Journal of the Brazilian Computer Society, 2026, 32:1, doi: 10.5753/jbcs.2026.5558

A Triad of Defenses to Mitigate Poisoning Attacks in Federated Learning

Blenda Oliveira Mazetto [**State University of Maringá** | ***pg56350@uem.br***]
Bruno Bogaz Zarpelão [**State University of Londrina** | ***brunozarpelao@uel.br***]

Department of Informatics, State University of Maringá, Av. Colombo, 5790, Jd. Universitário, Maringá, PR, 87020-900, Brazil.

Received: 15 February 2025 • **Accepted:** 30 June 2025 • **Published:** 16 March 2026

Abstract. Federated learning (FL) enables the training of machine learning models on decentralized data, potentially improving data privacy. However, the FL distributed architecture is vulnerable to poisoning attacks. In this paper, we propose an FL method capable of mitigating these attacks through a triad of defense strategies: organizing clients into groups, evaluating the local performance of global models during training, and using a voting scheme during the inference phase. The proposed approach first divides the clients into randomly sampled groups, each generating a distinct global model. Each client trains a local model on their private data and submits it to the central server. The central server aggregates the local models within each group to generate the global models. Then, each client receives all global models, selects the best performing one as their new local model, and the process repeats until training is complete. During the inference phase, each client classifies its inputs according to a majority-based voting scheme among the global models. Our experiments using the HAR and MNIST datasets demonstrate that our method can effectively mitigate poisoning attacks without compromising the global model's performance.

Keywords: Federated Learning, Poisoning Attacks, Machine Learning

1 Introduction

Traditional Machine Learning approaches require centralizing data on a single machine or datacenter. This data relating to users and organizations may contain private information that should not be shared, raising privacy concerns [Liu *et al*., 2021]. Federated Learning (FL) allows participants to collaboratively train a shared model while ensuring that all data remains stored locally on their devices, decoupling the ability to do Machine Learning from the need to store all data in a centralized server [Yang *et al*., 2019; McMahan and Ramage, 2017]. Compared to centralized learning, FL significantly reduces server computation costs by outsourcing and parallelizing the training process. FL also is a promising paradigm to empower on-device intelligence and mitigate the privacy and scalability issues in IoT systems [Witt *et al*., 2023].

In each iteration of a FL training scheme, the central server sends the current global model to all clients. Then, these clients train the global model using their private datasets and upload the trained local model to the central server. After receiving the local models of all clients, the central server calculates the new global model by aggregating the local models. The above steps will be repeated until the algorithm converges. Sequentially, the clients use the learned global model to make predictions for new inputs during the inference phase [Zhang *et al*., 2021].

Despite its benefits, the Federated Learning (FL) paradigm is vulnerable to Byzantine attacks, which represent a broad class of adversarial behaviors in which malicious clients (also known as Byzantine clients) act arbitrarily or deceitfully with the intent to disrupt the collaborative training process [Wang *et al*., 2022; Fang *et al*., 2020]. In the FL setting, such behavior may include sending incorrect, inconsistent, or deliberately manipulated models to the central server, thereby compromising the integrity of the global model. Poisoning attacks represent a particularly impactful and well-studied subclass of Byzantine attacks. These can be categorized into two main types: in data poisoning attacks, adversaries inject malicious data into their local training datasets, while in model poisoning attacks, they directly tamper with the gradients or model updates sent to the server [Tolpegin *et al*., 2020; Bouacida and Mohapatra, 2021]. The result is a corrupted global model with significantly reduced accuracy, undermining the system's performance on unseen inference data.

Robustness against Byzantine attacks, along with security and privacy preservation, has been a central focus in FL research. Exploring the field of Byzantine robust aggregation, Xu et al. [2022] and Li et al. [2023] propose aggregation techniques to identify suspicious local models and enhance robustness. Another widely used method against poisoning attacks is model analysis. Che et al. [2022] include a scoring system to differentiate clients, an election strategy to select representatives, and a selection strategy for committee formation, fostering a collaborative and secure training environment. Also using a model analysis method, Jebreel et al. [2024] propose a fragmentation technique and, in addition, global and local reputation vectors to select trustworthy clients. Zhang et al. [2023], Cao et al. [2021], and Cao et al. [2022] organize clients into subgroups to ensure a robust scenario against the influence of malicious clients. Finally, Andreina et al. [2020] use a method based on performance evaluation as a defense strategy, exploring a unique characteristic of FL, the multiple private datasets.

Our proposed approach combines three techniques to mit-

igate poisoning attacks in FL: dividing clients into groups, evaluating global model performance on local datasets, and making inferences based on a voting scheme. Initially, the central server randomly divides the clients into groups. After the clients complete local training, they send their local models to the central server, which generates a global model for each group. The global models are subsequently distributed to all clients, ensuring that every participant in the Federated Learning process receives all the global models generated by the groups. Once the clients receive the global models, they evaluate them using their own data and select the model with the best predictive performance. This selected global model becomes the client's new local model. These steps are repeated until the training is completed. After the training phase, the inference phase relies on a voting method. Given an input, a client uses the global models to make predictions, and the most frequent prediction is chosen as the final outcome for that input.

The combination of these three techniques leverages their strengths to address issues that arise when they are applied individually. While each technique alone can reduce the influence of corrupted local models to some extent, their effectiveness decreases quickly as the number of malicious clients increases. By integrating these three approaches, we develop a model that is more resilient to a growing number of malicious clients and can maintain training and test data within the clients at all times.

The remainder of this paper is organized as follows. In Section 2, we overview closely related work. In Section 3, we present our proposed approach to mitigate attacks on FL. We report on our experimental evaluation and results in Section 4. Finally, we conclude the paper in Section 5.

2 Related Work

In the FL field, robustness against Byzantine attacks and preservation of security and privacy have been key focus areas. Various methods and frameworks have been proposed to mitigate these threats and ensure the integrity of the globally trained models. According to Xia et al. [2023], defense strategies against poisoning attacks can be divided into three categories: 1) Model analysis, 2) Byzantine robust aggregation, and 3) Verification-based methods. Model analysis methods operate under the assumption that significant differences exist between poisoned and benign models, and that these differences can be distinguished. In response, the Byzantine robust aggregation strategy serves as a passive defense mechanism, mitigating the impact of poisoning attacks by altering the global model's aggregation method. Complementing this, the Verification-based defense strategy further strengthens security by introducing a verification step, which prevents attackers from forging data or models and complicates the execution of attacks.

Within the category of Byzantine robust aggregation defense, Xu et al. [2022] propose a filtering strategy based on Truth Discovery aggregation, an unsupervised iterative technique that identifies and eliminates unreliable local updates. Their approach, TDFL, demonstrates strong robustness even in Byzantine-majority scenarios (>50% malicious clients), effectively mitigating attacks without relying on a validation set or reference model. However, the method does not address incentive mechanisms for honest participation, which is noted as a limitation to be explored in future work. Complementing this category, Li et al. [2023] propose AutoGM, a secure aggregation rule, variant of the Geometric Median that adaptively excludes outliers and reweights updates based on skewness thresholds. AutoGM can be applied in both traditional FL paradigms and Personalized FL paradigms, demonstrating strong robustness against model and data poisoning attacks. However, the approach depends on user-defined hyperparameters, which may require careful tuning to maintain performance across varying conditions.

Expanding on this category, foundational methods such as Krum, Median, and Trimmed Mean have played pivotal roles in countering Byzantine threats. Blanchard et al. [2017] introduced Krum, an aggregation rule that selects a local model closest to the majority of other models, effectively filtering out adversarial outliers. Yin et al. [2018] proposed both Trimmed Mean and Median, each offering distinct strategies to enhance robustness. Trimmed Mean aggregates model parameters independently, discarding a fixed proportion of the highest and lowest values in each dimension, balancing computational efficiency and resilience to adversaries. Median computes the element-wise median of all updates, reducing the influence of extreme values and providing strong defense against malicious attacks. However, more recent studies have shown that tailored attacks can be designed to exploit the vulnerabilities of these aggregation rules, reducing their effectiveness in adversarial settings and highlighting the need for more adaptive or context-aware defenses.

Moving towards a decentralized approach, Che et al. [2022] explore the model analysis defense strategy, presenting CMFL, a serverless FL framework that employs a committee mechanism. In this framework, some clients are elected as committee members responsible for monitoring the training process and ensuring reliable aggregation of local gradients. CMFL introduces a scoring system to evaluate clients, an election strategy to select representatives, and multiple committee selection strategies tailored to different scenarios. The framework achieves faster convergence and improved model performance compared to traditional and Byzantine-tolerant FL models. However, its robustness primarily relies on detecting abnormal gradients, and it remains vulnerable to targeted attacks such as backdoor poisoning, which highlighs the need for more advanced election and selection mechanisms with formal guarantees under such conditions.

Continuing with the model analysis strategy, aimed at enhancing security and privacy, Jebreel et al. [2024] propose a novel lightweight protocol that enables participants to privately exchange and mix random fragments of their model updates before submission to the server. This design preserves the coordinate positions of parameters, allowing accurate aggregation while preventing the server from reconstructing original updates or linking them to specific users. The approach is reinforced by a reputation-based mechanism, where both global and local reputations guide participant selection and update weighting, improving resilience against adversarial behavior. However, the framework has not yet been evaluated under backdoor attacks or non-iid data scenarios,

which can be critical challenges in practical federated learning deployments.

Another widely used defense strategy in recent years involves the possible groupings of clients. In this context, Zhang et al. [2023] organize clients into subgroups with a hierarchical k-ary tree structure, using random partitioning and partial parameter disclosure to limit attacker influence. They propose SAFELearning, a secure aggregation protocol that detects backdoor and model-poisoning attacks even over encrypted updates, while preserving model privacy. The protocol also improves scalability in computation and communication.

