electronics

MDPI

Article

Defending Federated Learning from Collaborative Poisoning Attacks: A Clique-Based Detection Framework

Dimitrios Anastasiadis * and Ioannis Refanidis

Department of Applied Informatics, University of Macedonia, 546 36 Thessaloniki, Greece; yrefanid@uom.edu.gr
* Correspondence: danastas@uom.edu.gr

Abstract: Federated Learning (FL) systems are increasingly vulnerable to data poisoning attacks, in which malicious clients attempt to manipulate their training data in order to compromise the corresponding machine learning model. Existing detection techniques rely mostly on identifying clients who provide weight updates that significantly diverge from the average across multiple training rounds. In this work, we propose a Clique-Based Detection Framework (CBDF) that focuses on similarity patterns between client updates instead of their deviation. Specifically, we make use of the Euclidean distance to measure similarity between the weight update vectors of different clients over training iterations. Clients that provide consistently similar weight updates and exceed a predefined threshold are flagged as potential adversaries. Therefore, this method detects the coordination patterns of the attackers and uses them to strengthen FL systems against sophisticated, coordinated data poisoning attacks. We validate the effectiveness of this approach through extensive experimental evaluation. Moreover, we provide suggestions regarding fine-tuning hyperparameters to maximize the performance of the detection method. This approach represents a novel advancement in protecting FL models from malicious interference.

Keywords: federated learning; data poisoning; malicious client; coordinated attack detection; machine learning security

check for updates

Academic Editors: Sicong Shao and Jielun Zhang

Received: 7 April 2025
Revised: 7 May 2025
Accepted: 12 May 2025
Published: 15 May 2025

Citation: Anastasiadis, D.; Refanidis, I. Defending Federated Learning from Collaborative Poisoning Attacks: A Clique-Based Detection Framework. *Electronics* **2025**, *14*, 2011. https://doi.org/10.3390/electronics14102011

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) has become a groundbreaking concept regarding training machine learning models over decentralized datasets, since it allows the cooperation of multiple clients. It resolves certain important privacy issues of centralized techniques by enabling several clients to participate in the training of the model while maintaining the privacy of their local data [1]. However, FL entails several vulnerabilities in relation to data poisoning attacks, and this is mainly due to its decentralized nature. In data poisoning attacks, malicious clients intentionally inject corrupted data, modify their local data, or manipulate local training processes to alter the behavior of the performance of the global model or skew its outputs [2].

Data poisoning attacks pose significant challenges to the reliability and security of FL systems, especially in critical applications such as healthcare, finance, and autonomous driving. In such domains, the precision and robustness of machine learning models are paramount, as small perturbations in the training data or model behavior can lead to serious consequences, such as misdiagnoses in medical systems, fraudulent transactions in banking, or safety violations in autonomous vehicles. These sectors rely heavily on accurate predictions and consistent performance, which makes them particularly vulnerable to even subtle adversarial manipulation. Centralized oversight is often limited in FL; therefore,

trust in client contributions can become a critical point of failure. This underscores the need for robust and adaptive mechanisms to detect and mitigate coordinated poisoning attacks in FL environments [3].

Traditional defenses to data poisoning attacks in FL rely mainly on outlier detection mechanisms that flag clients who provide weight updates that deviate significantly from the average [4]. Such methods have been proven to be effective against isolated attackers, yet they may falter in the presence of coordinated adversaries. In such cases, attackers align their updates to mimic legitimate patterns, which can allow them to stealthily corrupt the global model and evade detection [5]. Robust aggregation methods typically aim to filter outliers and limit the influence of extreme updates, but they often struggle to detect more subtle or coordinated malicious contributions. Anomaly detection techniques based on statistical deviation or gradient similarity have shown promise in identifying malicious clients. However, many of these methods assess clients independently and fail to capture collaborative adversarial strategies. Moreover, adaptive weighting schemes can be applied to refine detection, but they generally operate on a per-round basis and lack the ability to track persistent patterns of coordination over time. Additionally, decentralized detection strategies can be limited by their reliance on local information, which can hinder their ability to identify globally coordinated behavior. These limitations highlight the need for more relational and temporally aware detection methods capable of identifying structured malicious activity across multiple rounds of federated training. A more detailed overview of related approaches and their specific drawbacks is provided in the subsequent section.

This research introduces the novel detection framework CBDF in order to overcome these limitations. The detection approach proposed here focuses on analyzing the similarity of weight updates between clients. In an earlier study, we introduced a method for detecting malicious clients in an FL system by computing the Euclidean distances between the weight update vectors of the clients across multiple training rounds. Then, the clients that produced highly similar updates were flagged as potentially malicious [6]. This research builds upon our previous work and expands on this foundation. We propose a new detection framework that identifies cliques of potentially coordinating clients and adjusts the malicious classification threshold dynamically according to the clique size. This method exploits the observation that coordinated attackers often synchronize their updates to amplify their impact [7]. Adversaries have often been shown to be able to effectively optimize and synchronize their updates to bypass common outlier-based defenses, which makes the detection of such data poisoning attacks increasingly challenging [8]. These findings confirm the importance of finding new detection mechanisms that go beyond deviation analysis and actively account for potential synchronization among malicious clients.

This similarity-based detection framework aims to enhance the robustness and security of FL systems by providing a reliable mechanism that can identify coordinated adversarial threats. We demonstrate the efficacy of our approach through systematic experimentation, in which we evaluate the performance of our proposed solution in detecting synchronized data poisoning attacks in FL systems. Furthermore, we offer guidelines and suggestions for hyperparameter tuning in order to balance detection sensitivity and false positive rates. This work represents an important step forward in securing FL systems, especially in areas where trust and integrity are vital.

The remainder of this article is organized as follows. Section 2 reviews existing research on data poisoning in FL, as well as current detection methodologies, setting the context for our study. Section 3 describes the proposed detection technique in detail and contains information on the experimental setup and the neural network architecture that was used. Section 4 presents the experimental results, evaluating the effectiveness and robustness of our method. Section 5 describes the implications of our findings, explains the setting and

impact of hyperparameter selection, and compares our approach with existing solutions. Finally, Section 6 concludes the article by summarizing our contributions to enhancing the security of FL systems and identifying challenges for future research.

2. Related Work

Early work on data poisoning attacks on FL systems highlighted their potential to degrade the performance of the global model by injecting malicious data into training datasets [9]. These attacks manipulate model updates such that the global model fails to generalize effectively, raising concerns about the robustness of FL.

An increasing amount of research is focused on understanding and categorizing data poisoning attacks in FL systems. Fang et al. [10] introduced local model poisoning attacks that directly manipulate model updates rather than the training data themselves. Their work demonstrated the ways attackers could modify their updates to optimize their adversarial objectives, showcasing the sophistication of modern poisoning strategies. Similarly, Bhagoji et al. [11] analyzed both targeted and untargeted model poisoning attacks and quantified their impact on FL systems using different aggregation mechanisms. Their work showed that even a relatively small number of malicious clients could significantly impact global model performance, especially if adversaries optimize their updates in such a way that they evade being detected.

