation from these trends is the relationship between the distribution of poisoned data and the model's ability to detect and identify malicious activity.

As the poisoned data are distributed across a larger number of devices, the overall impact on the global model tends to diminish and become more stable. This is evident from the mean model deviation in all three graphs, where deviations from the global model either decrease or stabilize as more devices are poisoned. This stabilization indicates that

when malicious activity is spread out across many devices, the local effects on the global model are less pronounced, making it harder for the model to detect these subtle deviations.

The trends in these graphs underscore the importance of robust detection mechanisms in federated learning systems. While distributing poisoned data across more devices might seem to reduce the impact, it actually introduces a higher risk of undetected, long-term attacks. This “slow poisoning” approach could lead to a gradual degradation of model performance or even cause the model to adopt harmful biases. Therefore, developing advanced detection techniques that can identify these subtle, distributed attacks is crucial for maintaining the integrity and security of federated learning models.

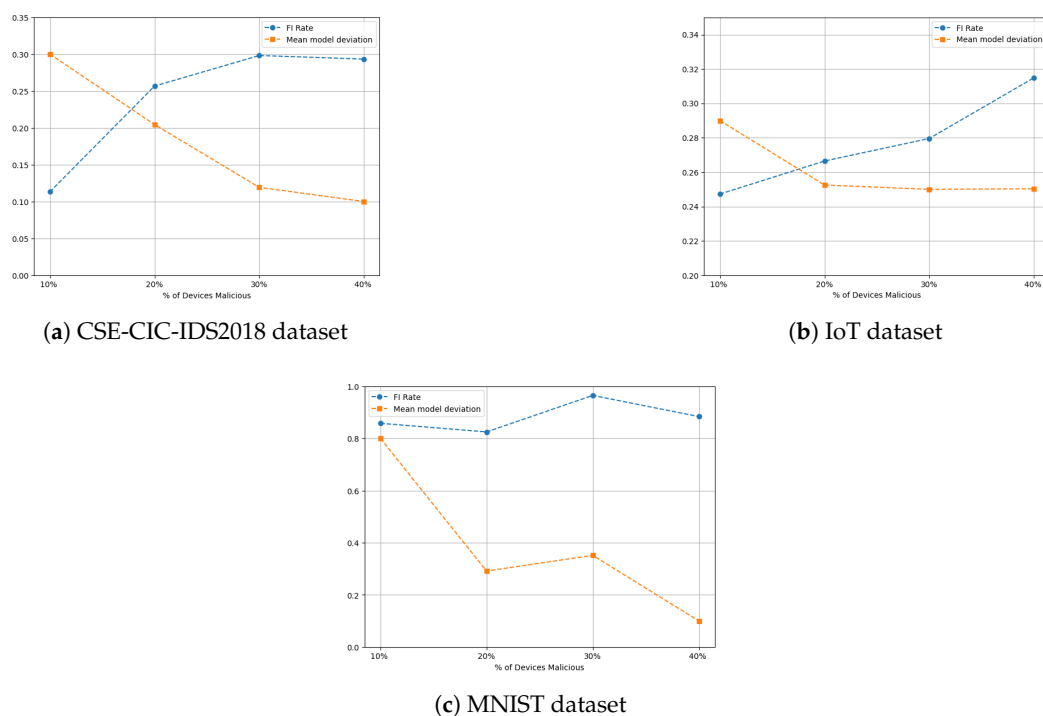


Figure 11. Detection and identification of malicious clients. The blue line shows the deviation in P_{FI} , and the red line shows the mean model deviation from the reference model. Values are taken from the global aggregated model.

4.7. Experiment VI: Impacts of Differential Privacy Across Varying Model Architectures

To evaluate the effects of differential privacy noise and poisoning on models with varying capacities, an experiment was performed using four convolutional neural network (CNN) architectures of increasing complexity: SmallCNN, MediumCNN, ExtendedMediumCNN, and DeepCNN. These architectures differ in the number of convolutional layers, hidden units, and depth. Table 4 summarizes the model configurations and the corresponding number of trainable parameters.

Table 4. Model architectures and parameter counts.