Cao et al. [2021] propose an ensemble Federated Learning approach that uses majority voting among multiple global models trained on subsets of clients, also employing the defense strategy based on dividing clients into groups. This method ensures robustness when the majority of clients are honest, even in the presence of a limited number of malicious clients. Their approach achieves certified accuracy of 88% on MNIST when 20 out of 1,000 clients are malicious. Similarly, Cao et al. [2022] extend the method by grouping clients into probabilistic or deterministic subgroups, with each global model trained on a subgroup. Their final aggregation combines predictions from all models, enhancing robustness against malicious influence. The proposed FLCert framework provides provable security against poisoning attacks by using ensemble models from client groups, ensuring that the majority vote remains unaffected by malicious clients. However, a limitation of this work is that it does not incorporate prior knowledge about the learning task or the base FL algorithm when deriving certified security levels, which may affect its generalizability to diverse scenarios.

Finally, Andreina et al. [2020] propose a defense strategy that leverages a unique characteristic of FL (clients' access to private datasets) to detect backdoor attacks through performance evaluation.The authors propose BaFFLe, utilizing validation clients to detect if the global model update has been compromised by poisoning attacks, and discarding such updates when necessary. The results obtained from BaFFLe can achieve a detection accuracy of 100% with a false-positive rate below 5%, on both CIFAR-10 and FEMNIST datasets, even with small validation sets or when activated late in training. The defense remains compatible with secure aggregation protocols and requires minimal changes to existing FL setups. However, the method's effectiveness relies on the assumption that validation clients behave honestly and consistently hold representative data, which may not always apply in practice.

The proposed approach combines three defense techniques against poisoning attacks. Similarly to what Cao et al. [2021] and Cao et al. [2022] propose, the first step of our approach is a probabilistic grouping division. The next technique we used for attack mitigation is model performance evaluation. Andreina et al. [2020] propose a strategy where clients' private datasets are used to verify if an attack has compromised the global model. In our approach, each client receives the global models and uses their private dataset to evaluate these models. After the evaluation, each client selects the global model with the best predictive performance to be their new local model. Finally, this work uses a voting strategy for inference, similar to the ensemble Federated Learning used by Cao et al. [2021] and Cao et al. [2022]. During the final step of our approach, each client receives the global models and uses majority voting among them to predict the labels. The objective of combining these three strategies is to enhance the model's resilience against an increasing number of malicious clients, addressing the challenge that other methods face when dealing with large proportions of compromised clients. Table 1 compares the reviewed studies and the proposed approach.

3 Proposed Approach

Unlike traditional FL models, our proposed method begins with the random grouping of n clients into N groups of k clients each. The purpose of sampling the clients into groups is that, as long as we do not have a vast majority of malicious clients, we still have a high chance of retaining uncompromised groups. When most clients are benign, the influence of malicious clients is reduced, as a malicious client can only affect the groups to which it belongs. It is also important to highlight that the random division of groups is done in a way that a client can belong to more than one group. Each client is assigned to each group independently with probability $p = \frac{k}{n}$, the probability of a client belonging to more than one group is given in Equation 1. Accordingly, when a client is assigned to multiple groups, its local model update is aggregated into the global model of each group to which it belongs. Figure 1 shows the division of 5 clients (n = 5) into 3 groups (N = 3), with each group containing 2 clients (k = 2). Randomly, clients 1 and 4 were assigned to group 1, clients 3 and 5 to group 2, and finally, clients 2 and 4 to group 3.

$$P(X \geq 2) = 1 - (1-p)^N - N\,p\,(1-p)^{N-1} \quad (1)$$

Once the groups are defined, the training is initiated. The central server sends a learning model to all clients, with this initial model having automatic weights that follow a uniform distribution, which will be updated over the training process. After receiving the model, the clients update it using their local data, thus generating n local models. Then, the clients send their local models to the central server.

The central server uses the groups defined earlier to aggregate the local models. Each one of the N groups generates a global model GM_i, which is the result from the aggregation of the local models of the clients belonging to that group. Thus, we will have $GM_1, GM_2, GM_3, ..., GM_N$. This process can be observed in Figure 2 and detailed in Algorithm 1. Aggregation is an important step in FL systems. Our approach makes it possible to choose any aggregation method, as this does not affect the functioning of our method. After the global models are computed, the central server sends them to each client, a step that can be seen in Figure 3.

Table 1. Comparison among the reviewed approaches based on their reliance on client grouping strategies, whether the central server allows different types of aggregation, and the use of client feedback for evaluating model performance. It also lists the types of attacks simulated in each study.

Related work	Uses grouping	Allows different aggregations	Uses client feedback	Simulated Attacks
Xu *et al*. [2022]	×	×	×	Label-Flipping, Arbitrary Model, Krum, Trim and Backdoor
Li *et al*. [2023]	×	×	×	Label-Flipping and Gaussian
Che *et al*. [2022]	×	✓	✓	Malicious Gradients
Jebreel *et al*. [2024]	✓	×	✓	Label-Flipping and Gaussian
Andreina *et al*. [2020]	×	×	✓	Label-Flipping
Zhang *et al*. [2023]	✓	×	×	Label-Flipping and Adaptive Semantic
Cao *et al*. [2021]	✓	✓	×	Malicious Gradients
Cao *et al*. [2022]	✓	✓	×	Label-Flipping, Same-Value, Krum and Trim
Our Approach	✓	✓	✓	Label-Flipping, Same-Value, Gaussian-Noise and Gradient-Scaling

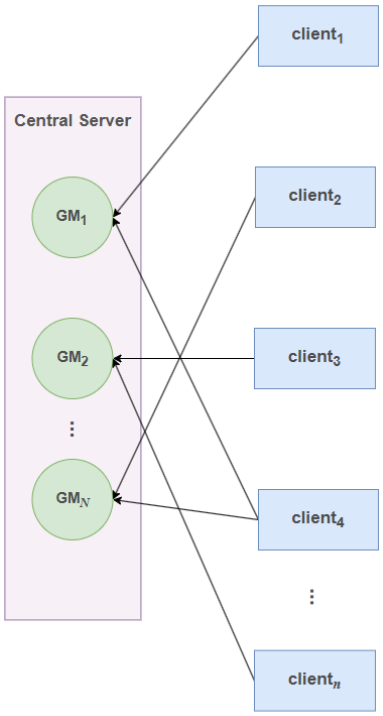

Figure 1. Example of the group division process with $n = 5$, $N = 3$ and $k = 2$.

Algorithm 1 Group-based aggregation of local models by the central server

Input:
- *local_models*: List of models, where the i-th entry corresponds to client i
- *groups*: List of groups, each containing indices of participating clients

Output:
- *global_models*: List of global models, one per group

Server executes:

global_models ← [ ]
for each *group* in *groups* **do**
  group_models ← [ ]
  for each *client_index* in *group* **do**
    model ← *local_models[client_index]*
    append *model* to *group_models*
  end for
  Compute *aggregated_model* from *group_models* using chosen aggregation method (e.g., FedAvg)
  Append *aggregated_model* to *global_models*
end for
return *global_models*

Sequentially, the proposed approach carries out a performance evaluation step, which aims to improve the whole system performance using clients' private datasets $D_1, D_2, ..., D_n$. Once the clients receive all the global models, they use their private validation datasets D_i to evaluate the global models $GM_1, GM_2, ..., GM_N$. Then, each client computes the F1-score for each global model based on their own data, using the counts of true positives (TP), false positives (FP) and false negatives (FN) extracted from the confusion

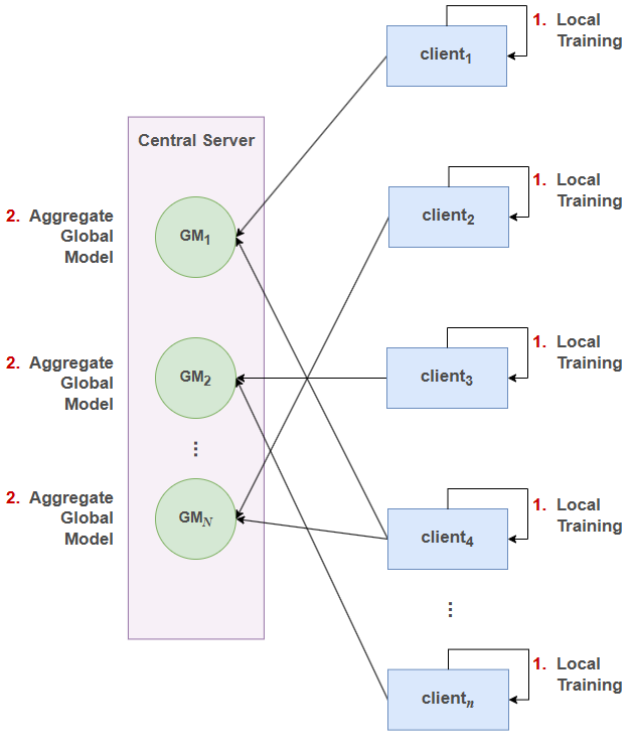

Figure 2. First and second steps of our proposal, where we can visualize the clients training a local model with their private dataset and the central server aggregating the local models into the global models according to the group division.