Recent studies have expanded on these foundational efforts by proposing defense mechanisms designed to detect and mitigate data poisoning attacks. Xie et al. [12] investigated the "Fall of Empires" attack, where malicious clients undermine the global model by exploiting the aggregation process. Their findings emphasized the importance of robust aggregation techniques to minimize the influence of poisoned updates. Furthermore, Sun et al. [13] evaluated the resilience of FL systems against various poisoning strategies, and revealed several vulnerabilities that can be found in common federated optimization techniques.

Detecting data poisoning attacks in FL can be challenging due to the decentralized nature of the system and the lack of visibility into local data. Some early attempts to counter data poisoning attacks focused on robust aggregation techniques. One such technique is Krum, a Byzantine-resilient algorithm that was introduced by Blanchard et al. [14]. In this technique, the most reliable updates are selected in order to prevent the global model from being skewed by malicious contributions. Although such methods have proven to be effective in specific scenarios, they can often struggle to adapt to synchronized poisoning attacks in which adversarial clients act collaboratively to manipulate the behavior of the global model.

Furthermore, recent studies have explored dynamic thresholding mechanisms to support the detection of malicious updates. Zhang et al. [15] proposed the Dynamic Weight-Adjusted Mahalanobis Algorithm (DWAMA). This method dynamically adjusts thresholds to identify anomalies in weight updates. It effectively isolates poisoned updates while preserving the integrity of benign contributions. Similarly, Xu and Shu [16] developed a variance minimization method that reduces variance in aggregated weights, which led to an improvement in the robustness of the model against data poisoning.

Deng et al. [17] introduced a novel detection method named the Federated Learning Poisoning Defense System (FLPD), which is based on joint similarity to enhance the identification of diverse poisoning attacks in FL. Their technique computes two complementary similarity metrics between client model updates. These are the Euclidean distance, which captures the magnitude differences, and the cosine similarity, which assesses directional alignment. The method combines these two perspectives into a joint similarity score

and evaluates internal consistency among client updates. Clients that exhibit significant deviation in either dimension magnitude or direction are flagged as suspicious.

Jiang et al. [18] proposed a data-centric approach called Malicious Clients Detection Federated Learning (MCDFL), which was designed to counter label-flipping attacks. MCDFL does not focus solely on model updates, but rather evaluates the statistical quality of local datasets to detect anomalies that might indicate mislabeled data. The framework can effectively identify and mitigate the influence of malicious participants during the aggregation phase by monitoring deviations in data distributions across clients. The central server distributes a generator model and a global model to all active clients, who compute a data quality score (DQ) based on their local data. These scores are sent back to the server, where K-means clustering is used to distinguish between benign and potentially malicious clients. However, this method is limited by its reliance on the assumption that label distributions in honest clients will follow predictable patterns. That assumption may not hold in highly heterogeneous or class-imbalanced environments. Also, the detection depends critically on how good the generator is, and clustering adds non-trivial server overhead.

Birchman and Thamilarasu [19] proposed an enhanced defense mechanism for FL against collaborative poisoning attacks. Their approach, Area Similarity FoolsGold (ASF), extends the classic FoolsGold [4] method by incorporating geometric similarity measures. This results in a finer-grained analysis of client update patterns, and it allows for more precise detection of synchronized clients that have highly correlated gradients. More specifically, the approach in [19] relies on Triangle Area Similarity and Sector Area Similarity to capture the immediate geometric similarity of weight updates on a per-round basis. In contrast, our method employs Euclidean distance applied to the final-layer weight updates and focuses on identifying persistent similarity patterns over multiple training rounds. Therefore, CBDF shifts focus from per-round gradient alignment to persistent behavioral coordination over time. By analyzing weight update similarity across multiple rounds and constructing a client similarity graph, we identify cliques of clients that consistently behave in a coordinated manner. This is explained in detail and analyzed in the subsequent section. While [19] detects malicious clients based on strongly correlated updates in isolated rounds, our approach captures structured and sustained coordination among clients through graph-based analysis. Furthermore, the method in [19] utilizes a fixed threshold based on geometric similarity, whereas CBDF introduces an adaptive thresholding mechanism that adjusts dynamically based on the size of each detected clique. This makes the solution more flexible and allows for an accurate differentiation between coincidental similarity and intentional collusion. To our knowledge, this is the first method that combines temporal similarity tracking, graph-based clique analysis, and size-sensitive thresholding to detect synchronized poisoning attacks in FL.

You et al. [20] introduced Breakwater, a self-debiasing security framework for multi-hop edge FL. Their approach makes use of on-device malicious weight discriminators to locally evaluate and filter incoming model updates. Breakwater enables each participant to autonomously retain or discard suspicious weights, and therefore, it effectively mitigates model poisoning and ensures robust global model integrity.

Machine learning-based techniques are increasingly utilized to uncover subtle anomalies in federated model updates. Yan et al. [21] developed a proactive gradient analysis framework that uses predictive modeling to identify and isolate adversarial behaviors in FL systems. Their method demonstrated high efficacy in the detection of model poisoning attacks, reinforcing the importance of FL aggregation strategies that favor security and robustness. Purohit et al. [22] extended this line of work and introduced a poisoned data detector (PDD), which combines anomaly detection and predictive analytics in order to

distinguish between benign and malicious updates. These methods highlight the potential of integrating advanced learning techniques to increase the security of FL systems.

In addition, recent work has explored dynamic and adaptive defense strategies. Yue and Han [23] proposed FedDefense, an aggregation mechanism that is enhanced with reinforcement learning. It adjusts defense actions in real time based on the behavior of the participating clients. Their system dynamically penalizes dishonest contributors and rewards honest participants, making the system more robust against poisoning and collusion attacks in federated settings. Yin et al. [24] developed a robust aggregation algorithm capable of handling Byzantine failures, including data poisoning. This algorithm employs median and trimmed mean methods to exclude extreme updates, thus reducing the impact of malicious contributions. Their work provided theoretical foundations for optimal statistical performance and has since influenced numerous adaptive defense strategies in FL environments.

By reading through the most recent bibliography on the topic, it can be observed that prior work has mainly focused on robust aggregation mechanisms [14,24], anomaly detection [15], client-level statistical deviation [16], and geometric similarity in gradient space [19] to detect and mitigate poisoning attacks in FL environments. The method proposed in this article introduces a fundamentally different detection strategy based on relational analysis and structural coordination patterns. A key innovation of our detection method lies in the adaptive thresholding mechanism, which adjusts the sensitivity of detection based on the size of each detected clique. Along with the relational tracking of client behaviors over time, this adjustment enables our method to detect malicious clients that collectively perform synchronized data poisoning attacks. Such adversarial behavior would not be evident through simple outlier detection or aggregation rules.

