2023 IEEE International Conference on Big Data (BigData) | 979-8-3503-2445-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/BigData59044.2023.10386784

Defending Federated Learning Against Model Poisoning Attacks

Ibraheem Aloran
School of Computer Science
University of Windsor
Windsor, Canada
alorani@uwindsor.ca

Saeed Samet
School of Computer Science
University of Windsor
Windsor, Canada
saeed.samet@uwindsor.ca

***Abstract*—Federated Learning (FL) is a machine learning framework that allows multiple clients to contribute their data to a single machine learning model without sacrificing their privacy. Although FL addresses some security issues, it is still susceptible to model poisoning attacks where malicious clients aim to corrupt the main learning model by sending poisoned updates. Byzantine-robust methods are defenses that aim to prevent corruption of the main learning model by tolerating a certain number of malicious clients. However, they can only resist a small number of malicious clients. This proposed method uses Gap statistics to determine the optimal number of clusters to cluster clients. This will improve the detection accuracy of malicious clients while preventing the mislcassification of honest clients in a Federated Learning setting. Our experiments so far show us an improvement over the base method.**

***Index Terms*—Machine Learning, Federated Learning, Clustering, Privacy and Security, Model Poisoning Attack.**

I. Introduction

Traditional machine learning (ML) approaches involve various security and privacy issues with respect to users' data. Federated learning (FL) [3,20] aims to solve this issue by allowing the data to stay decentralized while still training a machine learning model. FL helps in situations where there is a large amount of data or clients that are distributed across multiple locations such as hospitals and smartphones [3]. In this case it would be hard to collect all the data while maintaining the privacy and security of each client as well as the resources and space needed to store and process such data. FL is a client-server architecture where the server is responsible for initializing the model, sharing the model and parameters to clients, and aggregating the resulting parameters received from clients. The clients are given the model by the server and are tasked with training it with their own data.

FL is an iterative process where it improves the ML model each iteration. The FL process involves three main steps: (1) The global ML model at the central server is initialized with all of its parameters and weights and is shared with a sample or all of the clients in the FL environment. (2) Each client that receives the model trains it with their own individual data and sends their updated local models back to the server. (3) The server then aggregates and updates the global ML model using a certain aggregation rule. Once the global model is updated, it is then shared again to the clients for the next iteration. Steps 2 and 3 are repeated to keep the global model updated across the clients. A common aggregation rule that is used in a non-adversarial setting is to average all the local model parameters received by the clients. Figure 1 shows the general process of Federated Learning.

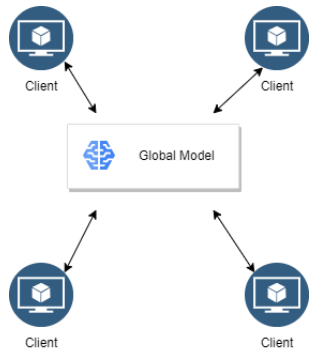

Fig. 1. General Process of Federated Learning. The server initializes the global model and sends it to the clients. The clients train the model on their data and send it back to the server. The server aggregates all the received models and shares the new global model to the clients.

Although federated learning helps keep data secure and decentralized, it still has its own vulnerabilities. The main threat that is going to be addressed in this paper is model poisoning attacks. Model poisoning attacks aim to deviate the global model by compromising local model updates during training. Attackers can inject malicious clients into the federated learning system or compromise genuine clients to craft local model updates and send it to the central server where it is aggregated and could poison the global model. Several defenses were developed to prevent this attack. However, some of these defenses require the central server to hold a validation dataset which is not always practical, while others can only tolerate a small number of malicious clients. FLDetector[8] addresses this issue by detecting malicious clients based on their model-update history. One disadvantage to this method is that it misclassifies a fraction of honest clients as malicious. Thus, the need for a robust defense that can efficiently eliminate malicious clients in an FL setting is still in demand.

The main contributions of this paper are as follows:

- We improve FLDetector by proposing a new clustering

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on October 22,2024 at 13:28:00 UTC from IEEE Xplore. Restrictions apply.

method to reduce the misclassification of honest clients while still removing majority of malicious clients. Our solution leverages the Gap Statistic[10] to get the optimal number of clusters.
- We present implementation and experimentation details of the proposed method.

The remainder of the paper is organized as follows. Section II provides prerequisite knowledge used in this paper. Section III describes existing works including attacks and defenses. Section IV outlines our methodology and experimental setup. Section V includes expected results. Lastly, Section VI concludes the paper.