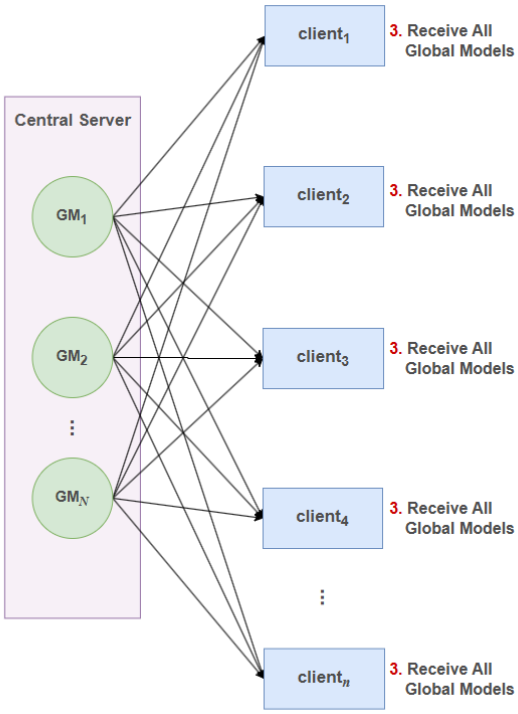

Figure 3. Third step of our approach, where each client receives all previously calculated global models.

matrix, as defined in Equation 2. The F1-score is a global metric for evaluating predictive performance, particularly useful in scenarios with class imbalance. Next, each client selects the global model that achieved the highest F1-score in the evaluation process and this global model becomes the new local model for that client, as shown in Figure 4 and outlined in Algorithm 2.

Global models produced by compromised groups are expected to perform worse in terms of F1-score, since they were affected by poisoned models. This expectation is based on the nature of the poisoning attacks evaluated in our experiments, which are specifically designed to degrade model performance. As a result, clients tend to discard these compromised models during evaluation. By systematically avoiding underperforming global models during the performance evaluation process, the influence of poisoned models is gradually reduced over successive training rounds, as they are less likely to be chosen and propagated. This dynamic acts as a filtering mechanism that helps suppress the long-term impact of adversarial behavior, contributing to the disappearance of poisoning effects on the global model. Furthermore, this solution allows us to use the clients' private datasets to guide model selection, without compromising data privacy.

$$F1 = \frac{2\,TP}{2\,TP + FP + FN} \tag{2}$$

Algorithm 2 Performance evaluation and selection of the best global model by the client

Input:
- *global_models*: List of global models, one per group
- *validation_data*: Client's private validation dataset

Output:
- *selected_model*: Global model with the highest F1-score

Client executes:

best_f1 ← $-\infty$
selected_model ← **None**
for each *model* in *global_models* **do**
  f1 ← EvaluateF1(*model*, *validation_data*)
  if *f1* > *best_f1* **then**
    best_f1 ← *f1*
    selected_model ← *model*
  end if
end for
return *selected_model*

The steps described above are repeated until the end of the training. When the training is completed, we move to the inference phase, which relies on a voting method. Similarly to the previous steps, the clients' local models are aggregated according to the initially defined groups. Shortly after, the global models are sent to all clients. Once the clients have received all the global models, they start the voting step, where each client makes inferences with their own data. During the inference phase, the N global models are used to predict labels for inputs. Specifically, given a test input x, the client uses each global model to predict its label. After that, the client calculates the frequency of all predicted labels, which

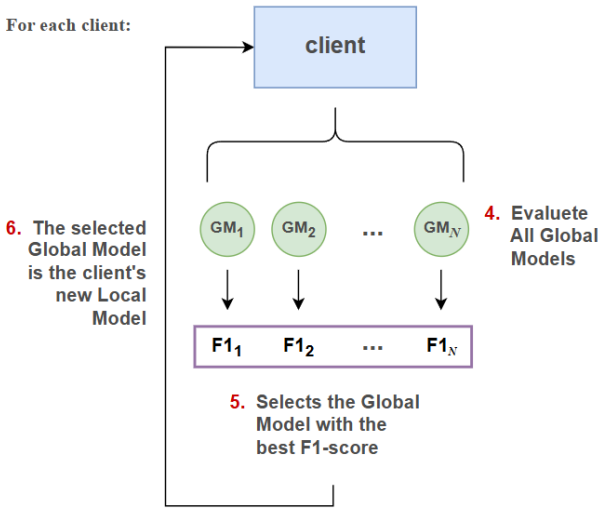

Figure 4. Fourth, fifth and sixth steps of our approach, where each client evaluates the received global models and selects the best of them to become their new local model.

is the number of global models that predict a certain label for x. Thus, the client takes a majority vote among the N global models to predict the label for the input x. The label with the highest number of predictions is the resulting label. A detailed step-by-step description of this process is provided in Algorithm 3, complementing the illustration in Figure 5. The aim of this step is to ensure that the resulting label from the majority vote among the N global models remains unaffected by a limited number of malicious clients. When there are ties, i.e., multiple labels have the same highest frequency, the client randomly selects one of the tied labels.

Algorithm 3 Inference using majority voting over global models

Input:
- *global_models*: List of global models, one per group
- x: Test input

Output:
- *final_label*: Predicted label for x via majority voting

Client executes:

Initialize *label_votes* as an empty map (label → count)
for each *model* in *global_models* **do**
  label ← *model*.predict(x)
  Increment count for *label* in *label_votes*
end for
Identify label(s) with the maximum vote count
if there is a tie among top labels **then**
  final_label ← randomly select one of the tied labels
else
  final_label ← label with the highest vote
end if
return *final_label*

3.1 Dealing with Malicious Selections

During the performance evaluation phase of our method, each client assesses the global models $GM_1, GM_2, \ldots, GM_N$ us-

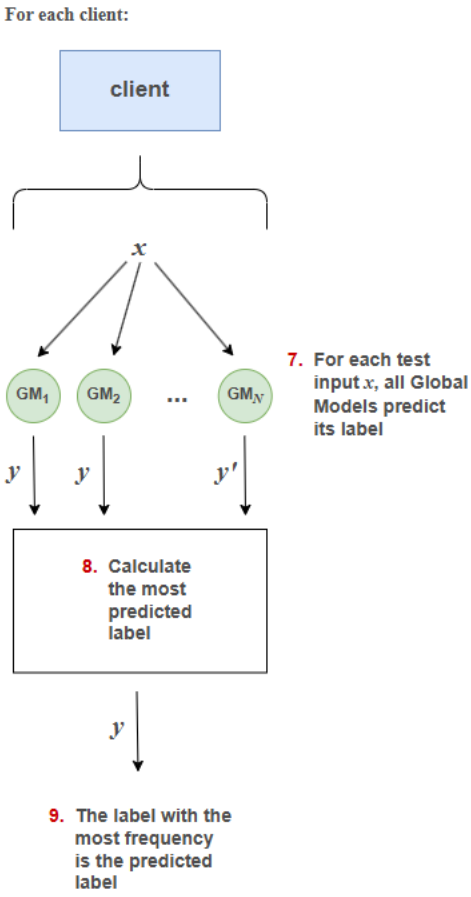

Figure 5. Seventh, eighth and ninth steps of our approach, where each client makes inferences for their test data taking into account the majority vote among global models.

ing its private validation dataset and selects the one with the highest F1-score as its new local model. A malicious client, however, may attempt to compromise the system by selecting the worst-performing global model, i.e., the one with the lowest F1-score, instead of the best, to degrade overall system performance.

Despite this threat, the system demonstrates resilience to such adversarial behavior as long as the number of malicious clients m remains less than half of the total number of clients n, i.e., $m < \frac{n}{2}$.

This resilience stems from the majority voting mechanism used during inference. Each client uses all N global models to predict the label of a test input x, and the final output label is determined by majority vote over the predictions from these models. Even if some global models are influenced by malicious clients during training, they are likely to be outvoted if most global models were selected and trained by benign clients who based their choices on accurate model evaluations.

This approach aligns with principles from Byzantine Fault Tolerance (BFT) theory, where systems that rely on majority consensus can tolerate up to $m < \frac{n}{2}$ malicious participants without compromising correctness [Marcozzi and Mostarda, 2024]. While classical BFT requires $m < \frac{2}{3}$ for full consensus (as in blockchain or secure databases), systems based on simple majority voting can remain correct as long as fewer than 50% of participants are malicious. Empirical results supporting this claim are presented in Section 4.2.4. The experiments show that model performance remains stable until approximately 50% of the clients are malicious. When this threshold is exceeded, a significant drop in accuracy and F1-score is observed, confirming the expected breakdown point.

4 Evaluation and Results

This section presents the evaluation of our proposed approach through a series of experiments. The experiments were designed to reflect practical FL scenarios, incorporating various attack strategies to thoroughly evaluate the effectiveness and resilience of the proposed method. We provide details on the experimental setup, including datasets, FL configuration, and the adversarial strategies applied. Following that, we present and analyze the results across different perspectives, comparing our method with a baseline and existing defense techniques.

4.1 Experimental Setup

Datasets: We utilize the MNIST and Human Activity Recognition Using Smartphones (HAR) datasets. Next, more details about them are presented:

- MNIST: The MNIST dataset [Deng, 2012] is a widely used dataset in machine learning, comprising 70,000 grayscale images of handwritten digits (0-9), each sized 28×28 pixels. Given its popularity for training machine learning models, we employed it to simulate FL scenarios. Our experiments were conducted with 30 and

50 clients. Initially, the dataset was split into training, validation, and test sets, with 50,000 samples for training, 10,000 for validation, and 10,000 for testing. In our federated environment, these subsets were evenly distributed among the clients, simulating each client having its own private dataset.