In terms of robustness, CBDF does not rely solely on instantaneous similarity or isolated deviations, but instead accumulates evidence of coordinated behavior across rounds. This temporal and group-aware structure helps detection become more resilient to sophisticated adversaries that minimize their per-round deviation in order to evade detection. Our approach is less vulnerable to transient fluctuations than geometric similarity-based methods, which may be sensitive to noise or might fail to capture long-term patterns. From a practical standpoint, our detection strategy is model-agnostic and compatible with standard FL pipelines, and it does not require access to client data or training objectives. This makes it applicable to a wide range of real-world settings where privacy and scalability are critical.

3. Proposed Approach

In this section, we present the design and operational flow of CBDF, the method proposed to detect coordinated malicious clients that perform poisoning attacks in FL environments. We begin by providing an overview of the detection method, then outline the experimental setup, followed by a detailed explanation of the proposed method, and conclude with an exploration of hyperparameter fine-tuning. The subsequent section presents the results.

3.1. Overview of the Detection Pipeline

This method is based on the similarity of the weight updates across multiple communication rounds, enabling the detection of clients that exhibit persistent and coordinated behavior. The detection pipeline comprises the following steps:

1. Local Training and Weight Collection : In each communication round, all clients independently train a local model and submit their updated weights to the central server.

2. Pairwise Similarity Calculation: The central server computes the Euclidean distance between the final-layer weight updates of every pair of clients, quantifying their similarity.
3. Top-K Similarity Tracking: For each client, the K most similar clients (i.e., with the smallest distance values) are identified and recorded in every round.
4. Closeness Counter Construction: A matrix is maintained to count how frequently each pair of clients appears in each other's top-K lists across all rounds, forming the basis for assessing persistent coordination.
5. Threshold-Based Suspicion Marking: Clients that exceed a predefined coordination threshold based on their accumulated counts are flagged as potentially malicious.
6. Graph Construction and Clique Detection: A graph is constructed in which each node represents a suspicious client, and edges indicate strong coordination. Clique detection is then applied to uncover tightly knit groups of potentially colluding clients.
7. Adaptive Thresholding Based on Clique Size: To improve robustness, the threshold for confirming malicious behavior is adjusted dynamically based on the size of each detected clique. This prevents the misclassification of small random patterns as malicious while increasing sensitivity to larger coordinated groups.

Each of the steps of the detection method is described in detail in Section 3.3.

3.2. Datasets and Experimental Setup

Our experiments were carried out using two datasets sourced from Kaggle. The first is the Stroke Prediction dataset [25], which is designed to estimate the likelihood of a stroke based on various input features. The second dataset is designed for the detection of fraudulent bank transactions [26]. The two datasets were chosen to reflect the applicability and robustness of the proposed detection method in diverse and high-stakes real-world scenarios.

The Stroke Prediction dataset contains medical and demographic features that are used for the prediction of stroke occurrences. This dataset is particularly suitable for evaluating FL security in the healthcare domain, where model reliability is critical and adversarial behavior can have direct consequences on patient outcomes. Its binary classification structure and heterogeneous feature types simulate realistic client data distributions in federated settings; thus, it is a strong benchmark for detecting subtle poisoning attempts. The Fraudulent Bank Transactions dataset was selected to represent the financial sector, where data integrity and secure decision making are equally vital. The dataset includes a large number of real-world-like financial transactions with clearly labeled fraudulent and legitimate instances. Its high volume and dynamic structure make it ideal for simulating a multi-client federated environment with varying behaviors. In this way, we can assess the ability of our suggested detection method to identify synchronized malicious clients who aim to exploit financial systems by corrupting the model collaboratively. Together, these two datasets offer complementary evaluation scenarios, enabling us to test the generalizability and effectiveness of our approach across domains with different data characteristics and operational sensitivities.

The Stroke Prediction dataset contains 5110 instances, each characterized by the following attributes: *id* (unique identifier), *gender*, *age*, *hypertension* (status), *heart disease*, *ever_married*, *work_type*, *Residence_type*, *avg_glucose_level*, *bmi* (body mass index), *smoking_status*, and *stroke* (occurrence).

For this study, particular focus was given to the combination of *gender* and *work_type*, specifically in cases where this combination originally led to non-stroke outcomes.

The dataset was partitioned into 30 subsets, and each subset was assigned to a distinct client in the FL network. Among these, some selected clients were designated as malicious actors simulating a synchronized data poisoning attack by modifying specific data labels. These malicious clients attempted to perform a backdoor trigger attack and targeted instances where the attributes (*gender*: "Male", *work type*: "children" and *stroke*: 0) appeared, altering the *stroke* label from 0 to 1. In the original dataset, this attribute combination corresponded to 361 non-stroke cases.

A feedforward neural network (FNN) was employed as the learning model in this federated setting, implemented using TensorFlow's Keras API. The FNN was selected as the learning model for this study due to its simplicity, flexibility, and suitability for a wide range of structured data classification tasks. In the context of FL, models were trained independently across multiple decentralized clients, and FNNs offer a computationally efficient architecture that can be easily replicated and trained on resource-constrained devices. FNNs are well suited for tabular datasets such as those used in our experiments (i.e., structured medical and financial records), unlike convolutional or recurrent networks, which are mostly suitable for spatial or sequential data. Moreover, the goal of this study was to evaluate the effectiveness of the proposed malicious client detection method, rather than to optimize predictive performance. Hence, the interpretable and lightweight architecture of the FNN provides a controlled experimental setup, and it ensures that the detection results reflect malicious behavior rather than model complexity or variance in architectural performance. The structure used offers a balance between expressiveness and training stability, which makes it a practical and representative choice for federated environments.

The model architecture begins with an input layer tailored to the feature dimensions of the dataset, followed by three hidden layers with progressively smaller neuron counts: 32, 16, and 8. Each hidden layer uses the ReLU activation function to enable nonlinear learning while mitigating gradient-related issues. The output layer consists of a single neuron with a sigmoid activation function, which is suitable for binary classification such as this particular task of stroke prediction. The model was optimized using the Adam optimizer, known for its adaptability and effectiveness in handling sparse gradients, while binary crossentropy was used as the loss function since it is generally suitable for binary classification tasks. Model performance was primarily assessed using accuracy, providing a straightforward measure of predictive precision. Table 1 shows the structure of the neural network that was used for this experiment.

Table 1. Structure of the feedforward neural network used.

Layer	Input Size	Neurons	Activation	Parameters
Input Layer	12			
Hidden Layer 1		32	ReLU	$12 \times 32 + 32 = 416$
Hidden Layer 2		16	ReLU	$32 \times 16 + 16 = 528$
Hidden Layer 3		8	ReLU	$16 \times 8 + 8 = 136$
Output Layer		1	Sigmoid	$8 \times 1 + 1 = 9$
Total				1089

The FL process was conducted in 10 rounds. In each round, the central server distributed the current global model to all participating clients. Each client trained the model on its local subset for 10 epochs and then submitted weight updates to the central server. These updates were aggregated using an averaging technique and then redistributed to the clients in the next iteration.

