A Triad of Defenses to Mitigate Poisoning Attacks in Federated Learning [*]

Blenda Oliveira Mazetto[1], **Bruno Bogaz Zarpelão**[1]

[1]Computer Science Department – State University of Londrina
Londrina – PR – Brazil

{blenda.mazetto, brunozarpelao}@uel.br

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, checking 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, with each group generating a different global model. Each client then receives all global models and selects the one with the best predictive performance to continue training. The selected global models are updated by the clients and then submitted again to the central server, which aggregates these models. 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 show that our method can effectively mitigate poisoning attacks without compromising the global model's results.*

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) enables the participants to collaboratively learn a shared learning model while keeping all the data on the device, 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 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 the server receives the local models of all clients, it calculates the new global model by aggregating the received models. The above steps will be repeated until the algorithm converges. After that, the clients

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

use the learned global model to make predictions for new inputs during the inference phase [Zhang et al. 2021].

Despite its benefits, the FL paradigm is vulnerable to poisoning attacks. More specifically, malicious clients may infiltrate in the FL scheme to corrupt the global model. As a result, the corrupted global model would have a low accuracy, which significantly decreases the performance of the trained model on the inference set [Tolpegin et al. 2020], [Bouacida and Mohapatra 2021]. There are two main types of poisoning attacks. In data poisoning attacks, attackers inject malicious data into their local training datasets. In model poisoning attacks, attackers directly manipulate the gradients or model updates they send to the central server. An attack in which clients of a distributed system begin acting maliciously is called a Byzantine attack [Wang et al. 2022], [Fang et al. 2020].

Robustness against Byzantine attacks and the preservation of security and privacy in FL have been central research topics. 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. Ultimately, Andreina et al. [2020] uses 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 mitigate poisoning attacks in FL: dividing clients into groups, checking global model performance, 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 rest 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 the Truth Discovery aggregation, which is an unsupervised iterative data aggregation technique designed to determine the most reliable updates, to identify and eliminate suspicious local models. Complementing this category, Li et al. [2023] introduce AutoGM, a secure aggregation rule that enhances the robustness of the Geometric Median (GM) method. AutoGM is applied in both traditional FL paradigms and Personalized FL paradigms, addressing data heterogeneity challenges by learning personalized models for each device.

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 includes 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.

Continuing with the model analysis strategy, aimed at enhancing security and privacy, Jebreel et al. [2024] propose a novel lightweight protocol enabling participants to privately exchange and mix random fragments of their updates before submitting them to the server. Since this exchange preserves the original coordinate positions of the parameters, the server can accurately calculate the average of the mixed updates, preserving the integrity of the global model. Additionally, global and local reputation vectors are used to select trustworthy clients and avoid the influence of attackers, strengthening the robustness of federated training.

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. They use model analysis techniques to ensure the security of the aggregation process, preventing collaboration among attackers.

In [Cao et al. 2021], an ensemble global model is proposed that uses majority voting among multiple global models trained on subsets of clients, also leveraging the defense strategy based on dividing clients into groups. This approach ensures that when the majority of clients are honest, the resulting model is robust against a limited number of malicious clients. For instance, their method can achieve a certified accuracy of 88% on MNIST when 20 out of 1,000 clients are malicious. Similarly, Cao et al. [2022] group clients into probabilistic or deterministic subgroups to train multiple global models. Each global model is used to predict input labels, ensuring that the final aggregation is robust against the influence of malicious clients.

Finally, Andreina et al. [2020] explore a unique FL characteristic, the multiple private datasets, in a defense strategy that relies on the model 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.

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 global model 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. Table 1 compares the reviewed studies and the proposed approach.

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 and Same-Value

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. Figure 1 shows the grouping process.

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.