II. Background

In this section we give a brief overview of the method we are going to improve and the knowledge and tools used.

A. Model Poisoning Attacks

Model poisoning attacks are split into two groups and are categorized as follows [2].

Targeted Attacks: In these attacks, the attacker embeds a trigger into the input and trains the model to classify it to a certain label. For example, the trigger could be a group of pixels located at the corner of an image. The attacker plants this trigger in multiple images and trains the model with these images. These attacks are harder to detect because they do not directly reduce the testing accuracy of the model. They can still correctly classify other classes but misclassify the label chosen by the attacker. Examples of Targeted Model Poisoning attacks include Scaling Attack [5], where the attacker scales the model updates before sending them to increase its maliciousness, and Distributed Backdoor Attack [11], where the attacker divides the trigger into different patterns and distributes them into the training data.

Untargeted Attacks: These attacks aim to lower the accuracy of the global model as much as possible. The attacker can craft or alter local model updates in order to lower the global model's accuracy. Fang et al. [4] showed how the attacker can map the poisoning attack to an optimization problem to minimize the difference between the original and poisoned model. This can make it difficult for the attack to be mitigated or detected. These attacks can be used against any aggregation rule and can reduce the testing accuracy of the global model.

B. Byzantine-Robust FL Aggregation Methods

As mentioned earlier, the mean aggregation rule is a commonly used rule in a non-adversarial setting which averages the weights or parameters received from the clients to craft the global model. However, this method is not robust in an adversarial setting and an attacker can easily manipulate the global model by compromising one client. To counter this, researchers have developed aggregation rules to withstand Byzantine failures. We will consider three common Byzantine-robust FL aggregation methods.

Krum [12]: Krum selects one of the client models that is most similar to the rest of the client models. Suppose there are at most c compromised clients. For each local model w_i, the central server computes n-c-2 local models that are closest to w_i using Euclidean distance. The local model with the smallest sum of squared distance becomes the global model.

Median [13]: For each j^{th} model parameter, the central server sorts the j^{th} parameters of the n local models and takes the median as the j^{th} parameter for the global model.

Trimmed Mean [13]: This method is similar to Median and aggregates each model parameter independently. For each model parameter j, the central server sorts the j^{th} parameters of the n models where j_{ij} is the j^{th} parameter for the i^{th} local model. It removes the b smallest and b largest of them and computes the mean of the remaining n-2b parameters and sets it as the j^{th} parameter for the global model.

C. FLDetector

In this section we will give a brief explanation of how FLDetector [8] detects malicious clients. We denote g_i^t as the model update sent from client i in iteration t and w_t as the global model at the beginning of iteration t. Let $\Delta w_t = w_t - w_{t-1}$ be the global model difference in iteration t and $\Delta g_t = g_t - g_{t-1}$ be the global model update difference in iteration t. We denote $\Delta W_t = \{\Delta w_{t-N}, \Delta w_{t-N+1}, \dots, \Delta w_{t-1}\}$ as the global model differences in the past N iterations, and $\Delta G_t = \{\Delta g_{t-N}, \Delta g_{t-N+1}, \dots, \Delta g_{t-1}\}$ as the global model update differences in the past N iterations in iteration t. N is called the window size.

Using the L-BFGS algorithm [18], we can approximate the Hessian matrix H^t = L-BFGS(ΔW_t,ΔG_t) in iteration t. Based on the estimated Hessian H^t, FLDetector can predict a client i's model update in iteration t using the Cauchy mean value theorem [19]:

$$\hat{g}_i^t = g_i^{t-1} + H^t(w_t - w_{t-1}) \tag{1}$$

as shown in [8]. A suspicious score is assigned to each client by calculating the Euclidean distance between the predicted model update $\hat{g}_i^t$ and the received model update g_i^t. We denote d^t the vector of n Euclidean distances for n clients in iteration t, i.e.,

$$d^t = [||\hat{g}_1^t - g_1^t||_2, ||\hat{g}_2^t - g_2^t||_2, \dots ||\hat{g}_n^t - g_n^t||_2]. \tag{2}$$

as shown in [8]. The vector d^t is normalized as $\hat{d}^t = d^t/||d^t||_1$. Finally, we assign the suspicious score s_i^t for client i in iteration t over the past N iterations as the average normalized Euclidean distance, i.e.,

$$s_i^t = (\sum_{r=0}^{N-1} \hat{d}_i^{t-r})/N. \tag{3}$$

as shown in [8]. The next step is to cluster the clients based on their assigned suspicious score. Gap statistics [10] is used in every iteration to check if the clients can be clustered into more than one cluster. If this is true, then the clients are clustered into two clusters using k-means. The cluster with the larger average suspicious score is labelled as malicious and the server removes the clients in the malicious cluster and restarts training. Algorithms 1 and 2 show the pseudo codes

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on October 22,2024 at 13:28:00 UTC from IEEE Xplore. Restrictions apply.

for Gap Statistics and FLDetector respectively. More details and description for this method are included in the original paper.