A second experiment was conducted using the other aforementioned Kaggle dataset, namely the Fraudulent Transactions data. This dataset, sourced from a financial simulation, comprises 6,362,620 transactions with 10 attributes, providing a detailed representation of transactional behaviors over a simulated 30-day period. The objective of this FL training was to develop a model capable of detecting fraudulent transactions.

Each transaction in the dataset is characterized by a set of features: *step* is a unit of time measured in hours, with a total of 744 steps representing 30 days. The feature *type* is the type of transaction, categorized as *CASH-IN*, *CASH-OUT*, *DEBIT*, *PAYMENT*, or *TRANSFER*. *amount* is the feature that corresponds to the amount of transaction in the local currency. *nameOrig* is the identifier of the customer initiating the transaction. The features *oldbalanceOrg* and *newbalanceOrig* are the initiating customer's balance before and after the transaction respectively. *nameDest* corresponds to the identifier of the transaction recipient. *oldbalanceDest* and *newbalanceDest* are the balance of the recipient before and after the transaction respectively (both excluding merchants). Finally, *isFraud* is a binary label that indicates whether the transaction was fraudulent, while *isFlaggedFraud* is a binary flag that marks transactions as potentially illegal when they exceed 200,000 in a single transfer.

Similarly to the first experiment, the dataset was partitioned into 30 subsets, each assigned to a distinct client in the FL network. This time, the selected malicious clients specifically targeted transactions where *type* = "PAYMENT", *isFlaggedFraud* = 0, and *isFraud* = 0, and modified the *isFraud* label from 0 to 1. This alteration introduced a backdoor trigger to the model, which can potentially disrupt the ability of the model to accurately differentiate fraudulent from legitimate activities, especially for transactions that follow the above conditions.

The same feedforward neural network architecture described for the first experiment was used for this experiment. Also, the FL process followed the same methodology as the stroke prediction experiment.

3.3. Detecting Malicious Clients

The proposed method aims to detect synchronized malicious clients in an FL system by analyzing the weight update vectors from the local model training. Specifically, the approach examines the final layer of the neural network. Changes in this layer can give the most indications of any important alterations that might take place in the behavior of the model, since it has direct influence on output predictions.

Overall, the detection method consists of weight update similarity analysis, threshold-based identification of potentially coordinating clients, graph-based clique detection of malicious groups, and adaptive threshold adjustment using a linear decay factor. The process is described in detail below.

In each communication round, each client submits weight updates to the central server. The Euclidean distance between every pair of clients' final-layer weight updates is computed as

$$\text{dist}(i,j) = \sqrt{\sum_{k=1}^{d} \left(w_i^{(k)} - w_j^{(k)} \right)^2} \tag{1}$$

where $w_i^{(k)}$ represents the k-th component (parameter) of the final-layer weight vector for client (i). The sum runs over all d dimensions, capturing the total squared difference between corresponding weights.

The K closest clients, defined as those with the smallest Euclidean distances, are then identified for each client. Here, K represents the number of closest clients considered for each client based on their weight updates. To determine whether a client is potentially

malicious, a closeness counter matrix $closeness_counter(i, j)$ records the number of rounds in which two clients (i) and (j) appear in the (K) closest client lists of each other. N denotes the total number of clients participating in the training process, which remains constant throughout each individual experiment. After the training is completed, the client (i) is flagged as suspicious if

$$closeness_counter(i, j) > t \tag{2}$$

where t is the coordination threshold. It is defined as

$$t = \left(1 - \frac{(1 - K/N)}{10}\right) \times num_rounds \tag{3}$$

where num_rounds is the total number of rounds of the model training. The constant value 10 in the denominator was empirically determined to yield optimal results across different values of K, tested in the range $4 \leq K \leq 12$, and for different numbers of malicious clients, tested in the range $3 \leq M \leq 14$. A client that frequently appears in another client's closest list beyond the threshold t is flagged as potentially malicious. This step serves as a first estimation of suspicious clients and it is required in order to proceed to the next step.

Afterwards, to further analyze potentially malicious clients, a graph representation G is constructed. In the graph, nodes represent flagged clients and edges exist between two clients if their closeness counter exceeds t, indicating strong alignment in weight update contributions. Then, we apply clique detection on that graph in order to extract fully connected subgroups. Within a clique C, every client is highly coordinated with all others.

To refine detection accuracy, an adaptive threshold adjustment is applied based on clique size. This ensures that larger malicious groups are more likely to be detected. The base threshold for the smallest cliques (i.e., pairs of clients) is defined as

$$t_{\text{base}} = \left(1 - \frac{(1 - K/N)}{15}\right) \times num_rounds \tag{4}$$

Similarly to the previous formula for the initial coordination threshold, the constant value 15 in the denominator was empirically determined as the optimal choice based on experiments with different values of K and numbers of malicious clients. The aforementioned ranges of these parameters apply here as well. Although these constants in the threshold definitions were determined empirically based on extensive experimentation, the development of a data-driven or adaptive thresholding mechanism is left as an important direction for future work to enhance generalization across diverse FL environments. For larger cliques, a linear decay factor is applied to the threshold:

$$t_{\text{adjusted}} = t_{\text{base}} \times \left(1 - \frac{|C|}{N}\right) \tag{5}$$

where $|C|$ is the size of the detected clique with regards to the number of nodes it contains. Since N is treated as a constant throughout each individual experiment, the threshold depends primarily on clique size. If the clique size increases, the threshold decreases, making it easier to flag larger colluding groups. Conversely, if the clique size decreases, the threshold remains closer to t_{base}, ensuring that small, possibly random correlations are not misclassified as malicious activity. The reason for this threshold adjustment is due to the fact that larger cliques are less likely to appear randomly, and are therefore considered to be more highly associated with coordinated attacks. Thus, we evaluate larger cliques more stringently, while smaller cliques are treated with more caution to avoid high false positive rates. This dynamic thresholding approach effectively differentiates genuine model similarities from coordinated adversarial behavior.

For each clique found, the coordination counts of its members are checked against their respective adjusted threshold. For a pair of clients (i) and (j), if at least one of the conditions $closeness_counter(i, j) > t$ or $closeness_counter(j, i) > t$ is satisfied, then both clients are identified as malicious. If any malicious clients are found, they are reported. Otherwise, the system concludes that no significant collusion has been detected.

Algorithm 1 starts by defining the input parameters, which include the weight update history of all clients across multiple rounds, the number of closest clients to consider (K), the total number of training rounds, and the total number of clients.

Afterwards, the essential data structures are initialized (lines 4–7). The distance matrix is set to zero to store pairwise Euclidean distances between client weight updates across training rounds. An empty list is prepared to store the K closest clients for each client in each round. The closeness counter matrix is initialized to track how often clients appear in the closest client lists of each other, and an empty list is created to store the final set of malicious clients.