- HAR: The Human Activity Recognition Using Smartphones (HAR) dataset [Reyes-Ortiz *et al*., 2012] was created from recordings of daily activities performed by individuals carrying a smartphone on their waist, which was equipped with inertial sensors. The experiments involved 30 volunteers aged 19 to 48, each performing six activities corresponding to the six labels in the dataset (WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, SITTING, STANDING, LAYING). The dataset includes 561 features and 10,299 instances. The HAR dataset is naturally federated for 30 clients, since the data for each volunteer can be easily converted to the private dataset for a client. For this reason, HAR clients do not have the same number of samples (since the volunteers did not produce the same amount of samples). Therefore, the first step was to partition the dataset into private datasets for the clients, followed by splitting these private datasets into training, validation, and test sets based on percentages: 70% for training, 10% for validation, and 20% for testing.

FL setup: For the MNIST dataset, experiments were conducted with two variations in the number of clients: $n = 30$ and $n = 50$. For the 30-client scenario, one variation of N were tested: $N = 15$. In the 50-client variation, three variations of N were also tested: $N = 15$, $N = 25$, and $N = 35$. For the HAR dataset, experiments were conducted with 30 clients, as the dataset is naturally federated for 30 clients. Three variations of N were tested: $N = 9$, $N = 15$, and $N = 21$. Regarding the number of clients per group, values closely aligned with those reported in the literature were selected. Thus, the scenarios mentioned above were tested with 3 and 5 clients per group ($k = 3$ and $k = 5$). The aggregation method chosen was FedAvg, which calculates the average of local models. This method was selected for its performance, efficiency, and scalability potential. Additionally, compared to other aggregation methods like Krum, Trimmed Mean, and Median, FedAvg has reduced operational costs.

Model Architectures and Parameter Settings: For the MNIST dataset, we used a Convolutional Neural Network (CNN) architecture proposed by Cao *et al*. [2021]. Key parameters included a batch size of 32, a learning rate of 0.001, and Stochastic Gradient Descent as the optimizer. The number of epochs was set to 100, with 10 global iterations. For the HAR dataset, we employed a Deep Neural Network (DNN) with two fully connected hidden layers, each containing 256 neurons and using ReLU activation functions, an architecture proposed by Cao *et al*. [2021]. The parameters for this model included a batch size of 64, a learning rate of 0.001, and Stochastic Gradient Descent as the optimizer. The number of epochs and global iterations were 200 and 20, respectively.

Attacks: We chose four poisoning attacks with distinct strategies, one targeting the data (Label-Flipping Attack), and the others targeting the model (Same-Value Attack, Gaussian-Noise Attack and Gradient-Scaling Attack). Although the Same-Value Attack is a highly impactful method due to its ability to completely nullify the functionality of the model, it is also relatively easy to detect due to its extreme and uniform nature. To ensure a more complete evaluation, we included additional attacks, such as the Gaussian-Noise Attack and the Gradient-Scaling Attack. These methods are subtler, making detection more challenging and better reflecting real-world scenarios where adversaries often employ sophisticated strategies to evade detection.

- **Label-Flipping Attack**: The training data is targeted by altering the labels of specific samples, as reported in Xu *et al*. [2022]; Li *et al*. [2023]; Jebreel *et al*. [2024]; Andreina *et al*. [2020]; Zhang *et al*. [2023]; Cao *et al*. [2022]. The objective is to induce a local model trained with incorrectly labeled data. For the MNIST dataset, malicious clients relabeled all their labels as "0", while for the HAR dataset, labels were changed to "WALKING".
- **Same-Value Attack**: The learning model is compromised by setting all its parameters to a single value (as described in Cao *et al*. [2022]), zero in our case. This strategy nullifies the model's ability to learn and make accurate predictions. In our scenario, the attack is executed by a malicious client when sending its local model to the central server.
- **Gaussian-Noise Attack**: Gaussian noise is introduced into the model updates by adding random perturbations to the model parameters. This noise is generated by sampling from a standard Gaussian distribution, which has a mean (μ) of 0 and a standard deviation (σ) of 1. The generated noise is then added directly to the existing parameters of the model, based on the method described in Jebreel *et al*. [2024]. This disruption prevents the model from converging properly, thus hindering the learning process and making it harder for the model to achieve accurate predictions.
- **Gradient-Scaling Attack**: Malicious clients scale their local gradients to manipulate the learning model. Each gradient element is multiplied by a random value $\lambda \in [a, 1)$, where a is a constant determining the attack's intensity. In this experiment, a is set to 0.5, ensuring the gradients remain within a controlled range while amplifying the attack's impact, as detailed in Che *et al*. [2022].

4.2 Results

In this section, we present the results of experiments on the approach proposed in Section 3. In our method, inference is performed on each client's device, and each client generates their own performance metrics. However, for visualization and comparative purposes, we report an overall F1-score computed across all clients. It is important to clarify that we calculated the global F1-score by first summing the true positives (TP), false positives (FP), and false negatives (FN)

from all clients, and then applying the standard F1-score formula, as shown in Equation (2). This corresponds to the micro-averaged F1-score [Takahashi *et al*., 2022]. We chose the F1-score as it balances precision and recall in a single metric. Precision measures the proportion of correct positive predictions among all positive predictions made, while recall measures the proportion of correct positive predictions among all actual positive instances.

We then evaluate our approach against the following attacks: Label-Flipping, Same-Value, Gaussian-Noise, and Gradient-Scaling. Our method successfully mitigates label alterations in the Label-Flipping attack and reduces the impact of the Same-Value attack, maintaining model functionality. It also handles more subtle attacks like Gaussian-Noise and Gradient-Scaling, preventing major disruptions to the training process. These results demonstrate the robustness of our solution in ensuring model accuracy and reliability.

4.2.1 Our Approach vs. Single-global-model FedAvg

To assess the effectiveness of our defense strategies, we implemented an FL setup based on the FedAvg aggregation algorithm without any defense mechanism, referred to as the 'Single-global-model FedAvg'' or "defense-less" method. This setup enables us to evaluate whether our defense strategies improve the system's resilience to poisoning attacks under various attack types, different client configurations, and group setups. Figures 6, 7, 8, 9, and 10 present a comparison between the proposed approach and the Single-global-model FedAvg method. In all figures, it is clear that our strategy outperforms the defense-less approach.

For the MNIST dataset, Figure 6 shows a significant difference between the proposed approach and the defense-less method for the Label-Flipping and Same-Value attacks. While the defense-less method experiences a performance drop below 0.8 with more than 20% of malicious clients, the proposed approach maintains an F1-score above 0.8 even with 90% of malicious clients. In Figure 8, for the Gaussian-Noise attack, the defense-less method shows a performance decay to approximately 0.6 from the start, with only 10% of malicious clients, whereas the proposed approach keeps an F1-score of 0.9 even with up to 70% of malicious clients. For the Gradient-Scaling attack, the defense-less method manages to maintain an F1-score above 0.8 with up to 50% of malicious clients but is outperformed by the proposed approach, which sustains an F1-score above 0.9 even with 90% of malicious clients.

For the HAR dataset, Figure 7 demonstrates that for the Label-Flipping attack, our approach achieves the same metrics as the defense-less method but with approximately 20% to 33% more malicious clients. For the Same-Value attack, the defense-less method quickly deteriorates, maintaining an F1-score above 0.8 only for 10% of malicious clients. In contrast, our approach maintains an F1-score above 0.8 with up to 50% malicious clients, declining slowly thereafter. In Figure 9, for the Gaussian-Noise attack, we observe that our approach outperforms the defense-less method, maintaining the same metrics but with 20% to 55% more malicious clients. Similarly, for the Gradient-Scaling attack, our method maintains the same metrics but with 10% to 53% more malicious clients.

In this work, there are three important variables, n (number of clients), N (number of groups), and k (number of clients per group), whose values vary throughout the study. Next, we will analyze the impact of each variable.

In Figures 6 and 7, we can observe the impact of varying k. Figure 6, which corresponds to the MNIST dataset, shows the results for $k = 3$ and $k = 5$ (values chosen based on those commonly used in the literature) under the Label-Flipping and Same-Value attacks. In these results, the variation of k was barely noticeable. Similarly, Figure 7, corresponding to the HAR dataset, demonstrates a greater variation in the results, where a smaller value of k ($k = 3$) achieved superior performance. The improved results for a smaller k can be attributed to two factors: the lower number of clients per group reduces the likelihood of a malicious client compromising that group, and the HAR dataset is more heterogeneous than the MNIST dataset, which means that malicious clients can have a greater impact.

Regarding the impact of varying the number of groups (N), we can conclude that there is no significant difference in mitigation capacity across all attack types for both MNIST and HAR datasets. This trend is evident in Figures 6, 7, 8, and 9, where performance consistently starts to decline at the same point regardless of the value of N. However, for the HAR dataset under the Gaussian-Noise attack, it is possible to observe a difference in performance between $N = 9$ and $N = 15$ or $N = 21$, which appears to be an exception to this general trend. Overall, this observation indicates that we can opt for the smallest N, as it reduces the number of groups and the associated global models, thereby lowering the computational cost without compromising performance.

In Figures 6a, 8a, and 10, we utilized the same dataset (MNIST) and the same number of groups ($N = 15$) while varying the number of clients. Figures 6 and 8 correspond to $n = 50$, whereas Figure 10 represents $n = 30$. This allows us to observe the impact of changing the number of clients on the results. We can observe a slight difference in mitigation capacity. For 30 clients, depending on the attack, performance started to decline earlier.

Aside from the numerical results, we can highlight the architectural aspects that explain the superior performance of the proposed approach over the defense-less method. First, the use of multiple global models assigned to different groups reduces reliance on a single centralized aggregation, thereby mitigating the impact of compromised data or malicious client behavior. This decentralized structure dilutes the influence of poisoning attacks across several aggregators, making it more difficult for adversarial contributions to dominate the final model. Additionally, grouping clients leverages the natural variability of local data, which proves particularly beneficial in heterogeneous scenarios, such as the HAR dataset.