Architecture	Layer Composition	Parameter Count
SmallCNN	Conv(1,8,3) + ReLU + MaxPool + FC(1352,10)	13.6 K
MediumCNN	2 Conv + ReLU + MaxPool + FC(3872,10)	43.5 K
ExtendedMediumCNN	3 Conv + ReLU + MaxPool + FC(5184,256) + FC(256,10)	1.35 M
DeepCNN	3 Conv + ReLU + MaxPool + FC(5184,256) + FC(256,128) + FC(128,10)	1.38 M

In this experiment, we introduce label poisoning by flipping a single class label in a controlled manner. Specifically, we select a small proportion of training samples with the true label “1” and flip them to “7”, thus simulating a targeted label poisoning attack. The

models are trained under DP constraints with a fixed noise standard deviation computed using $\epsilon = 0.1, \delta = 10^{-5}$.

Figure 12 shows the accuracy of each architecture on the MNIST test set under DP training with label poisoning. We observe a clear trend aligned with (16) and (20); larger models with more parameters exhibit better resilience to DP-induced noise and label poisoning. This confirms the intuition that deeper architectures can better absorb the noise introduced during gradient updates. Moreover, the experiment shows that even in the presence of privacy-preserving mechanisms, label poisoning attacks, though limited to a single class, can impact simpler models more severely. Hence, a comprehensive robustness evaluation under varying architectural depths is essential when deploying privacy-aware models in adversarial environments.

By showing that applying DP does not dramatically deteriorate the anomaly detection procedure performed during the federated learning phase, we encourage the use of DP. The presented theoretical scaling law of equations (16) and (20) clearly demonstrates that the size of the NN can compensate for the introduced discrepancy by adding Gaussian noise of $\mathcal{N}(0, \sigma^2)$. This, coupled with the finding that DP enhances the resilience of the NN towards poisoning attacks [7], further justifies the practical use of DP and, in particular, the Gaussian mechanism during any federated learning training to not only enhance privacy and robustness against poisoning attacks but also allow for the undeterred detection of data poisoning.

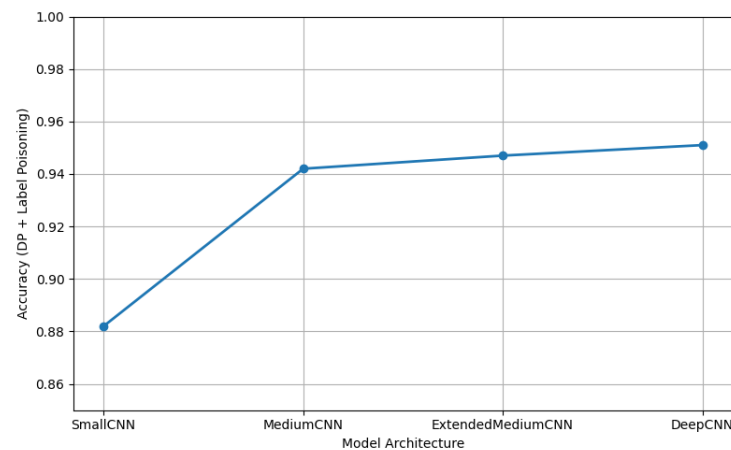


Figure 12. Impact of DP noise on model accuracy across varying architectures under label flipping attacks.

5. Conclusions and Future Scope

In this study, the main focus was the investigation of the effect of introduced DP noise on the detectability of poisoning attacks in federated learning. We considered data and model poisoning attacks on three datasets with different model architectures in the setup and analyzed the impact. Extensive experiments were conducted with varying ranges of malicious nodes, poisonous data, and DP noise levels (10% to 30% of malicious nodes, 5% to 100% of poisonous data, and DP ($\epsilon \in \{0.1, 1, 10\}, \delta = 10^{-5}$)). An analysis of the results indicated that the impact of the attack was proportional to the level of DP noise added and the percentage of malicious devices in the system. Furthermore, we proposed three algorithms for the detection of attacks and the identification of malicious devices in our simulated setup. Although the impact of attacks and the effectiveness of the proposed algorithms were visible on all datasets, the intensity of attack impacts and the levels of accuracy varied with respect to the dataset. We presented some mathematical analyses to better understand the asymptotic behavior of the proposed algorithms under

DP noise. The results confirmed the capabilities of the proposed algorithms in detecting attacks and identifying malicious nodes. Our results demonstrate that differential privacy noise does not significantly hinder the accuracy of attack detection or the identification of malicious clients. However, we observed that a massive increase in the magnitude of DP noise led to challenges in attack detection. This impact varied depending on the dataset characteristics and model architecture used. Therefore, a careful and context-aware selection of detection thresholds is essential to maintain robustness and reliability. Adaptive threshold tuning, aligned with the underlying data distribution and model complexity, is crucial for sustaining high detection performance in DP-integrated federated learning systems.