Next, the Euclidean distance between each pair of clients in each training round is computed (lines 9–20). The algorithm iterates over all rounds, and for each client (i), it computes the Euclidean distance between its weight update and the weight updates of all other clients (j) (where $i \neq j$). Once all distances are computed and stored in the distance matrix, the algorithm identifies the K closest clients for each client based on the smallest distance values and stores their indices.

Moving forward, the algorithm counts how frequently each client appears in the list of closest clients K of another client throughout all training rounds (lines 22–29). This frequency is stored in the closeness counter matrix, providing a quantitative measure of coordination between clients.

The coordination threshold is computed (line 31). This threshold determines the minimum frequency required for a client to be flagged as potentially malicious.

Then, the coordination threshold is applied to identify potentially malicious clients (lines 32–38). If two clients appear in the list of the closest clients K of each other more frequency than the threshold, they are added to the list of potentially malicious clients.

A graph G representing the relationships between potentially malicious clients is constructed (lines 39–41). Each node in the graph corresponds to a flagged client, and an edge exists between two nodes if their coordination frequency exceeds the threshold.

Afterwards, the base threshold for clients that are in cliques of size 2 (pairs) is computed (line 43).

The threshold is adjusted for larger cliques (lines 44–45). This ensures that larger colluding groups receive a lower threshold, making them easier to detect, while smaller groups retain a higher threshold to avoid false positives.

Malicious clients within the identified cliques are detected (lines 46–52). If two clients exceed the adjusted threshold in the closeness counter matrix, they are added to the final list of malicious clients.

Finally, the list of detected malicious clients is returned (line 55). If no clients meet the detection criteria, the list remains empty, indicating that no coordinated adversarial activity was found.

Algorithm 1 CBDF: Detect Synchronized Malicious Clients

Input: weight_updates[clients, rounds], K, num_rounds, num_clients
Output: final_malicious_clients_list

distance_matrix[num_clients, num_clients, num_rounds] = 0
closest_clients[num_clients, num_rounds] = []
closeness_counter[num_clients, num_clients] = 0
malicious_clients_list = []

for each round r from 1 to num_rounds **do**
    for each client i from 1 to num_clients **do**
        for each client j from 1 to num_clients **do**
            if $i \neq j$ **then**
                Compute Euclidean distance $d(i, j)$ between weight_updates[i, r] and weight_updates[j, r]
                Store $d(i, j)$ in distance_matrix[i, j, r]
            end if
        end for
        Identify K closest clients based on smallest distances in distance_matrix[i, :, r]
        Store indices of these K closest clients in closest_clients[i, r]
    end for
end for

for each client i from 1 to num_clients **do**
    for each client j from 1 to num_clients **do**
        if $i \neq j$ **then**
            Count occurrences where j appears in closest_clients[i] over all rounds
            Store count in closeness_counter[i, j]
        end if
    end for
end for

$t = \left(1 - \frac{(1-K/num_clients)}{10}\right) \times num_rounds$
for each client i from 1 to num_clients **do**
    for each client j from 1 to num_clients **do**
        if closeness_counter[i, j] $> t$ **then**
            Add i and j to potential_malicious_clients_list
        end if
    end for
end for
Construct graph G where:
  Nodes represent clients in potential_malicious_clients_list
  Edges exist if closeness_counter[i, j] $> t$

$t_{\text{base}} = \left(1 - \frac{(1-K/num_clients)}{15}\right) \times num_rounds$
for each clique C_k in G **do**
    $t_{\text{adjusted}} = t_{\text{base}} - \left(t_{\text{base}} \times \frac{|C_k|}{num_clients}\right)$
    for each client i in C_k **do**
        for each client j in C_k **do**
            if closeness_counter[i, j] $> t_{\text{adjusted}}$ **then**
                Add i and j to final_malicious_clients_list
            end if
        end for
    end for
end for

return final_malicious_clients_list

4. Results

To evaluate the effectiveness and robustness of the proposed CBDF method, we conducted experiments in which we used the two datasets described in Section 3.2. The goal was to assess the ability of the method to detect malicious clients under different attack intensities and configurations. The key parameters varied in these experiments were the number of closest clients K and the number of malicious clients. The number of closest clients K was tested in the range of 4 to 8, while the number of malicious clients varied from 3 to 7. These ranges were selected to reflect both mild and more aggressive adversarial scenarios, and to evaluate the sensitivity of the method to the choice of K, which directly impacts detection performance and false positives.

4.1. Performance Evaluation

Each experiment comprised multiple training rounds in which the detection algorithm monitored client weight updates and identified coordinated behavior using the closeness threshold. The objective was to correctly detect all malicious clients while minimizing false positives. Performance was evaluated using the accuracy, precision, recall, and F1-Score. Each of these metrics captures a different aspect of the effectiveness of the method. Accuracy reflects the overall correctness of the classification, as it measures how many clients were correctly identified as either malicious or benign. Precision is particularly important in our experiment, as it indicates the proportion of clients flagged as malicious that were indeed malicious. This helps minimize false accusations and maintain trust in the system. Recall measures the ability of the method to detect actual malicious clients without misclassifying them as benign (false negatives), which is critical for ensuring the security of the global model. F1-Score combines precision and recall into a single metric to provide a balanced assessment, which is especially useful when dealing with imbalanced classes. Possible additional metrics that could be incorporated into future work are the false positive rate (FPR) and the false negative rate (FNR), which could help quantify specific types of classification errors. In addition, the Area Under the ROC Curve (AUC) would offer a threshold-independent view of the discriminative ability of the detection method.

The results for both datasets are presented in Tables 2 and 3.

The overall performance of the detection method across all tested configurations for the Fraudulent Bank Transactions dataset resulted in the following overall metrics:

$$\text{Accuracy} = 0.86, \quad \text{Precision} = 0.63, \quad \text{Recall} = 0.94, \quad \text{F1-Score} = 0.75$$

For the Stroke Prediction dataset, it resulted in the following:

$$\text{Accuracy} = 0.96, \quad \text{Precision} = 0.89, \quad \text{Recall} = 0.96, \quad \text{F1-Score} = 0.92$$

Thus, the overall performance across both datasets can be reflected as follows:

$$\text{Accuracy} = 0.91, \quad \text{Precision} = 0.76, \quad \text{Recall} = 0.95, \quad \text{F1-Score} = 0.84$$

The evaluation of the proposed CBDF included a performance comparison with two recent and relevant detection techniques: the FLPD method [17] and the MCDFL method [18], which were described in Section 2. All methods were tested under the same experimental conditions using both the Fraudulent Bank Transactions and Stroke Prediction datasets. Table 4 summarizes the average results across both datasets for each method.

Table 2. Performance results for Fraudulent Bank Transactions dataset.