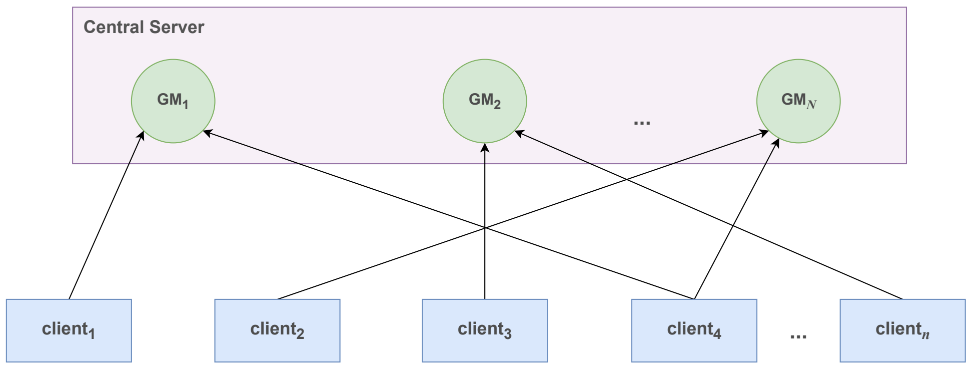

Figure 1. Example of the group division process with $n = 5$, $N = 3$ and $k = 2$.

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 that 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. 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.

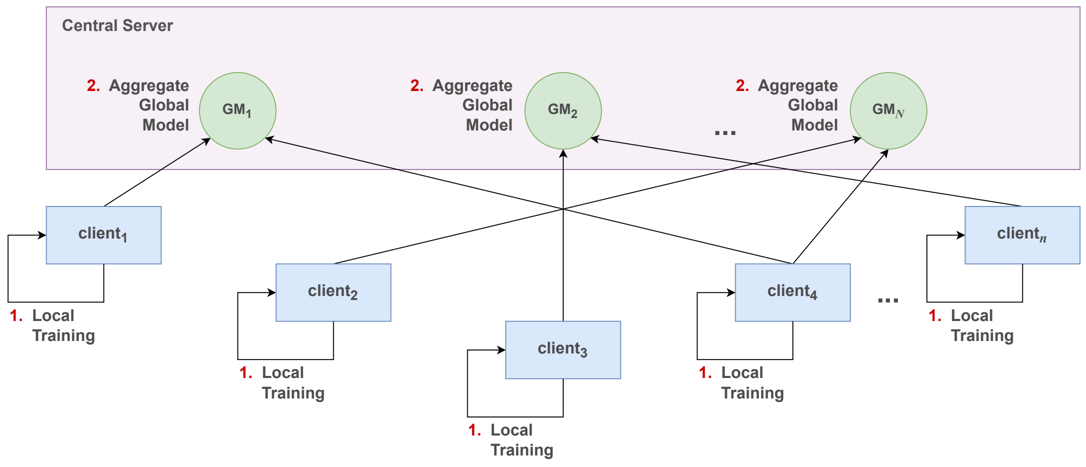

Figure 2. First and second steps of our proposal, where we can visualize the clients training a local model with their private dataset and the central server aggregating the local models into the global models according to the group division.

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. 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. Global models produced by compromised groups tend to perform worse in terms of F1-score, as they were affected by poisoned models. Avoiding these models provides an

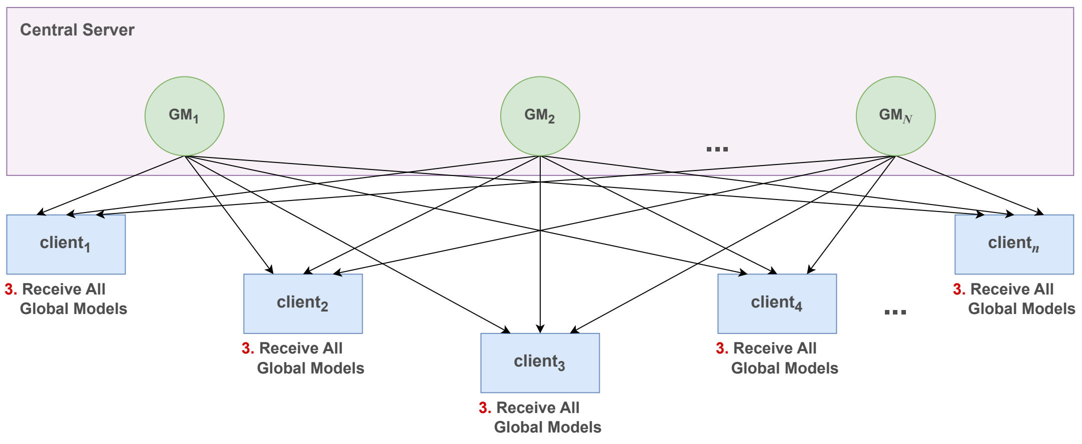
Central Server
GM$_1$
GM$_2$
GM$_N$
...
client$_1$
3. Receive All Global Models
client$_2$
3. Receive All Global Models
client$_3$
3. Receive All Global Models
client$_4$
3. Receive All Global Models
...
client$_n$
3. Receive All Global Models

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

additional layer of security and aims to preserve the integrity of the global model. Furthermore, this solution allows us to use the availability of clients' private datasets without compromising privacy.