As part of future work, we aim to focus on extending the evaluation of the proposed detection framework to more sophisticated attack scenarios, such as backdoor traps, and assessing its robustness on non-iid datasets. A key direction will also be the differentiation between genuine data drift and malicious client drift, enabling more reliable and adaptive detection in dynamic environments.

Author Contributions: Conceptualization, C.A., D.G.N., J.A.M., J.J.N., and J.O.; Formal analysis, J.A.M.; Mathematical investigation, J.A.M.; Methodology, C.A., D.G.N., and J.A.M.; Software, C.A. and D.G.N.; Supervision, J.J.N. and J.O.; Validation, C.A. and D.G.N.; Visualization, D.G.N.; Writing—original draft, C.A., D.G.N., and J.A.M.; Writing—review and editing, C.A., J.A.M., J.J.N., and J.O. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are openly available in [IDS 2018] at [<https://www.unb.ca/cic/datasets/ids-2018.html>]; [CIC IoT dataset 2023] at: [<https://www.unb.ca/cic/datasets/iotdataset-2023.html>]; and [The MNIST Database], at [<https://ieeexplore.ieee.org/document/6296535>].

Conflicts of Interest: Chaitanya Aggarwal, Divya G. Nair and Jafar Mohammadi affiliate with Nokia, Section 3 of the paper have Algorithms 1–3 which are protected by Nokia with Patents. The Publication number WO2025/003839 A1 (original) Application number PCT/IB2024/056005 (original). The company has no role in the design of the study. Other authors declare no conflicts of interest.

References

1. Kairouz, P.; McMahan, H.B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A.N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; et al. Advances and open problems in federated learning. *Found. Trends[®] Mach. Learn.* **2021**, *14*, 1–210. [[CrossRef](#)]
2. Zhang, C.; Xie, Y.; Bai, H.; Yu, B.; Li, W.; Gao, Y. A survey on federated learning. *Knowl.-Based Syst.* **2021**, *216*, 106775. [[CrossRef](#)]
3. Dwork, C.; McSherry, F.; Nissim, K.; Smith, A. Calibrating Noise to Sensitivity in Private Data Analysis. In Proceedings of the Third Conference on Theory of Cryptography, New York, NY, USA, 4–7 March 2006; pp. 265–284. [[CrossRef](#)]
4. Dwork, C.; Talwar, K.; Thakurta, A.; Zhang, L. Analyze Gauss: Optimal Bounds for Privacy-preserving Principal Component Analysis. In Proceedings of the 46th Annual ACM Symposium on Theory of Computing, New York, NY, USA, 31 May–3 June 2014; pp. 11–20. [[CrossRef](#)]
5. Abadi, M.; Chu, A.; Goodfellow, I.; McMahan, H.B.; Mironov, I.; Talwar, K.; Zhang, L. Deep Learning with Differential Privacy. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016; ACM: New York, NY, USA, 2016. [[CrossRef](#)]
6. Wei, K.; Li, J.; Ding, M.; Ma, C.; Yang, H.H.; Farokhi, F.; Jin, S.; Quek, T.Q.; Poor, H.V. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 3454–3469. [[CrossRef](#)]
7. Sun, Z.; Kairouz, P.; Suresh, A.T.; McMahan, H.B. Can You Really Backdoor Federated Learning? 2019. Available online: <http://arxiv.org/abs/1911.07963> (accessed on 30 July 2025).
8. Bouacida, N.; Mohapatra, P. Vulnerabilities in Federated Learning. *IEEE Access* **2021**, *9*, 63229–63249. [[CrossRef](#)]
9. Koh, P.; Steinhardt, J.; Liang, P. Stronger Data Poisoning Attacks Break Data Sanitization Defenses. *arXiv* **2018**, arXiv:1811.00741. [[CrossRef](#)]