K	Malicious Clients	Accuracy	Precision	Recall	F1-Score
4	3	0.87	0.43	1.00	0.60
5	3	0.93	0.60	1.00	0.75
6	3	0.93	0.60	1.00	0.75
7	3	0.86	0.43	1.00	0.60
8	3	0.93	0.60	1.00	0.75
4	4	0.96	0.80	1.00	0.89
5	4	0.87	0.50	1.00	0.67
6	4	0.87	0.50	1.00	0.67
7	4	0.77	0.36	1.00	0.53
8	4	0.73	0.33	1.00	0.50
4	5	1.00	1.00	1.00	1.00
5	5	0.86	0.55	1.00	0.71
6	5	0.80	0.45	1.00	0.63
7	5	0.80	0.45	1.00	0.63
8	5	0.80	0.45	1.00	0.63
4	6	0.97	1.00	0.83	0.90
5	6	1.00	1.00	1.00	1.00
6	6	0.90	0.67	1.00	0.80
7	6	0.93	0.75	1.00	0.86
8	6	0.66	0.38	1.00	0.55
4	7	0.77	0.50	0.29	0.37
5	7	1.00	1.00	1.00	1.00
6	7	1.00	1.00	1.00	1.00
7	7	0.77	0.50	1.00	0.67
8	7	0.60	0.37	1.00	0.54

Table 3. Performance results for Stroke Prediction dataset.

K	Malicious Clients	Accuracy	Precision	Recall	F1-Score
4	3	1.00	1.00	1.00	1.00
5	3	1.00	1.00	1.00	1.00
6	3	0.90	0.50	1.00	0.67
7	3	0.93	0.60	1.00	0.75
8	3	0.93	0.60	1.00	0.75
4	4	1.00	1.00	1.00	1.00
5	4	1.00	1.00	1.00	1.00
6	4	1.00	1.00	1.00	1.00
7	4	1.00	1.00	1.00	1.00
8	4	1.00	1.00	1.00	1.00
4	5	0.90	1.00	0.40	0.57
5	5	1.00	1.00	1.00	1.00
6	5	1.00	1.00	1.00	1.00
7	5	1.00	1.00	1.00	1.00
8	5	0.86	0.56	1.00	0.71
4	6	0.90	1.00	0.50	0.67
5	6	0.93	0.75	1.00	0.86
6	6	0.90	1.00	0.50	0.67
7	6	1.00	1.00	1.00	1.00
8	6	0.83	0.55	1.00	0.70
4	7	0.87	0.71	0.71	0.71
5	7	1.00	1.00	1.00	1.00
6	7	1.00	1.00	1.00	1.00
7	7	1.00	1.00	1.00	1.00
8	7	0.93	0.78	1.00	0.86

Table 4. Performance comparison of CBDF with FLPD and MCDFL methods.

Method	Accuracy	Precision	Recall	F1-Score
FLPD	0.87	0.67	0.87	0.76
MCDFL	0.90	0.83	0.71	0.76
CBDF	0.91	0.76	0.95	0.84

It can be observed that the proposed method achieves the highest overall performance, particularly in terms of recall and F1-Score. This indicates its capability to identify malicious clients without significantly increasing false positives for the given setup and attack scenario. In the Fraudulent Bank Transactions dataset, CBDF achieves high recall, which means that most malicious clients were detected, but at the cost of increased false positives, which led to moderate precision. The Stroke Prediction dataset shows better balance, with high precision, recall, and F1-Score, indicating strong overall performance.

These findings highlight the importance of parameter selection. Lower values of K tend to miss some malicious clients (lower recall), while higher values of K may increase false positives (lower precision). Balancing precision, recall, and F1-Score is crucial for optimizing detection performance in FL environments.

4.2. Statistical Significance Test

The robustness of the proposed detection method was further validated by a statistical significance test. This served the purpose of determining whether the observed performance results were consistent and not due to random variation. The Wilcoxon signed-rank test was selected as an appropriate statistical significance test, since the experiments involved paired observations, which means that the same experimental conditions (i.e., random seeds and dataset splits) were used to evaluate the proposed CBDF, FLPD, and MCDFL methods. This test is particularly suitable when the normality assumption for paired differences cannot be guaranteed and is therefore an appropriate choice for FL experiments where variance between runs may not follow a normal distribution.

The evaluation was performed over 10 independent experimental runs. For each run, the datasets were split differently among the clients, and the full training and detection process was executed for all three methods. The average performance across both datasets (Fraudulent Bank Transactions and Stroke Prediction) was considered for each run, focusing on F1-Score as the primary performance metric due to its balanced measurement of precision and recall, which is critical in malicious client detection.

The results of the Wilcoxon signed-rank test demonstrated that the proposed CBDF outperformed both FLPD and MCDFL. More specifically, the p-value for the comparison between the proposed method and FLPD was 0.003906, and similarly, the p-value for the comparison between the proposed method and MCDFL was 0.013671. Both values are well below the commonly accepted significance level of 0.05, confirming that the improvements offered by the proposed method are statistically significant and not due to random fluctuations.

5. Discussion

The effectiveness of the proposed detection method is influenced by several key factors. Among them are the number of closest clients considered (K) and the number of training rounds before detection is applied. The choice of these parameters directly impacts the sensitivity and precision of the method to detect coordinated malicious clients while minimizing false positives.

Parameter K was empirically determined to balance the ability of the detection algorithm to accurately capture clients that perform synchronized data poisoning attacks as well as to avoid unnecessary false positives. By setting K higher than the expected number of malicious clients, the method ensures a broader perspective on monitoring client interactions, and therefore increases the likelihood of detecting subtle or dispersed malicious behaviors. This is particularly important in scenarios with a large number of total clients. In such scenarios, setting K too low might result in missing some malicious clients. For instance, in a federated network with 100 clients that includes five known malicious ones, setting K to 6 rather than 3 significantly increases the chances of detecting all malicious clients, as the detection process incorporates a wider range of interactions among clients.

The optimal setting of K and its effect on detection efficiency in various federated environments remain an open research question. Suggested future work includes the systematic analysis of how different values of K influence detection accuracy, particularly in FL environments with various levels of adversarial activity. This exploration will enhance the adaptability of the detection mechanism and provide guidance in selecting appropriate K values in diverse real-world scenarios.

Another important consideration is the number of training rounds (*num_rounds*) before applying the detection mechanism. In the current implementation, malicious client detection takes place after all training rounds have been completed. However, an alternative approach would be to apply detection periodically or in earlier stages of training to allow for real-time intervention and countermeasure deployment. By evaluating client behaviors in the initial portion of the training rounds, detection could be applied midway through the training process. This would provide the opportunity to react to potential threats in real time instead of waiting until the end of the training, and therefore mitigate the impact of attacks before they significantly alter the federated model.

The proposed detection method offers several advantages in comparison to previous poisoning detection techniques in FL. Traditional methods rely mainly on anomaly detection using statistical deviation analysis, geometric similarity in gradient space, or model divergence metrics. These techniques can be vulnerable to adversarial adaptation, as malicious clients can manipulate their updates and appear statistically similar to benign clients in order to avoid detection. In contrast, the method introduced in this article uses a relational analysis of weight updates and identifies groups of clients that consistently provide similar updates over multiple rounds. This strategy allows for the detection of subtler and more sophisticated attacks that traditional outlier-based detection techniques would not be able to spot.