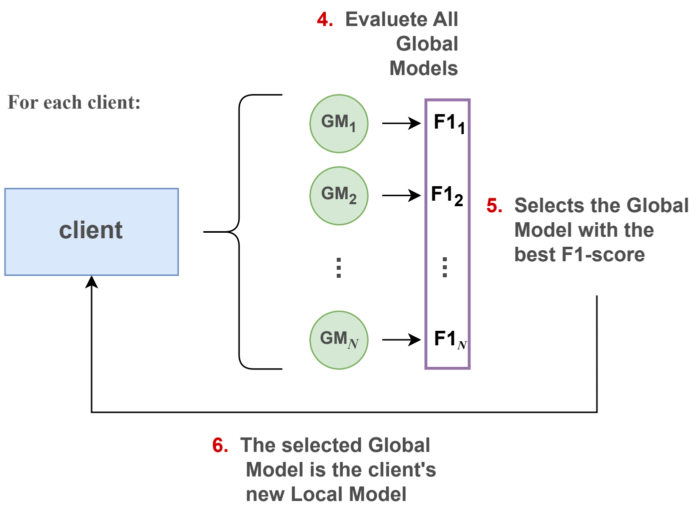
4. Evaluete All Global Models
For each client:
client
GM$_1$
GM$_2$
GM$_N$
F1$_1$
F1$_2$
F1$_N$
5. Selects the Global Model with the best F1-score
6. The selected Global Model is the client's new Local Model

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.

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 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. This scheme can be observed 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.

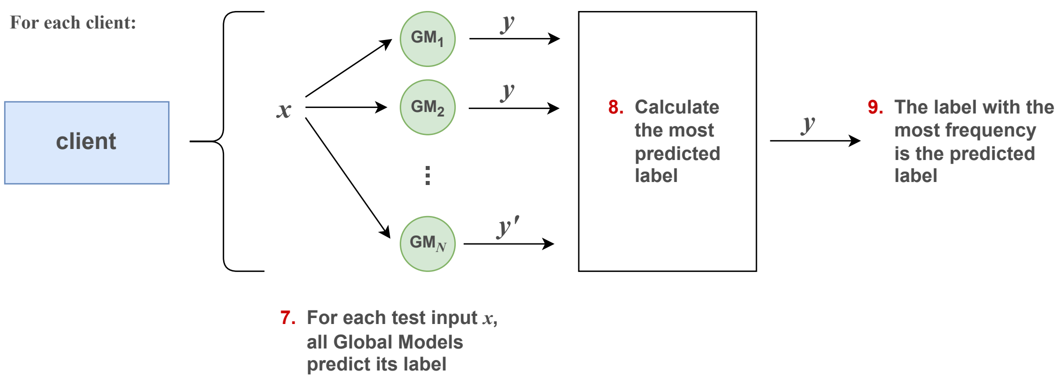
For each client:
client
x
GM1
GM2
GMN
y
y
y'
8. Calculate the most predicted label
y
9. The label with the most frequency is the predicted label
7. For each test input x, all Global Models predict its label

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.

3.1. Dealing with Malicious Selections

As the global models are evaluated by clients, malicious clients may deliberately lie to compromise the entire system. In the approach described earlier, during the performance evaluation step, a malicious client might, instead of selecting the global model with the highest F1-score, do the opposite, choosing the global model with the lowest F1-score as the new local model. However, the results are not affected as long as the number of malicious clients, m, is less than $\frac{n}{2}$.

This occurs because, in the voting system, the most frequent label wins. Therefore, if the majority of clients are benign ($m < \frac{n}{2}$), the output label is more likely to be the correct one, minimizing the influence of malicious clients. This way, even if some clients attempt to sabotage the process, the benign majority ensures the integrity and accuracy of the final global model.