10. Liu, Y.; Ma, S.; Aafer, Y.; Lee, W.C.; Zhai, J.; Wang, W.; Zhang, X. Trojaning Attack on Neural Networks. In Proceedings of the Network and Distributed System Security Symposium, San Diego, CA, USA, 18–21 February 2018.
11. Steinhardt, J.; Koh, P.W.; Liang, P. Certified Defenses for Data Poisoning Attacks. 2017. Available online: <http://arxiv.org/abs/1706.03691> (accessed on 30 July 2025).
12. Raghunathan, A.; Steinhardt, J.; Liang, P. Certified Defenses Against Adversarial Examples. 2020. Available online: <http://arxiv.org/abs/1801.09344> (accessed on 30 July 2025).
13. Ma, Y.; Zhu, X.; Hsu, J. Data Poisoning Against Differentially-Private Learners: Attacks and Defenses. 2019. Available online: <http://arxiv.org/abs/1903.09860> (accessed on 30 July 2025).
14. IDS 2018 | Datasets | Research | Canadian Institute for Cybersecurity | UNB—unb.ca. Available online: <https://www.unb.ca/cic/datasets/ids-2018.html>. (accessed on 31 July 2023).
15. Neto, E.C.P.; Dadkhah, S.; Ferreira, R.; Zohourian, A.; Lu, R.; Ghorbani, A.A. CICIOT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment. *Sensors* **2023**, *23*, 5941. [\[CrossRef\]](#)
16. Deng, L. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Process. Mag.* **2012**, *29*, 141–142. [\[CrossRef\]](#)
17. Raza, A.; Li, S.; Tran, K.P.; Koehl, L. Using Anomaly Detection to Detect Poisoning Attacks in Federated Learning Applications. 2023. Available online: <http://arxiv.org/abs/2207.08486> (accessed on 30 July 2025).
18. Yang, Z.; Chen, M.; Wong, K.K.; Poor, H.V.; Cui, S. Federated learning for 6G: Applications, challenges, and opportunities. *Engineering* **2022**, *8*, 33–41. [\[CrossRef\]](#)
19. Nair, D.G.; Nair, J.J.; Reddy, K.J.; Narayana, C.A. A privacy preserving diagnostic collaboration framework for facial paralysis using federated learning. *Eng. Appl. Artif. Intell.* **2022**, *116*, 105476. [\[CrossRef\]](#)
20. Zhou, J.; Wu, N.; Wang, Y.; Gu, S.; Cao, Z.; Dong, X.; Choo, K.K.R. A differentially private federated learning model against poisoning attacks in edge computing. *IEEE Trans. Dependable Secur. Comput.* **2022**, *20*, 1941–1958. [\[CrossRef\]](#)
21. Chhetri, B.; Gopali, S.; Olapojoye, R.; Dehbash, S.; Namin, A.S. A Survey on Blockchain-Based Federated Learning and Data Privacy. *arXiv* **2023**, arXiv:2306.17338. [\[CrossRef\]](#)
22. Venkatasubramanian, M.; Lashkari, A.H.; Hakak, S. IoT Malware Analysis using Federated Learning: A Comprehensive Survey. *IEEE Access* **2023**, *11*, 5004–5018. [\[CrossRef\]](#)
23. Balakumar, N.; Thanamani, A.S.; Karthiga, P.; Kanagaraj, A.; Sathiyapriya, S.; Shubha, A. Federated Learning based framework for improving Intrusion Detection System in IIOT. *Network* **2023**, *3*, 158–179.
24. Li, S.; Cheng, Y.; Liu, Y.; Wang, W.; Chen, T. Abnormal client behavior detection in federated learning. *arXiv* **2019**, arXiv:1910.09933. [\[CrossRef\]](#)
25. Xing, L.; Wang, K.; Wu, H.; Ma, H.; Zhang, X. FL-MAAE: An Intrusion Detection Method for the Internet of Vehicles Based on Federated Learning and Memory-Augmented Autoencoder. *Electronics* **2023**, *12*, 2284. [\[CrossRef\]](#)
26. Sáez-de Cámara, X.; Flores, J.L.; Arellano, C.; Urbiet, A.; Zurutuza, U. Clustered federated learning architecture for network anomaly detection in large scale heterogeneous IoT networks. *Comput. Secur.* **2023**, *131*, 103299. [\[CrossRef\]](#)
27. Yazdinejad, A.; Dehghantanha, A.; Karimipour, H.; Srivastava, G.; Parizi, R.M. A robust privacy-preserving federated learning model against model poisoning attacks. *IEEE Trans. Inf. Forensics Secur.* **2024**, *19*, 6693–6708. [\[CrossRef\]](#)
28. Li, X.; Guo, Y.; Zhang, T.; Yang, Y. REC-Fed: A robust and efficient clustered federated system for dynamic edge networks. *IEEE Trans. Mob. Comput.* **2024**, *23*, 15256–15273. [\[CrossRef\]](#)
29. Zhang, Y.; Yang, Q.; Liu, J. FLDetector: Detecting adversarial clients in federated learning via update consistency. *arXiv* **2022**, arXiv:2207.09209.
30. Dou, Z.; Wang, J.; Sun, W.; Liu, Z.; Fang, M. Toward Malicious Clients Detection in Federated Learning (SafeFL). *arXiv* **2025**, arXiv:2505.09110.
31. Mu, X.; Cheng, K.; Shen, Y.; Li, X.; Chang, Z.; Zhang, T.; Ma, X. FedDMC: Efficient and Robust Federated Learning via Detecting Malicious Clients. *IEEE Trans. Dependable Secur. Comput.* **2024**, *21*, 5259–5274. [\[CrossRef\]](#)
32. Kavuri, L.A.; Mhatre, A.; Nair, A.K.; Gupta, D. SecureFed: A two-phase framework for detecting malicious clients in federated learning. *arXiv* **2025**, arXiv:2506.16458.
33. Cai, N.; Han, J. Privacy-Preserving Federated Learning against Malicious Clients Based on Verifiable Functional Encryption. *arXiv* **2025**, arXiv:2506.12846. [\[CrossRef\]](#)
34. Tasnim, N.; Mohammadi, J.; Sarwate, A.D.; Imtiaz, H. Approximating Functions with Approximate Privacy for Applications in Signal Estimation and Learning. *Entropy* **2023**, *25*, 825. [\[CrossRef\]](#) [\[PubMed\]](#)
35. Imtiaz, H.; Mohammadi, J.; Silva, R.; Baker, B.; Plis, S.M.; Sarwate, A.D.; Calhoun, V.D. A Correlated Noise-Assisted Decentralized Differentially Private Estimation Protocol, and its Application to fMRI Source Separation. *IEEE Trans. Signal Process.* **2021**, *69*, 6355–6370. [\[CrossRef\]](#) [\[PubMed\]](#)
36. Imtiaz, H.; Mohammadi, J.; Sarwate, A.D. Distributed Differentially Private Computation of Functions with Correlated Noise. 2021. Available online: <http://arxiv.org/abs/1904.10059> (accessed on 30 July 2025).