4.2.2 Separated Defenses

As previously mentioned, our approach combines three defense strategies. In this subsection, we assess the individual effectiveness of each strategy to better understand their contributions to the overall defense mechanism. By testing separate configurations, we aim to highlight that although each strategy

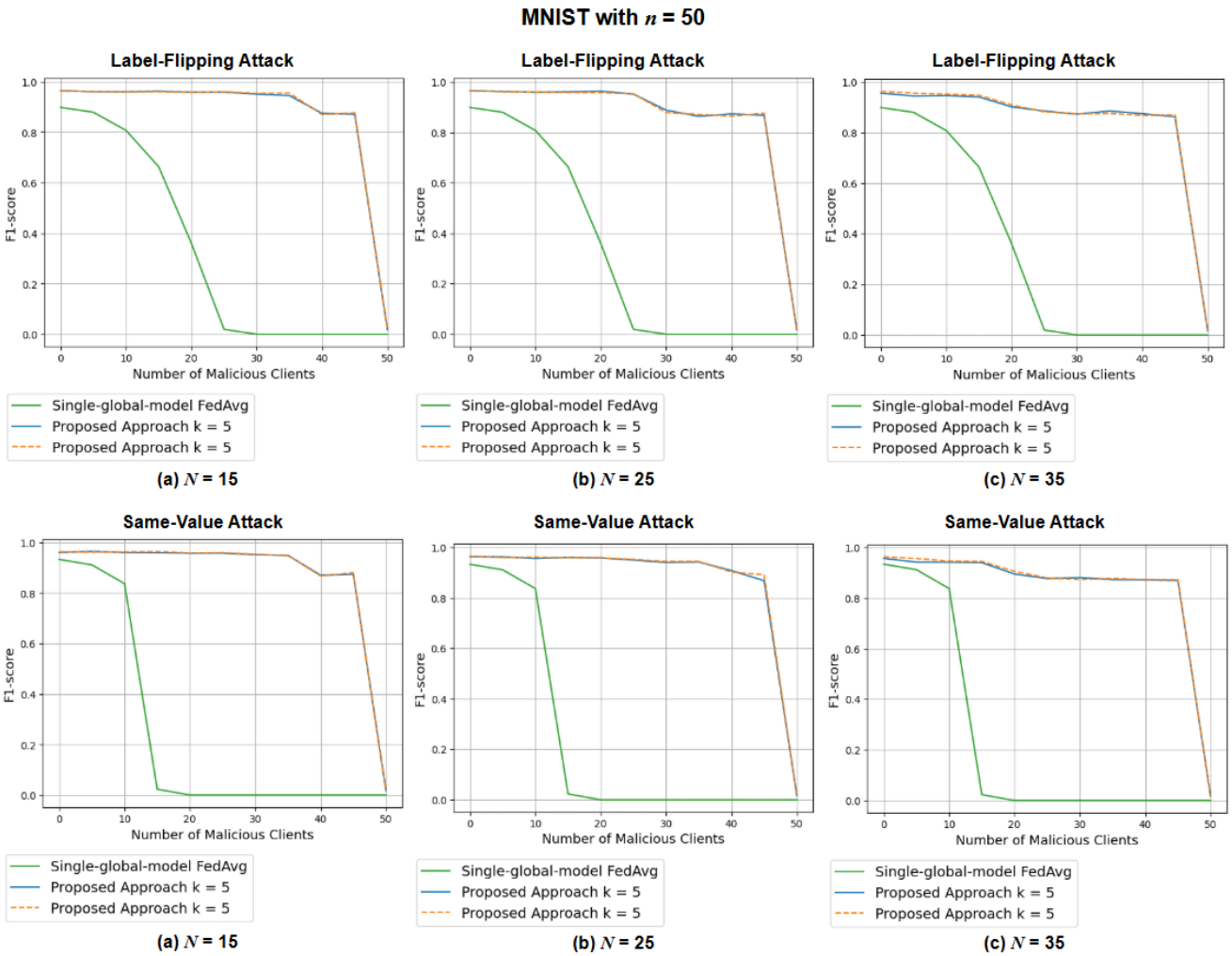

Figure 6. Comparison of the Single-global-model FedAvg approach with the proposed approach using the MNIST dataset, with $n = 50$, N varying between 15, 25, and 35, and k varying between 3 and 5, for Label-Flipping and Same-Value attacks.

can offer some level of resilience against poisoning attacks, they are significantly more effective when combined.

We examine two feasible combinations: one that pairs group division with the evaluation of global model performance, and another that combines group division with a voting-based inference method. These are the only viable configurations for individual assessment, as both the evaluation mechanism and the voting process depend on the existence of client groups to operate effectively. Without group division, their logic cannot be meaningfully applied. At the same time, group division alone does not provide robust defense against adversarial clients, as it lacks mechanisms to detect or mitigate malicious behavior.

In the first variation (defenses 1 and 2), the final voting step is excluded. The n clients are first divided into N groups, with each group consisting of k clients. Once the groups are defined, local training begins. Afterward, the clients send their local models to the central server, which aggregates them based on the predefined groups. Following this aggregation, clients receive all the aggregated global models and evaluate their performance using private validation datasets. Each client selects the global model with the highest F1-score as their new local model. This process is repeated iteratively until the training concludes.

The second variation (defenses 1 and 3) follows a similar initial process. The n clients are first divided into N groups, with each group consisting of k clients, and local training is initiated. After training, clients send their local models to the central server, which aggregates the models based on the predefined groups. Each client then receives the global model from the group they belong to (or randomly selected from the groups they participate in, if they are part of multiple groups). This iterative process continues until the training is complete. Once training finishes, the inference phase begins, relying on the previously described voting method.

In Figure 11, we present the results obtained using the MNIST dataset. Across all tested attacks, the proposed approach outperformed methods that rely on only two out of the three defenses. Notably, for three of these attacks (Label-Flipping, Same-Value, and Gradient-Scaling), the F1-score remained above 0.82 even with 90% of the clients being malicious. For the other tested attack (Gaussian-Noise), the F1-score exceeded 0.85 even with 70% of the clients being malicious.

In Figure 12, we show the results for the HAR dataset. Similar to MNIST, the proposed approach consistently outperformed techniques that utilize only two of the three defenses across all attacks. Particularly noteworthy is the Gradient-

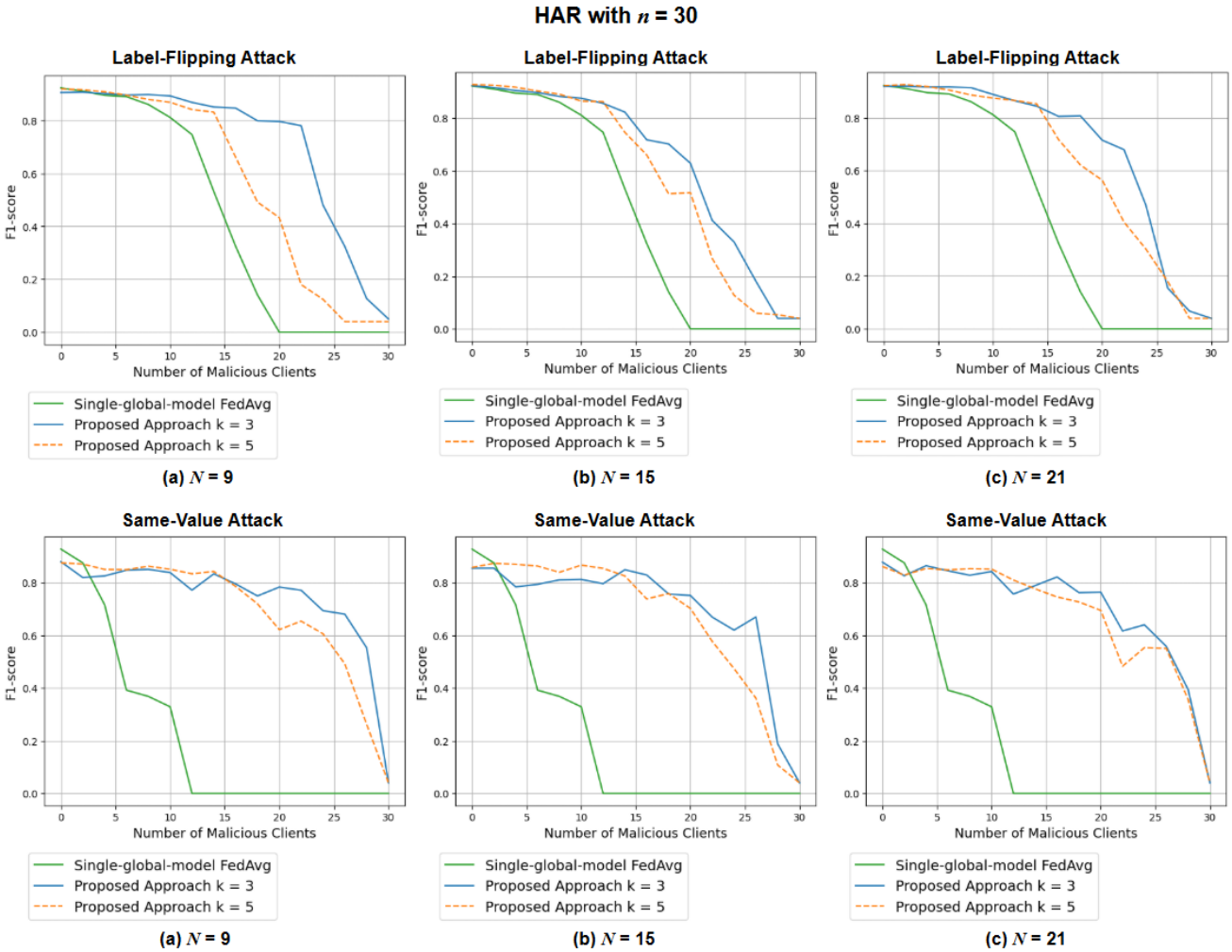

Figure 7. Comparison of the Single-global-model FedAvg approach with the proposed approach using the HAR dataset, with $n = 30$, N varying between 9, 15, and 21, and k varying between 3 and 5, for Label-Flipping and Same-Value attacks.