4. Evaluation and Results

In this section, we present the results from a series of experiments carried out on the approach proposed in Section 3 and the experimental setup. Since in our approach the inference step is performed on each client's device, each client will generate their own metrics individually. However, for visualization purposes, we calculate an overall F1-score of all clients. We chose to calculate the F1-score because it provides a balanced measure that accounts for both false positives and false negatives, as it combines precision (the accuracy of positive predictions) and recall (the ability to find all positive instances) into a single metric.

4.1. Experimental Setup

Datasets: We utilize the MNIST and Human Activity Recognition Using Smartphones (HAR) datasets.

- 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, scenarios with two variations in the number of clients were tested, $n = 30$ and $n = 50$. For the 30-client scenario, three variations of N were tested: $N = 7$, $N = 13$, and $N = 17$. In the 50-client variation, three variations of N were also tested: $N = 15$, $N = 25$, and $N = 35$. Experiments with the HAR dataset 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 $N = 21$. Moving on to the number of clients per group, we selected values that closely aligned with those reported in the literature, so all scenarios described above were tested with 3 and 5 clients per group ($k = 3$ and $k = 5$). The chosen aggregation method was FedAvg. The FedAvg aggregation is calculated using the average of local models. This approach 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, this architecture was proposed by [Cao et al. 2021]. The parameters were 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 two poisoning attacks with distinct strategies, one targeting the model and the other the data.

- Same-Value Attack: this attack targets the learning model by setting all its parameters to a single value, zero in our case. It aims to compromise the model's functionality by nullifying its ability to learn and make accurate predictions. In our scenario, this attack is performed on the local model of a malicious client whenever it sends its local model to the central server.
- Label-Flipping Attack: this attack targets the training data by altering the labels of training samples. The goal is to induce a local model with wrongly labeled data. For the MNIST dataset, malicious clients changed their data labels to "0". For the HAR dataset, malicious clients altered their data labels to "WALKING".

4.2. Results

Single-global-model FedAvg vs. Our Approach: As a baseline, we implemented an FL setup based on the FedAvg aggregation algorithm without any defense strategy, which we named "Single-global-model FedAvg". Using this basic FL setup, we can assess whether our defense strategies improve the FL's capacity to resist the poisoning attacks. The results for Single-global-model FedAvg are identified in the plots as "FL".

Figures 6, 7, and 8 show a comparison between the F1-score results of our proposal and the Single-global-model FedAvg approach. In Figure 6, we can observe the results of our proposal on the MNIST dataset with $n = 50$. These tests were conducted considering Label-Flipping and Same-Value attacks. Additionally, we varied both the number of groups (N) and the number of clients per group (k). Figure 7 shows the results obtained from experiments using the MNIST dataset with 30 clients ($n = 30$), also with variations in N and k. Figure 8 illustrates the results obtained from tests on the HAR dataset. As this dataset is already naturally federated, we kept $n = 30$ and conducted tests with $N = 9$, $N = 15$, and $N = 21$. Furthermore, all the tests also show the variation in the number of clients per group ($k = 3$ and $k = 5$).

It is important to note that compared to the Single-global-model approach, our method incurs higher computational costs due to the additional operations required to mitigate attacks. The enhanced security measures necessitate more extensive computations, which results in increased processing time and resource usage.

The Impact of n, N, and k: In Figures 6 and 7, we used the same dataset (MNIST) but varied the number of clients ($n = 50$ and $n = 30$, respectively). Therefore, it is possible to observe the impact of varying the number of clients, especially considering Figures 6a and 7b, which use almost the same number of groups ($N = 15$ and $N = 13$). We can observe a slight difference in mitigation capacity. For 30 clients, performance started to decline when 80% of the clients were malicious. For 50 clients, performance began to degrade when about 90% of the clients were compromised.