Furthermore, the proposed approach dynamically checks similarities in weight updates between clients, unlike threshold-based detection mechanisms that depend on absolute deviation values, and evaluate each client individually. Our method examines weight update similarity across multiple clients in multiple rounds, providing greater robustness against distributed poisoning strategies. The characteristic of such poisoning strategies is that multiple malicious clients attempt to blend in and appear as benign by modifying their updates incrementally rather than drastically. In addition, the proposed method integrates clique-based clustering, which enhances the detection of synchronized adversarial behaviors, making it particularly effective in environments where attackers operate in groups.

Based on the comparative evaluation presented in the previous section, specifically against the recent relevant approaches FLPD and MCDFL, we observed that the CBDF method outperformed them across multiple dataset splits. It achieved an F1-Score of about 84%, while both FLPD and MCDFL scored about 76%. This improvement may be

due to the ability of the CBDF method to capture persistent coordination patterns over multiple rounds and its adaptive thresholding mechanism, which adjusts based on the size of detected cliques. Additionally, in the Related Work section, we mentioned methods such as Area Similarity FoolsGold (ASF) [19] and Breakwater [20], which focus on per-round geometric similarity and decentralized client-level filtering, respectively. CBDF focuses on relational and temporal analysis through the graph-based modeling of client behavior. This design enables the detection of more structured and coordinated adversarial strategies that may not be identifiable through single-round or client-local evaluations.

While the proposed detection method demonstrates strong performance across diverse scenarios, it still includes several limitations. First, the approach assumes that malicious clients coordinate in a way that creates detectable similarity patterns in their weight updates. In other scenarios, more advanced attackers could reduce the effectiveness of the detection by disguising their behavior or injecting randomized noise. Additionally, the method relies on a fixed number of clients throughout training, and its performance under dynamic participation (e.g., clients joining or dropping out) remains unexplored. Although the adaptive thresholding mechanism is effective, it is currently empirically calibrated and may require tuning for new datasets or deployment environments. Furthermore, the method focuses on the detection of malicious behavior rather than directly mitigating it. This means that it should be combined with defense strategies for a complete security framework.

Another important aspect is the scalability of the proposed method in large-scale FL environments that involve hundreds or thousands of clients. The combination of relational analysis based on pairwise Euclidean distances and graph-based clique detection offers high detection effectiveness. However, it naturally introduces computational complexity that grows exponentially with the number of clients. As the system scales, both memory usage and processing time can become significant bottlenecks. Efficient data structures, as well as heuristic approaches, can be adopted to reduce this overhead. Similarly, the communication overhead associated with collecting and processing client updates could increase substantially in larger federated settings.

Overall, the findings of our work highlight the need for continued refinement and evaluation of data poisoning detection strategies in FL environments. It is valuable to investigate how parameters such as K, training rounds, number of clients, and detection timing can influence the detection performance of the method. Such a systematic study can support future research to develop more adaptive and resilient solutions that secure FL systems against malicious client behaviors.

6. Conclusions and Future Work

This work presented a novel method for detecting coordinating malicious clients that attempt to perform data poisoning attacks in FL environments. The proposed method includes the analysis of weight update similarities and leverages graph-based clique detection. It effectively identifies groups of synchronized malicious clients while attempting to minimize false positives. The method was evaluated on two datasets, the first concerning Fraudulent Bank Transactions and the other Stroke Prediction. It demonstrated robust detection capabilities, yielding promising results across all tested performance metrics on the Stroke Prediction dataset and a high accuracy and recall on the Fraudulent Bank Transactions dataset.

The results show that the choice of K (the number of closest clients considered) plays a crucial role in the detection performance. Larger values of K improve the sensitivity of the method to coordinated attacks, but can also increase false positives if set too high. The method integrates clique-based analysis and adjusts the detection threshold dynamically based on the size of the detected cliques. This refinement significantly enhances

the robustness of the detection method by reducing false positives in smaller cliques of potentially malicious clients and strengthening the detection sensitivity in larger ones.

With regard to future work, several directions can further enhance the proposed detection mechanism. The investigation of adaptive strategies to dynamically adjust K and detection thresholds based on real-time network behavior can support the reduction in dependence on empirical tuning and improve detection performance. The replacement of currently empirically determined constants within the threshold formulas with a more data-driven and adaptive mechanism is also an essential next step. We acknowledge that manual tuning may not generalize optimally to all FL settings, particularly in highly heterogeneous or large-scale environments. Developing an adaptive thresholding approach could involve dynamically adjusting threshold values based on observed client behavior patterns, clustering characteristics, or historical statistics during training. This would enhance the generalization capability of the detection method and help it become more robust and autonomous in practical deployments without requiring manual intervention. Furthermore, incorporating principles from statistical methods, such as the Bonferroni correction, could help decrease false positive rates when evaluating multiple client pair interactions. In this way, the adaptive thresholding mechanism will be able to remain statistically sound in the presence of multiple comparisons.

In addition, the implementation of real-time detection strategies would allow malicious clients to be identified during training, and therefore, it would mitigate adversarial influence before the model converges. Attackers can often apply more sophisticated adversarial techniques that might involve the strategic modifications of their weight updates in order to evade detection. Such cases may be another important avenue for exploration, as they can offer insights that could lead to the enhancement of the robustness of poisoning detection mechanisms. Moreover, it is essential to scale the method to FL environments that involve hundreds or thousands of clients. This can help ensure that detection remains computationally efficient and is applicable in real-world scenarios. Future research will focus on optimizing the computational efficiency of the detection method. A systematic evaluation of computational complexity, memory requirements, and communication overhead as a function of the number of clients will also be conducted. This will allow a clearer quantification of the trade-offs and help guide the deployment of the method in real-world, large-scale FL systems where efficiency is as critical as detection accuracy. To complement the above, further research should explore how adversarial clients and their weight updates are handled after being identified as malicious. For example, there might be additional mechanisms that reweigh their contributions, exclude them from aggregation, or apply model repair techniques to counteract their influence.

Furthermore, invaluable insights can be extracted from a deeper statistical analysis on the formation of cliques and their likelihood of occurring due to random coincidence. Based on our experimentation, some identified cliques do not necessarily represent actual coordinated adversarial behavior, and could therefore lead the mechanism to produce false positives. In FL training, there is a degree of inherent randomness in weight updates due to the diversity of client data and independent local training. This can result in benign clients being clustered together purely based on chance, which might lead the system to incorrectly classify them as malicious. By applying statistical significance testing, we aim to refine the classification process, so that we enhance the ability of the detection algorithm to distinguish between similarities that occur naturally and truly malicious coordination. In this way, the overall performance of the detection method will be improved. The exploration of such statistical patterns will help us take more informed and reliable approaches to adversarial client detection, making the method more robust and adaptable to real-world scenarios.