Algorithm 1 Gap Statistics

Require: Clients' suspicious scores s^t, number of sampling B, maximum number of clusters K, and number of clients n.

Ensure: Number of clusters k.

for $k = 1, 2, ..., K$ **do**
  Apply linear transformation on s^t so that the minimum of s^t equals 0 and maximum of s^t equals 1.
  Apply k-means on the suspicious scores to get clusters C_i and means u_i.
  $V_k = \sum_{i=1}^{k} \sum_{x_j \in C_i} ||x_j - u_i||^2$
  for $b = 1, 2, ...B$ **do**
    Sample n points uniformly in [0,1].
    Perform k-means and calculate $V^*_{kb} = \sum_{i=1}^{k} \sum_{x_{jb} \in C_{ib}} ||x_{jb} - u_{ib}||^2$
  end for
  $Gap(k) = (1/B) \sum_{i=1}^{B} log(V^*_{kb}) - log(V_k)$
  $v' = (\sum_{i=1}^{B} log(V^*_{kb}))/B$
  $sd(k) = (((1/B) \sum_{i=1}^{B} (log(V^*_{kb})) - v')^2)^{1/2}$
  $s'_k = (((1 + B)/B) sd(k))^{1/2}$
end for
$\hat{k}$ = smallest k such that Gap(k)-Gap(k+1)+$s'_{k+1} \geq 0$.
return $\hat{k}$

Algorithm 2 FLDetector

Require: Total training iterations iter and window size N.

Ensure: Detected honest clients or none.

for $t = 1, 2, ..., iter$ **do**
  $\hat{H}^t = L - BFGS(\Delta W_t, \Delta G_t)$.
  for $i = 1, 2, ...n$ **do**
    $\hat{g}^t_i = g^t_i - 1 + \hat{H}^t(w_t - w_(t-1))$.
  end for
  $d^t = [||\hat{g}^t_1 - g^t_1||_2, ||\hat{g}^t_2 - g^t_2||_2, ...||\hat{g}^t_n - g^t_n||_2]$.
  $\hat{d}^t = d^t / ||d^t||_1$
  $s^t_i = (\sum_{r=0}^{N-1} \hat{d}^{t-r}_i)/N$.
  Determine the number of clusters k by Gap Statistics.
  if $k > 1$ **then**
    Perform k-means clustering based on the suspicious scores with k = 2.
    return The clients in the cluster with the larger average suspicious score as malicious.
  end if
end for
return None.

III. Related Work

This section reviews the existing research done relating to Model Poisoning attacks in FL. Fang et al. [4] performed untargeted model poisoning attacks on the Byzantine-robust methods mentioned earlier by mapping it to an optimization problem and solving it. They tailor the optimization problem for each byzantine-robust aggregation method and design attacks for each method. They show that their attacks substantially increase the error rate for the global model. They also show that their attacks are transferable between the aggregation rules, i.e. an attack on Krum could also work for trimmed mean and median. They use Loss function-based rejection (LFR) to remove local models that negatively impact the loss of the global model and Error rate-based rejection (ERR) to remove local models that negatively impact the error rate of the global model.

They conclude that their defenses cannot entirely defend against their attacks and show that in some cases their attacks still manage to compromise the global model. Another disadvantage of their defenses is that it requires the central server to have access to a validation set which is not always available in reality. They emphasize the importance for further research to develop defenses for model poisoning attacks.

Bagdasaryan et al. [5] exploited a vulnerability in the averaging aggregation rule and proposed a new type of backdoor attack called semantic backdoor attack. This attack causes the model to output an attacker chosen-label for certain inputs. They developed a method to perform these backdoor attacks on federated learning tasks for image classification and word prediction. The attacker scales up the weights by a factor to increase the impact of the attack. The attacker can produce a model that avoids detection while scoring a high accuracy on the attack and the main learning task.

They point out that a driving factor that allows backdoor attacks to avoid detection is that a model is usually measured on how well or accurate it has learned its main task. There is no metric to measure any extra information it has learned. Backdoor attacks can thrive if they don't significantly impact the model's accuracy on the main task.