The impact of varying the number of groups (N) can be better observed in Figure 9, where the values of n and k are fixed for better comparison. We can conclude that

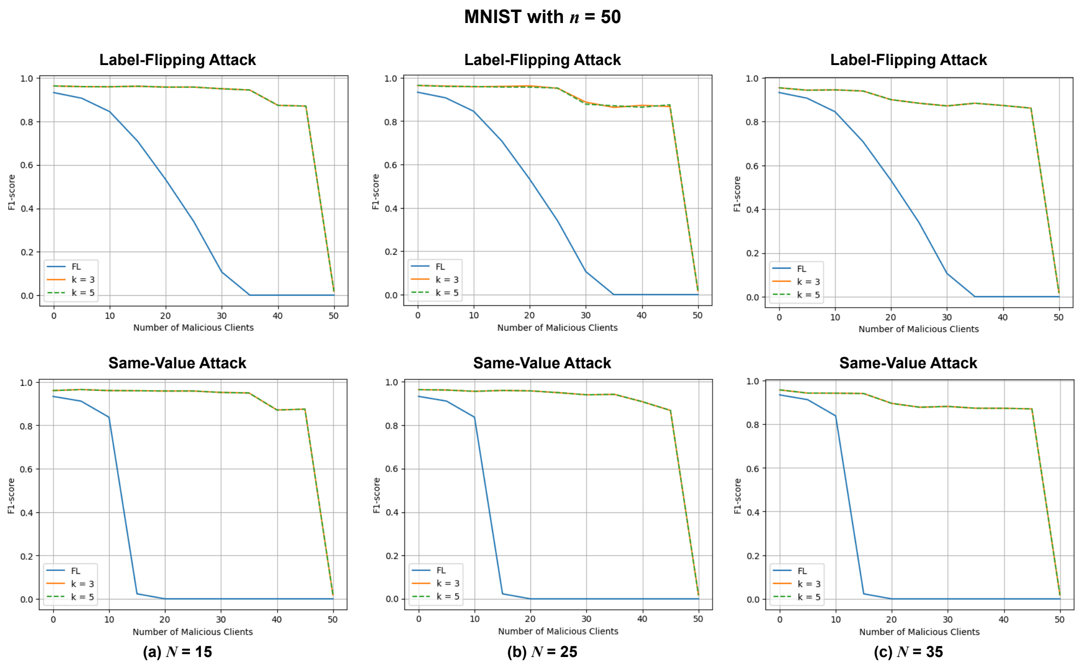

Figure 6. Comparing the results of the Single-global-model FedAvg approach with our Approach using the MNIST dataset with $n = 50$.

there is no difference in mitigation capacity. Performance consistently starts to decline at the same point. This indicates that we can use the smallest N among them, as the system would have fewer groups and need to work with fewer global models, reducing the computational cost. Figures 6, 7, and 8 all show the impact of varying the number of clients per group (k), and in all results, the variation of k was barely noticeable. In this scenario, it is more prudent to employ $k = 5$ to enhance the number of local models contributing to the development of the global model.

In all scenarios, the proposed approach presented a high mitigation capacity. In the case of MNIST, the system handled well with 80% to 90% of malicious clients. For HAR, it tolerated just over 2/3 of the malicious clients.

Dealing with different attacks: We analyze the effectiveness of our proposal against different attacks, namely the Label-Flipping Attack and Same-Value Attack. Each tested attack represents a distinct strategy, one targeting the model and the other the data. When testing our proposal against the Label-Flipping attack, we demonstrated that our approach can mitigate malicious label alterations, preserving the integrity of the trained model. Conversely, against the Same-Value attack, we demonstrated that our approach can mitigate the impact of setting all parameters to zero, without drastically compromising the model's functionality. Our solution proved robust in both scenarios, showcasing its ability to prevent diverse threats, thereby ensuring the reliability and accuracy of the trained models. Figures 6, 7, and 8 provide comparisons using both the Label-Flipping attack and the Same-Value attack.