Scaling attack, where the F1-score remained above 0.8 even with 90% of the clients being malicious.

A deeper analysis shows that combining the three defense strategies is essential for providing stronger protection against poisoning attacks. Group division alone is insufficient, as it only separates client data without addressing adversarial behavior. The global model performance evaluation (defense 2) serves as a filter, enabling clients to choose the best-performing model and enhancing the final model's robustness. The voting mechanism (defense 3) further strengthens the defense by enabling collective decision-making, minimizing the impact of adversarial data. In conclusion, the integration of these strategies creates a robust and adaptable defense system, significantly improving resilience against various poisoning attacks.

4.2.3 Our Approach vs. Other Existing Defenses

In this subsection, we compare the proposed defense strategy with well-established defense methods, namely Krum, Trimmed Mean, and Median. These methods represent some of the most widely used techniques in federated learning to address adversarial threats. By analyzing the performance of these methods across different experimental conditions, we can gain insights into the relative strengths and limitations of each defense technique. Furthermore, this comparison helps to highlight the unique advantages of our approach, especially in scenarios where traditional defenses may fall short.

In Figure 13, we present the results obtained using the MNIST dataset. Across all tested attacks, the proposed approach outperformed other existing defenses. For the Label-Flipping and Same-Value attacks, there is a significant difference, where our approach maintains an F1-score above 0.8 even with 90% of the clients being malicious, while other approaches achieve the same F1-score for at most 20% of malicious clients. On the other hand, the Gaussian-Noise and Gradient-Scaling attacks are more subtle, leading to defense strategies producing more similar results, with our approach performing marginally better.

In Figure 14, we show the results for the HAR dataset. Similar to MNIST, the proposed approach performed better than other existing defenses. For the Label-Flipping, Gaussian-Noise, and Gradient-Scaling attacks, our approach achieved an F1-score above 0.8, even with approximately 20% more malicious clients compared to other defenses. For the Same-Value attack, our approach maintained an F1-score above 0.7 even with 90% of the clients being malicious, while other defense strategies achieved the same F1-score with only about 20% of malicious clients.

While methods like Krum, Trimmed Mean, and Median

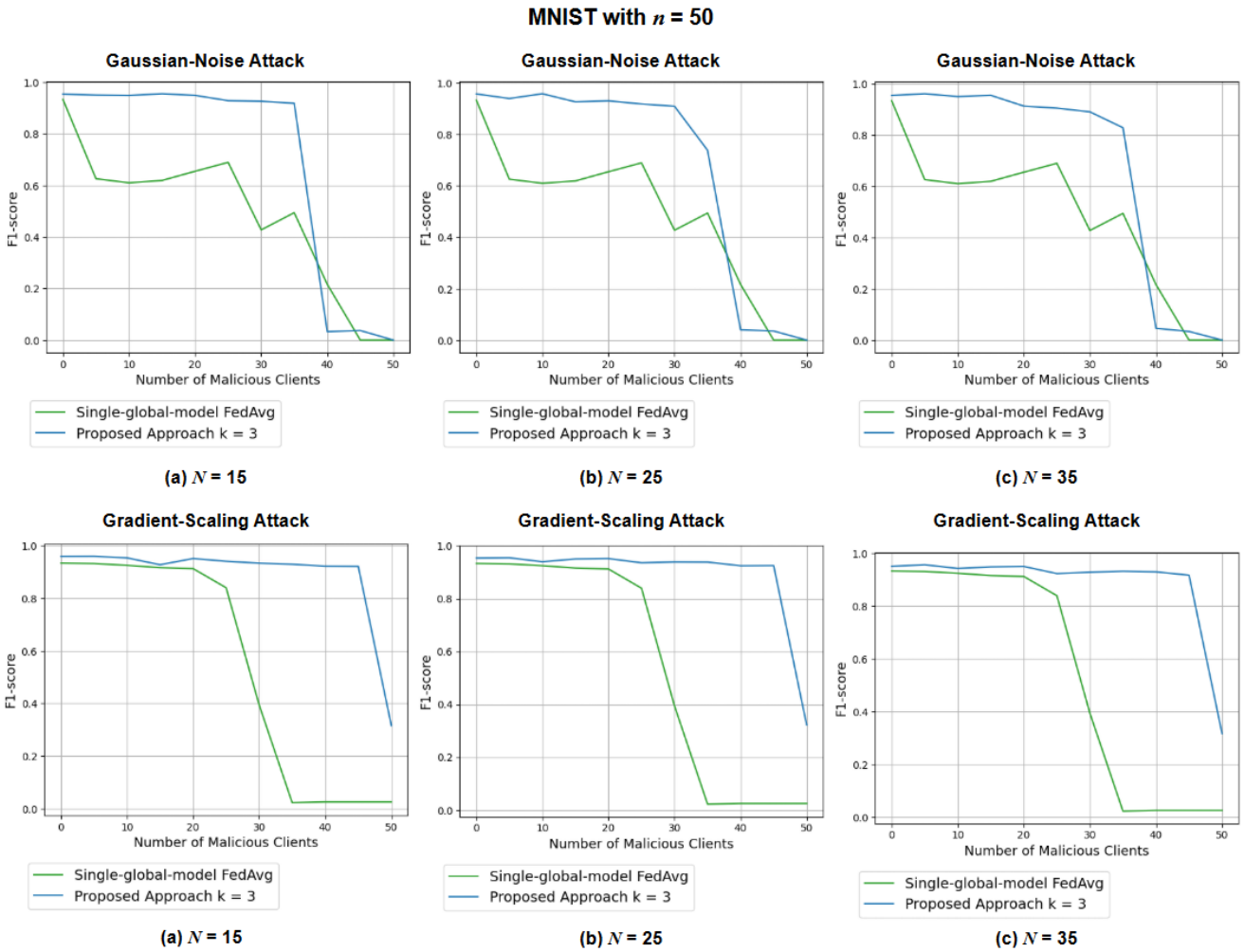

Figure 8. Comparison of the Single-global-model FedAvg approach with the proposed approach using the MNIST dataset, with $n = 50$, N varying between 15, 25, and 35, and $k = 3$ for Gaussian-Noise and Gradient-Scaling attacks.

rely primarily on model aggregation, our approach benefits from group division, global model evaluation, and voting, which help dilute the influence of malicious clients. This results in superior performance, especially for attacks like Label-Flipping and Same-Value, where our approach maintains high F1-scores even with a large percentage of malicious clients.

4.2.4 Dealing with Malicious Selections

As mentioned in Subsection 3.1, during the performance evaluation phase of our method, a malicious client may attempt to compromise the system by selecting the worst-performing global model, i.e., the one with the lowest F1-score, instead of the best, in an effort to degrade overall system performance. In Figure 15, the curve labeled "Malicious Selections" represents exactly this adversarial scenario: our proposed method is still being used, but the malicious clients intentionally misreport their evaluations to favor the worst model. Despite this, we observe that our approach remains superior to the Single-global-model FedAvg approach throughout the experiment. Furthermore, the results indicate that model performance remains stable even with malicious votes, up until approximately 50% of the clients are malicious.

5 Conclusion

In this article, we proposed an FL system combining three different techniques to mitigate poisoning attacks. Our approach divides the clients into randomly sampled groups, evaluates the global model's performance using the client's private datasets, and, during its final step, uses a majority voting scheme to predict the labels. We assessed the effectiveness of our proposal against four different attacks, three targeting the model and one targeting the data, to ensure a more comprehensive evaluation of the robustness against poisoning. Our solution proved robust in all scenarios, showcasing its ability to protect against threats. All results obtained demonstrated improvements over a basic FL approach without defenses. Moreover, we highlighted the strength of our method through the combination of these techniques, as isolating and testing them individually showed that the full potential is realized only when they are combined. This was further demonstrated by the superior performance of our approach compared to existing defense methods, such as Krum, Trimmed Mean, and Median, in all tested scenarios.