This study underscores the significance of analyzing interactions between clients to improve the security of FL systems. The proposed detection method effectively identifies synchronized malicious clients using weight update analysis and a clique-based detection technique. Further advances in adaptive detection strategies, proactive intervention, and robust mitigation techniques will be critical in ensuring the long-term reliability of FL frameworks against evolving adversarial threats.

Author Contributions: Conceptualization, D.A. and I.R.; methodology, D.A. and I.R.; software, D.A.; validation, D.A.; formal analysis, I.R.; investigation, D.A.; resources, D.A. and I.R.; data curation, D.A.; writing—original draft preparation, D.A.; writing—review and editing, D.A. and I.R.; visualization, D.A.; supervision, I.R.; project administration, D.A. and I.R.; funding acquisition, not applied. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding. The APC was waived by the publisher.

Data Availability Statement: The data that was used for the experiments of this research can be found in the cited sources [25,26].

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. McMahan, H.; Moore, E.; Ramage, D.; Hampson, S.; Agüera y Arcas, B. Communication-efficient learning of deep networks from decentralized data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS), PMLR, Fort Lauderdale, FL, USA, 20–22 April 2017.
2. Kairouz, P.; McMahan, H.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; et al. *Advances and Open Problems in Federated Learning*; Now Foundations and Trends: Norwell, MA, USA, 2019.
3. Xie, C.; Huang, K.; Chen, P.; Li, B. DBA: Distributed Backdoor Attacks against Federated Learning. In Proceedings of the International Conference on Learning Representations (ICLR), Addis Ababa, Ethiopia, 30 April 2020.
4. Fung, C.; Yoon, C.; Beschastnikh, I. Mitigating Sybils in federated learning poisoning. *arXiv* **2018**, arXiv:1808.04866.
5. Sun, W.; Gao, B.; Xiong, K.; Wang, Y. A GAN-based data poisoning attack against federated learning systems and its countermeasure. *arXiv* **2024**, arXiv:2405.11440.
6. Anastasiadis, D.; Refanidis, I. Enhancing security in federated learning: Detection of synchronized data poisoning attacks. In *Proceedings of the Artificial Intelligence: Methodology, Systems, and Applications*; Lecture Notes in Computer Science; AIMSA 2024; Springer: Berlin/Heidelberg, Germany, 2024; Volume 15462. [CrossRef]
7. Tolpegin, V.; Truex, S.; Gursoy, M.; Liu, L. Data poisoning attacks against federated learning systems. In *Proceedings of the Computer Security–ESORICS 2020*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 480–501.
8. Shejwalkar, V.; Houmansadr, A. Manipulating the Byzantine: Optimizing model poisoning attacks and defenses for federated learning. In Proceedings of the Network and Distributed System Security Symposium (NDSS), Virtual, 21–25 February 2021.
9. Biggio, B.; Nelson, B.; Laskov, P. Poisoning attacks against support vector machines. In Proceedings of the International Conference on Machine Learning, Boca Raton, FL, USA, 12–15 December 2012.
10. Fang, M.; Cao, X.; Jia, J.; Gong, N. Local model poisoning attacks to Byzantine-robust federated learning. In Proceedings of the USENIX Security Symposium, Boston, MA, USA, 12–14 August 2020.
11. Bhagoji, A.; Chakraborty, S.; Mittal, P.; Calo, S. Analyzing federated learning through an adversarial lens. In Proceedings of the International Conference on Machine Learning (ICML), PMLR, Long Beach, CA, USA, 9–15 June 2019.
12. Xie, C.; Koyejo, O.; Gupta, I. Fall of empires: Breaking Byzantine-tolerant SGD by inner product manipulation. In Proceedings of the Advances in Neural Information Processing Systems, Online, 6–12 December 2020.
13. Sun, J.; Kairouz, P.; Suresh, A.; McMahan, H. Can you really backdoor federated learning? *arXiv* **2019**, arXiv:1911.07963.
14. Blanchard, P.; El Mhamdi, E.; Guerraoui, R.; Stainer, J. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*; Curran Associates, Inc.: Long Beach, CA, USA, 2017.
15. Zhang, G.; Liu, H.; Yang, B.; Feng, S. DWAMA: Dynamic weight-adjusted Mahalanobis defense algorithm for mitigating poisoning attacks in federated learning. *Peer-to-Peer Netw. Appl.* **2024**, *17*, 3750–3764. [CrossRef]
16. Xu, H.; Shu, T. Defending against model poisoning attack in federated learning: A variance-minimization approach. *J. Inf. Secur. Appl.* **2024**, *82*, 103744. [CrossRef]
17. Deng, J.; Liu, S.; Li, C. Detecting Diverse Poisoning Attacks in Federated Learning Based on Joint Similarity. In Proceedings of the 16th International Conference on Wireless Communications and Signal Processing (WCSP), Hefei, China, 24–26 October 2024; IEEE: New York, NY, USA, 2024; pp. 133–138. [CrossRef]

18. Jiang, Y.; Zhang, W.; Chen, Y. Data Quality Detection Mechanism Against Label Flipping Attacks in Federated Learning. *IEEE Trans. Inf. Forensics Secur.* **2023**, *18*, 1625–1637. [CrossRef]
19. Birchman, B.; Thamilarasu, G. Securing federated learning: Enhancing defense mechanisms against poisoning attacks. In Proceedings of the IEEE 33rd International Conference on Communications, Kailua-Kona, HI, USA, 29–31 July 2024.
20. You, Y.; Yoon, J.; Lee, H. Breakwater: Securing federated learning from malicious model poisoning via self-debiasing. In Proceedings of the IEEE International Conference on Communications, Denver, CO, USA, 9–13 June 2024.
21. Yan, H.; Zheng, C.; Chen, Q.; Li, X.; Wang, B.; Li, H.; Lin, X. A Proactive Defense Against Model Poisoning Attacks in Federated Learning. *IEEE Trans. Dependable Secur. Comput.* **2025**, *Early Access*.
22. Purohit, K.; Das, S.; Bhattacharya, S.; Rana, S. A data-driven defense against edge-case model poisoning attacks on federated learning. *arXiv* **2024**, arXiv:2305.02022v2.
23. Yue, G.; Han, X. FedDefense: A Defense Mechanism for Dishonest Client Attacks in Federated Learning. *Neural Process. Lett.* **2025**, *57*, 28. [CrossRef]
24. Yin, D.; Chen, Y.; Ramchandran, K.; Bartlett, P. Byzantine-robust distributed learning: Towards optimal statistical rates. In Proceedings of the 35th International Conference on Machine Learning, Stockholm Sweden, 10–15 July 2018; pp. 5650–5659.
25. Fedesoriano. Stroke Prediction Dataset. 2021. Available online: https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset (accessed on 9 April 2025).
26. Manchanda, C. Fraudulent Transactions Data. 2022. Available online: https://www.kaggle.com/datasets/chitwanmanchanda/fraudulent-transactions-data (accessed on 9 April 2025).

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.