Dealing with malicious selections: As mentioned in Section 3.1, malicious clients can deliberately lie. During the performance evaluation phase of the proposed

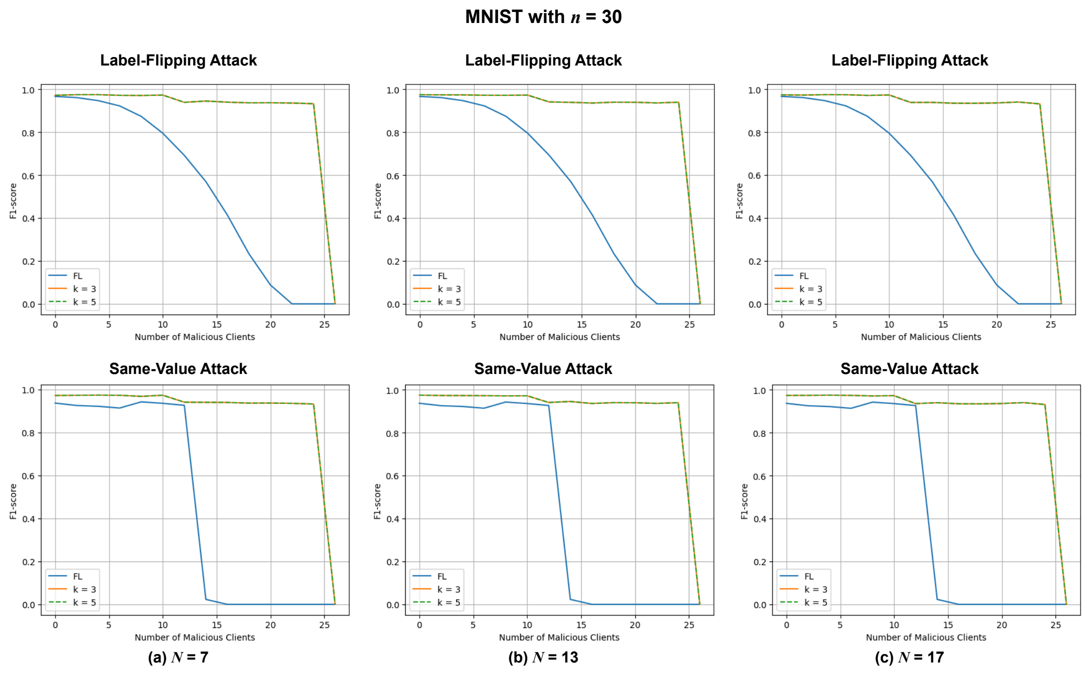

Figure 7. Comparing the results of the Single-global-model FedAvg approach with our Approach using the MNIST dataset with $n = 30$.

approach, a malicious client might choose the global model with the worst F1-score instead of the best one, using it as the new local model. In Figure 10, we can observe the results obtained in this case and analyze that even with malicious votes, our approach remains superior to the Single-global-model FedAvg approach. As long as the number of malicious clients is less than $\frac{n}{2}$, the learning model maintains a high F1-score.

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 models 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 two different attacks, one targeting the model and the other the data. Our solution proved robust in both scenarios, showcasing its ability to protect against both threats. All results obtained demonstrated improvements over a basic FL approach without defenses. For the MNIST dataset, the results showed an F1-score above 0.8 even with 90% of malicious clients, and for the HAR dataset, the F1-score results were above 0.8 even with 66.6% of malicious clients. For future work, we intend to expand our proposal to non-IID scenarios, explore new methods of protecting privacy, and finally, carry out tests with different datasets and aggregation rules.