Shejwalkar et al. [6] propose a new byzantine robust aggregation algorithm called Divide and Conquer (DnC) to defend against untargeted model poisoning attacks. The method uses spectral analysis and singular value decomposition to detect and filter outliers. DnC performs dimensionality reduction and then performs spectral analysis on the updates to ensure the detection of malicious updates.

They also present a framework for a new untargeted model poisoning attack on federated learning. The adversary computes a benign model using updates from honest clients. Finally, he alters the model in the malicious direction while also evading detection. They conclude that their algorithm outperforms existing aggregation methods and that DnC cannot defend at least one of their attacks by the strongest adversaries with complete knowledge of the gradients of benign clients.

Li et al. [7] argue that most existing defenses only consider the malicious updates as the global outlier to the FL system and overlook feature patterns from received updates. In this study, they propose a new defense for poisoning attacks in FL called LoMar which evaluates updates based on their parameter features. Their method can defend against both

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on October 22,2024 at 13:28:00 UTC from IEEE Xplore. Restrictions apply.

untargeted poisoning attacks and targeted poisoning attacks. Their method consists of two phases. Firstly, LoMar evaluates the update's maliciousness based on the statistical characteristics of the model parameters using k-nearest neighbors and assigns a malicious score to the client. Lastly, it develops a new aggregation rule that uses a binary controller to turn the scores into a Boolean status to filter out malicious clients. If the score is below the threshold they are removed, otherwise they are included in the global model.

From their experiments, they conclude that their defense can address both data and model poisoning attacks. However, there is no evidence that shows that their method can outperform other existing defenses. In some cases their defense works best but in other cases, other existing defense show more promising results.

Zhang et al. [8] studied defenses against model poisoning attacks on FL by detecting malicious clients. They find that existing defenses can only tolerate a small group of malicious clients and require a validation set on the server, which is not always typical in a FL setting. To address these limitations, they propose a new malicious-client detection method called FLDetector that can be combined with Byzantine-robust FL methods. FLDetector detects malicious clients by checking the consistency of their updates. The server predicts each client's model update in each iteration based on previous model updates and assigns a suspicious score for each client that is updated in each iteration. The clients are then clustered and the cluster with the larger suspicious score is classified as malicious. When a client is classified as malicious, the server stops the FL process, removes the client, and restarts the training.

They experiment and evaluate their method on three image datasets using one untargeted attack, three targeted attacks and develop an adaptive attack that is tailored to FLDetector to show that their method is still effective against adaptive attacks. They claim their results show that their method outperforms other methods for most scenarios and is still effective in others. However, their method starts to decrease in performance when around 20-30 malicious clients are introduced.

IV. Contribution

A. Motivation

Privacy and protection of personal data is an extremely important aspect, especially in machine learning. Being able to accurately and efficiently train a machine learning model without exposing personal data is imperative to developing useful models. FLDetector can detect majority of malicious clients but at a cost of falsely detecting a fraction of honest clients as malicious and removes them from the FL setting. This is due to its clustering method since every client is forced to be in one of the two clusters. Outliers or clients that are not necessarily malicious may be grouped in the malicious cluster and are therefore misclassified and removed from the FL setting. This denies the global model from learning useful data that would have been sent from honest clients. They do not consider the optimal number of clusters for the distribution of clients when clustering.

B. Problem Statement

We consider a typical Federated Learning setting where n clients participate to train a global model. Each client has their own dataset. The server first initializes the model and sends it to each client or a subset of clients. Each client then trains the model on their own dataset and sends it to the server. The server aggregates it using one of the existing aggregation methods sends the clients the updated global model.

Attack Model: The attacker controls m malicious clients. We assume the server stays honest and secure throughout the entire process. The attacker has knowledge of the local training data and model updates on the malicious clients. Each iteration all the honest clients will calculate and send their honest model updates while the malicious clients will send poisoned updates to the server.

Defense Goal: The goal is to improve the performance of the method FLDetector[8] by reducing the False Negative Rate (FNR) and prevent it from misclassifying honest clients and malicious clients while still maintaining the results for the rest of the metrics (i.e. testing accuracy, detection accuracy, etc.).

C. Methodology

In the proposed method, we can cluster the clients into more than two groups. We cluster them into an optimal number of clusters that Gap statistics allows us to. We use Gap statistics to find the optimal number of clusters. That way the intra-cluster distance is minimized while the distance between clusters is maximized. Specifically, each cluster will be smaller and more congested so that every point in the cluster is closer and very similar to other points in the cluster. Once the clients are clustered, we classify the cluster with the lowest malicious average as honest and only accept/aggregate model-updates from those clients. The rest of the clusters may be classified as malicious and removed from the setting. Algorithm 3 shows the pseudo codes for the proposed algorithm.