37. Dwork, C. Differential Privacy. In Proceedings of the Automata, Languages and Programming, Venice, Italy, 10–14 July 2006; pp. 1–12.
38. Dwork, C.; Roth, A. The Algorithmic Foundations of Differential Privacy. *Found. Trends Theor. Comput. Sci.* **2013**, *9*, 211–407. [[CrossRef](#)]
39. McSherry, F.; Talwar, K. Mechanism Design via Differential Privacy. In Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS '07), Providence, RI, USA, 21–23 October 2007; pp. 94–103. [[CrossRef](#)]
40. Chen, Z.; Yeo, C.K.; Lee, B.S.; Lau, C.T. Autoencoder-based network anomaly detection. In Proceedings of the 2018 Wireless telecommunications symposium (WTS), Phoenix, AZ, USA, 18–20 April 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–5.
41. Liu, P.; Xu, X.; Wang, W. Threats, attacks and defenses to federated learning: Issues, taxonomy and perspectives. *Cybersecurity* **2022**, *5*, 4. [[CrossRef](#)]
42. Kenney, J.F.; Keeping, E.S. *Mathematics of Statistics. Part Two*, 2nd ed.; D. Van Nostrand Company, Inc.: Princeton, NJ, USA, 1951.
43. Hasanpour, S.H.; Rouhani, M.; Fayyaz, M.; Sabokrou, M. Lets keep it simple, using simple architectures to outperform deeper and more complex architectures. *arXiv* **2016**, arXiv:1608.06037.
44. Authors, T.F. Flower: A Friendly Federated Learning Framework—flower.dev. Available online: <https://flower.dev/> (accessed on 31 July 2023).
45. Chicco, D.; Jurman, G. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genom.* **2020**, *21*, 6. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.