References

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

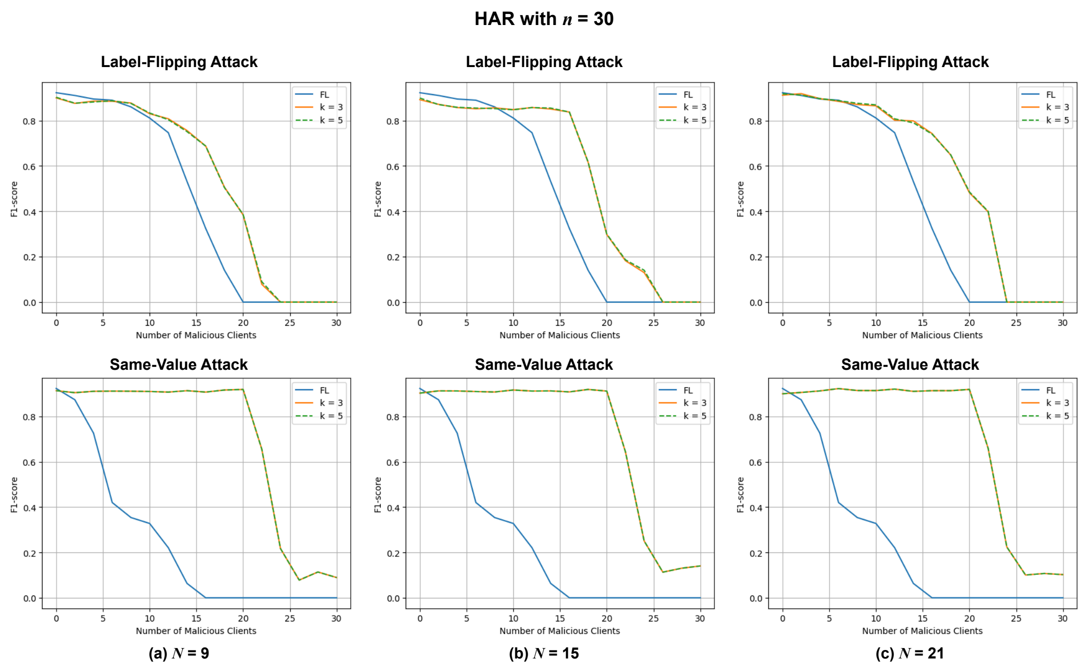

Figure 8. Comparing the results of the Single-global-model FedAvg approach with our Approach using the HAR dataset with $n = 30$.

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

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

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.

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.

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

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.

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.

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.

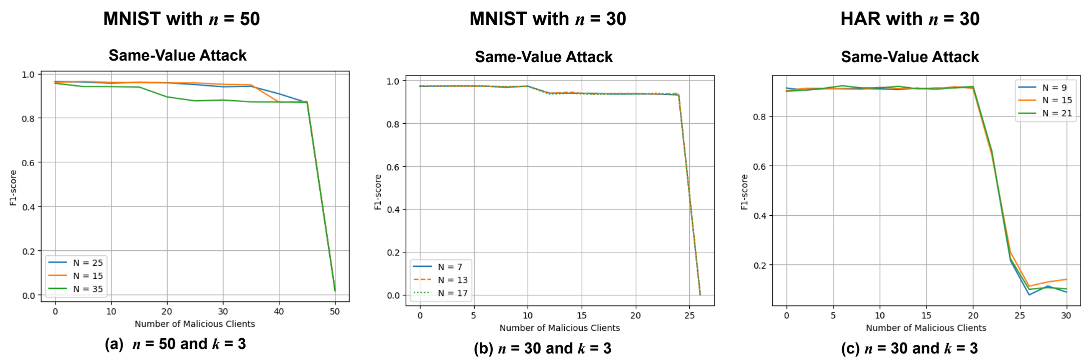

Figure 9. Impact of N on the MNIST datasets with $n = 50$, MNIST with $n = 30$ and HAR with $n = 30$.

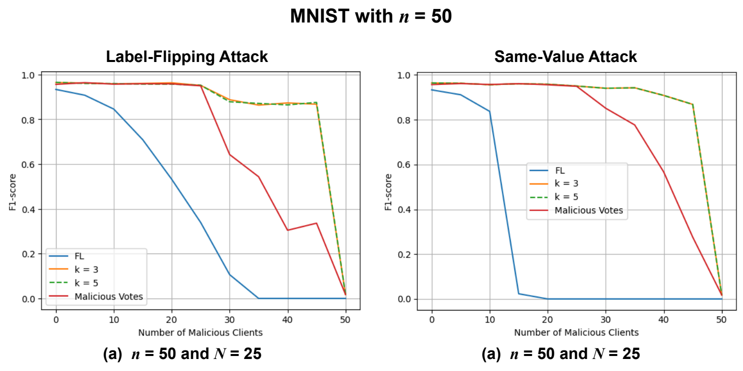

Figure 10. Analysis of the impact of clients issuing malicious votes using the MNIST dataset with $n = 50$.

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).

McMahan, B. and Ramage, D. (2017). 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. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C54S4K.

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.

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

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.

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

Xu, C., Jia, Y., Zhu, L., Zhang, C., Jin, G., and Sharif, K. (2022). Tdfl: Truth discovery based byzantine robust federated learning. *IEEE Transactions on Parallel and Distributed Systems*, 33(12):4835–4848.

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

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

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.