Despite its robustness against malicious clients, the proposed approach has limitations. The method introduces additional complexity that can be justified by the improved

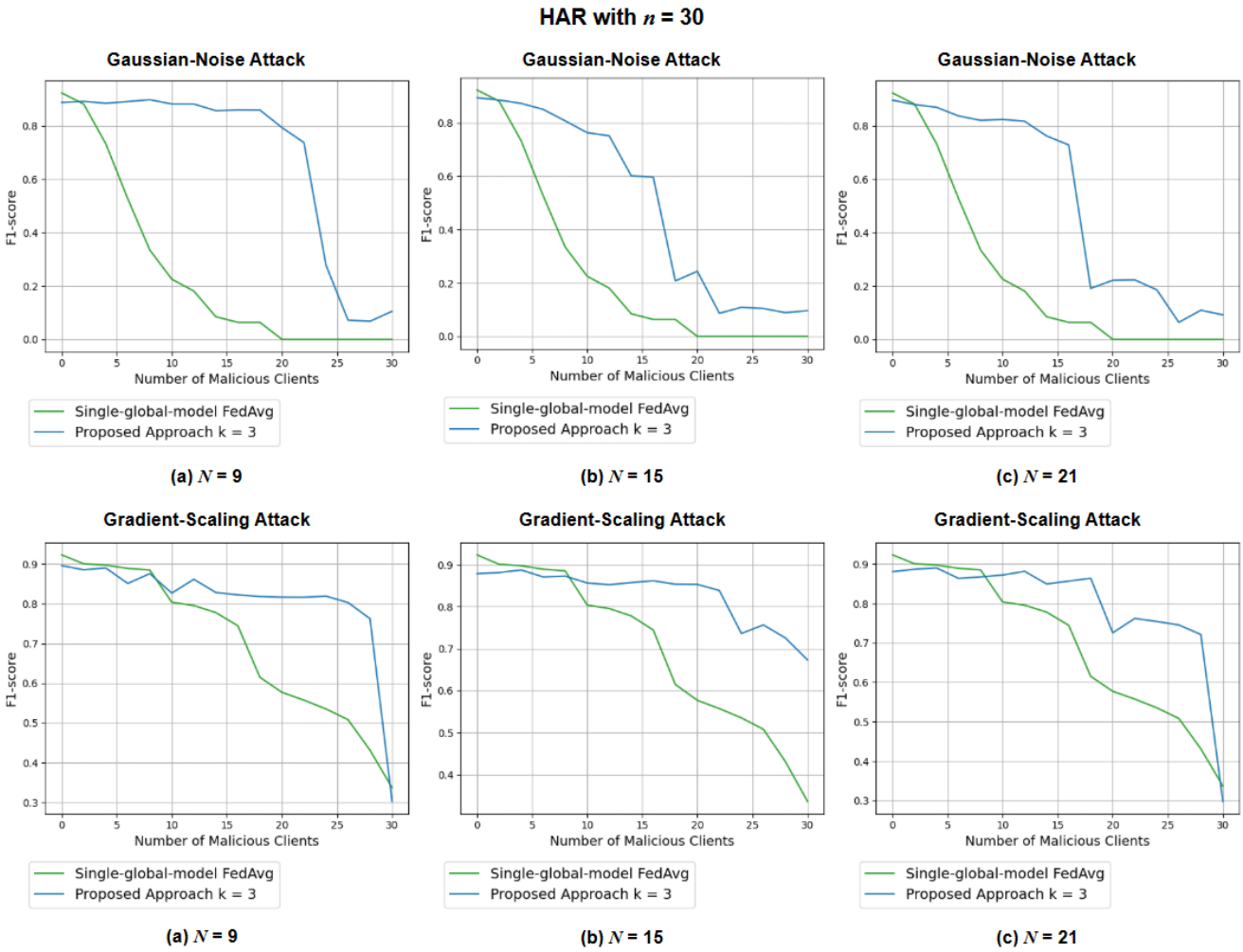

Figure 9. Comparison of the Single-global-model FedAvg approach with the proposed approach using the HAR dataset, with $n = 30$, N varying between 9, 15, and 21, and $k = 3$ for Gaussian-Noise and Gradient-Scaling attacks.

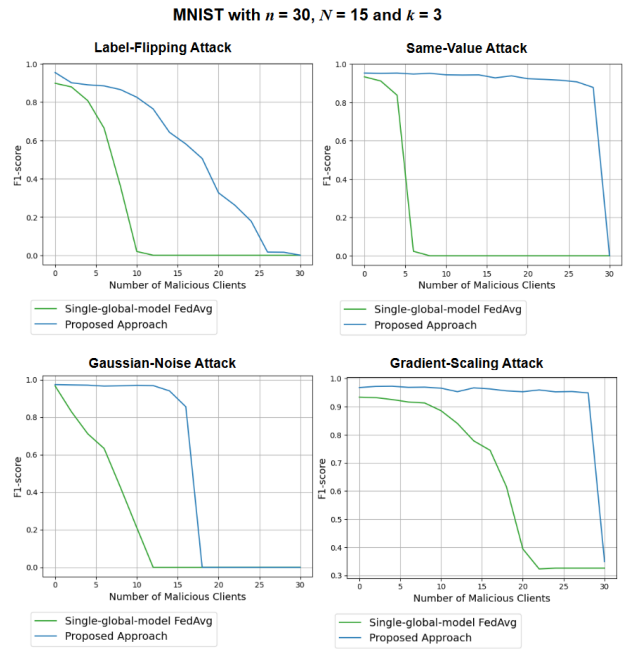

Figure 10. Comparison of the Single-global-model FedAvg approach with the proposed approach using the MNIST dataset, with $n = 30$, $N = 15$ and $k = 3$, for Label-Flipping, Same-Value, Gaussian-Noise and Gradient-Scaling attacks.

resilience of the system. In addition, the proposed approach does not distinguish between model degradation from poor data and malicious behavior, which could result in the exclusion of benign contributions. Moreover, strategically distributed malicious clients can still influence multiple global models, reducing the system's ability to filter out poisoned models.

Declarations

Authors' Contributions

Conceptualization, BOM and BBZ; Data curation, BOM; Formal analysis, BOM; Investigation, BOM; Methodology, BOM and BBZ; Project administration, BBZ; Software, BOM; Resources, BOM and BBZ; Supervision, BBZ; Validation, BOM and BBZ; Visualization, BOM; Writing – original draft, BOM; Writing – review & editing, BOM and BBZ; All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

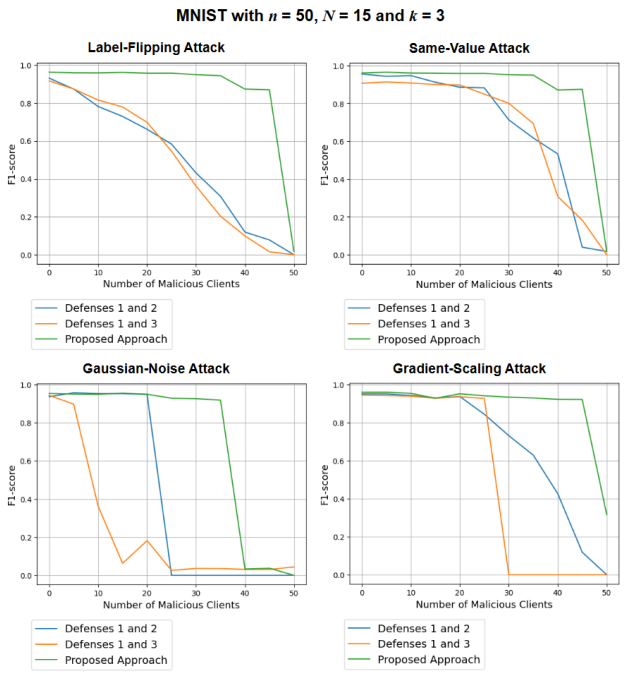

Figure 11. Comparison of the proposed approach with two variations, each isolating different defense combinations, using the MNIST dataset with $n = 50$, $N = 15$, and $k = 3$, for Label-Flipping, Same-Value, Gaussian-Noise, and Gradient-Scaling attacks. This comparison evaluates the performance of Defenses 1 and 2, and Defenses 1 and 3, separately.

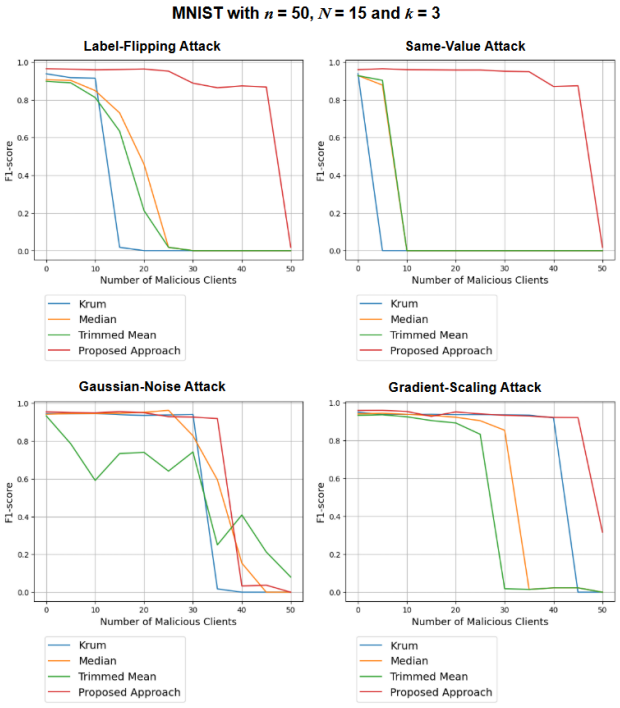

Figure 13. Comparison of the proposed approach with Krum, Trimmed Mean, and Median, using the MNIST dataset with $n = 50$, $N = 15$, and $k = 3$, for Label-Flipping, Same-Value, Gaussian-Noise, and Gradient-Scaling attacks.

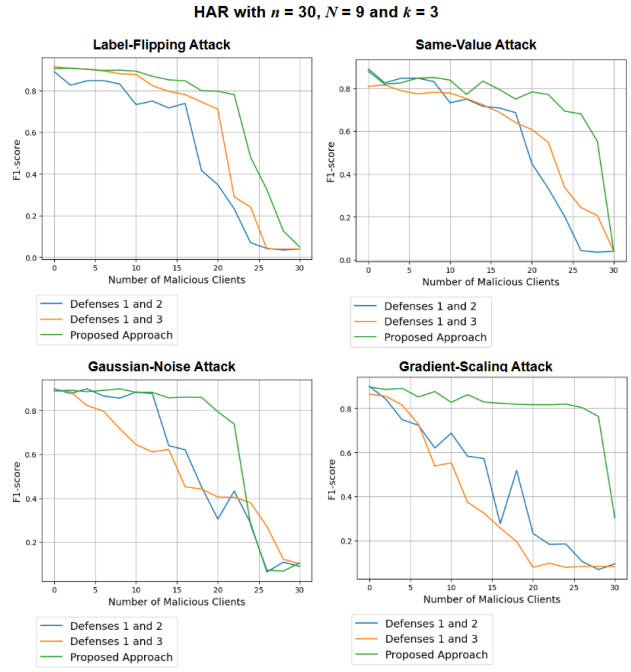

Figure 12. Comparison of the proposed approach with two variations, each isolating different defense combinations, using the HAR dataset with $n = 30$, $N = 9$, and $k = 3$, for Label-Flipping, Same-Value, Gaussian-Noise, and Gradient-Scaling attacks. This comparison evaluates the performance of Defenses 1 and 2, and Defenses 1 and 3, separately.

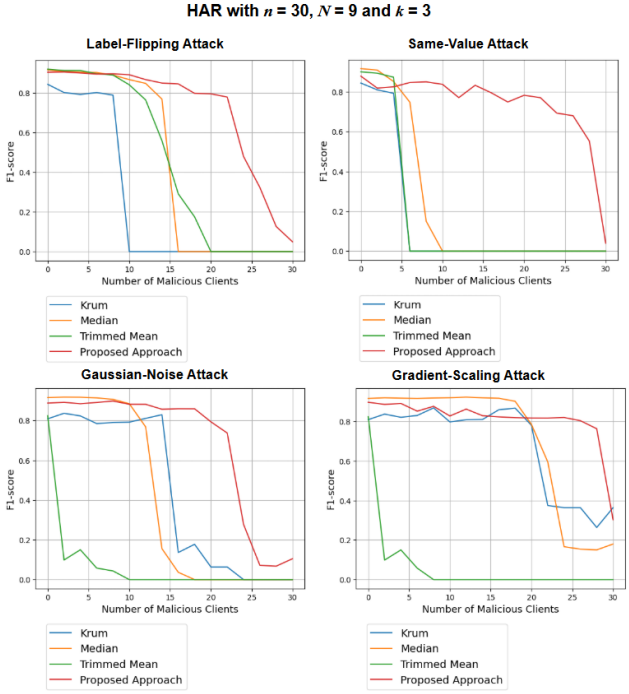

Figure 14. Comparison of the proposed approach with Krum, Trimmed Mean, and Median, using the HAR dataset with $n = 30$, $N = 9$, and $k = 3$, for Label-Flipping, Same-Value, Gaussian-Noise, and Gradient-Scaling attacks.

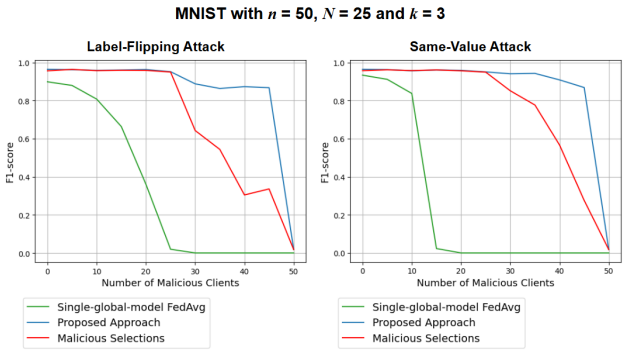

Figure 15. Analysis of the impact of clients issuing malicious votes using the MNIST dataset with $n = 50$, $N = 25$ and $k = 3$.

Funding

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) – Finance Code 001.

Availability of data and materials

The datasets (and/or softwares) generated and/or analysed during the current study will be made upon request.

References

Andreina, S., Marson, G. A., Möllering, H., and Karame, G. (2020). Baffle: Backdoor detection via feedback-based federated learning. *CoRR*, abs/2011.02167. Available at:`https://arxiv.org/abs/2011.02167`.

Blanchard, P., El Mhamdi, E. M., Guerraoui, R., and Stainer, J. (2017). Machine learning with adversaries: Byzantine tolerant gradient descent. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc. Available at:`https://proceedings.neurips.cc/paper_files/paper/2017/file/f4b9ec30ad9f68f89b29639786cb62ef-Paper.pdf`.

Bouacida, N. and Mohapatra, P. (2021). Vulnerabilities in federated learning. *IEEE Access*, 9:63229–63249. DOI: 10.1109/ACCESS.2021.3075203.

Cao, X., Jia, J., and Gong, N. Z. (2021). Provably secure federated learning against malicious clients. *CoRR*, abs/2102.01854. DOI: 10.1609/aaai.v35i8.16849.

Cao, X., Zhang, Z., Jia, J., and Gong, N. Z. (2022). Flcert: Provably secure federated learning against poisoning attacks. *IEEE Transactions on Information Forensics and Security*, 17:3691–3705. DOI: 10.1109/TIFS.2022.3212174.

Che, C., Li, X., Chen, C., He, X., and Zheng, Z. (2022). A decentralized federated learning framework via committee mechanism with convergence guarantee. *IEEE Transactions on Parallel and Distributed Systems*, 33(12):4783–4800. DOI: 10.1109/tpds.2022.3202887.

Deng, L. (2012). The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142. DOI: 10.1109/MSP.2012.2211477.

Fang, M., Cao, X., Jia, J., and Gong, N. (2020). Local model poisoning attacks to Byzantine-Robust federated learning. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 1605–1622. USENIX Association. Available at:`https://www.usenix.org/conference/usenixsecurity20/presentation/fang`.

Jebreel, N. M., Domingo-Ferrer, J., Blanco-Justicia, A., and Sánchez, D. (2024). Enhanced security and privacy via fragmented federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, 35(5):6703–6717. DOI: 10.1109/tnnls.2022.3212627.

Li, S., Ngai, E., and Voigt, T. (2023). Byzantine-robust aggregation in federated learning empowered industrial iot. *IEEE Transactions on Industrial Informatics*, 19(2):1165–1175. DOI: 10.1109/TII.2021.3128164.

Liu, B., Ding, M., Shaham, S., Rahayu, W., Farokhi, F., and Lin, Z. (2021). When machine learning meets privacy: A survey and outlook. *ACM Comput. Surv.*, 54(2). DOI: 10.1145/3436755.

Marcozzi, M. and Mostarda, L. (2024). Analytical model for performability evaluation of practical byzantine fault-tolerant systems. *Expert Systems with Applications*, 238:121838. DOI: 10.1016/j.eswa.2023.121838.

McMahan, B. and Ramage, D. (2017). Federated learning: Collaborative machine learning without centralized training data. Available at:`https://research.google/blog/federated-learning-collaborative-machine-learning-without-centralized-training-data` Accessed on june 06, 2024.

Reyes-Ortiz, J., Anguita, D., Ghio, A., Oneto, L., and Parra, X. (2012). Human Activity Recognition Using Smartphones. DOI: 10.24432/C54S4K.

Takahashi, K., Yamamoto, K., Kuchiba, A., and Koyama, T. (2022). Confidence interval for micro-averaged f 1 and macro-averaged f 1 scores. *Applied Intelligence*, 52(5):4961–4972. DOI: 10.1007/s10489-021-02635-5.

Tolpegin, V., Truex, S., Gursoy, M. E., and Liu, L. (2020). Data poisoning attacks against federated learning systems. In Chen, L., Li, N., Liang, K., and Schneider, S., editors, *Computer Security – ESORICS 2020*, pages 480–501, Cham. Springer International Publishing. DOI: 10.1007/978-3-030-58951-6$_2$4.

Wang, Z., Kang, Q., Zhang, X., and Hu, Q. (2022). Defense strategies toward model poisoning attacks in federated learning: A survey. DOI: 10.1109/wcnc51071.2022.9771619.

Witt, L., Heyer, M., Toyoda, K., Samek, W., and Li, D. (2023). Decentral and incentivized federated learning frameworks: A systematic literature review. *IEEE Internet of Things Journal*, 10(4):3642–3663. DOI: 10.1109/JIOT.2022.3231363.

Xia, G., Chen, J., Yu, C., and Ma, J. (2023). Poisoning attacks in federated learning: A survey. *IEEE Access*, 11:10708–10722. DOI: 10.1109/ACCESS.2023.3238823.

Xu, C., Jia, Y., Zhu, L., Zhang, C., Jin, G., and Sharif, K. (2022). Tdfl: Truth discovery based byzantine ro-

bust federated learning. *IEEE Transactions on Parallel and Distributed Systems*, 33(12):4835–4848. DOI: 10.1109/TPDS.2022.3205714.

Yang, Q., Liu, Y., Chen, T., and Tong, Y. (2019). Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol.*, 10(2). DOI: 10.1145/3298981.

Yin, D., Chen, Y., Kannan, R., and Bartlett, P. (2018). Byzantine-robust distributed learning: Towards optimal statistical rates. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5650–5659. PMLR. DOI: 10.48550/arxiv.1803.01498.

Zhang, C., Xie, Y., Bai, H., Yu, B., Li, W., and Gao, Y. (2021). A survey on federated learning. *Knowledge-Based Systems*, 216:106775. DOI: 10.1016/j.knosys.2021.106775.

Zhang, Z., Li, J., Yu, S., and Makaya, C. (2023). Safelearning: Secure aggregation in federated learning with backdoor detectability. *IEEE Transactions on Information Forensics and Security*, 18:3289–3304. DOI: 10.1109/tifs.2023.3280032.