D. Experimental Setup

We will be using the same datasets and models as used in the original paper so that we may compare the results. We will be using MNIST [14], CIFAR-10 [15], and FEMNIST [16] datasets. For MNIST and CIFAR10 we will implement 100 clients and set the degree of non-independently and identically distributed (iid) to 0.5. We will start with 28 malicious clients and increase it to 45 to test the robustness of our method. For FEMNIST, we will implement 300 clients starting from 84 malicious clients and increase it to 135 with a non-iid of 0.5. For the models we will use a four-layer Convolutional Neural Network (CNN) [8] for MNIST and FEMNIST as used in the original paper and ResNet20 [17] for CIFAR-10. We will test four FL aggregation methods: Mean, Krum, Median, and Trimmed-Mean to compare the impact of the proposed method on each one. We will apply Scaling attack, Distributed Backdoor Attack, Label-flip [4] , Gaussian attack [4], and an

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on October 22,2024 at 13:28:00 UTC from IEEE Xplore. Restrictions apply.

Algorithm 3 Proposed Method

Require: Total training iterations iter and window size N.
Ensure: Detected honest clients or none.

for $t = 1, 2, ..., iter$ **do**
  $\hat{H}^t = L - BFGS(\Delta W_t, \Delta G_t)$.
  for $i = 1, 2, ...n$ **do**
    $\hat{g}_i^t = g_i^t - 1 + \hat{H}^t(w_t - w_(t-1))$.
  end for
  $d^t = [||\hat{g}_1^t - g_1^t||_2, ||\hat{g}_2^t - g_2^t||_2, ...||\hat{g}_n^t - g_n^t||_2]$.
  $\hat{d}^t = d^t / ||d^t||_1$
  $s_i^t = (\sum_{r=0}^{N-1} \hat{d}_i^{t-r})/N$.
  k = the optimal number of clusters k by Gap statistics
  if $k > 1$ **then**
    Perform k-means clustering based on the
    suspicious scores with k.
    return The clients in the cluster with the smallest
    average suspicious score as honest.
  end if
end for
return None.

adaptive attack [8] to observe the robustness of the proposed method against different types of attacks.

V. Expected Results

The main purpose of this proposal method is to prevent honest clients from being falsely detected as malicious and reduce the False Positive Rate (FPR) and False Negative Rate (FNR) as much as possible, preferably to 0. The method should reduce the FPR and FNR while maintaining the same results for the other metrics as before such as Detection Accuracy (DACC) and Testing Accuracy. Some of the results from implementing this method for MNIST with 45 malicious clients out of a total of 100 clients using Mean aggregation are shown below in Table 1.

TABLE I
TABLE 1. RESULTS FROM IMPLEMENTING PROPOSED METHOD USING MNIST WITH 45% OF CLIENTS BEING MALICIOUS FOR LABEL FLIP AND GAUSSIAN ATTACK, 40% FOR DISTRIBUTED BACKDOOR (DBA), AND 35% FOR SCALING ATTACK ON MNIST USING MEAN AGGREGATION WITH 0.5 DEGREE OF NON-IID AND CNN FOR THE GLOBAL MODEL. THE BEST RESULTS ARE IN BOLD.

Attack	Method	DACC	FPR	FNR
Label-Flip	FLDetector	0.8300	0.0000	0.3778
	Proposed Method	**0.9800**	**0.0000**	**0.0444**
Gaussian	FLDetector	0.9400	0.0000	0.1333
	Proposed Method	**1.0000**	**0.0000**	**0.0000**
Scaling Attack	FLDetector	1.0000	0.0000	0.0000
	Proposed Method	**1.0000**	**0.0000**	**0.0000**
Distributed Backdoor	FLDetector	1.0000	0.0000	0.0000
	Proposed Method	**1.0000**	**0.0000**	**0.0000**

The results show the proposed method is an improvement from the original one. It reduces the FNR while maintaining the FPR and improving DACC. This has led to an increase in testing accuracy for the global model since there are now more honest clients contributing to the global model while less malicious clients have slipped past detection. Figure 2 shows the gap scores under Gaussian attack. Figure 4 and 5 show the difference in clustering between the two methods under Gaussian attack while Figure 3 shows the true labels.

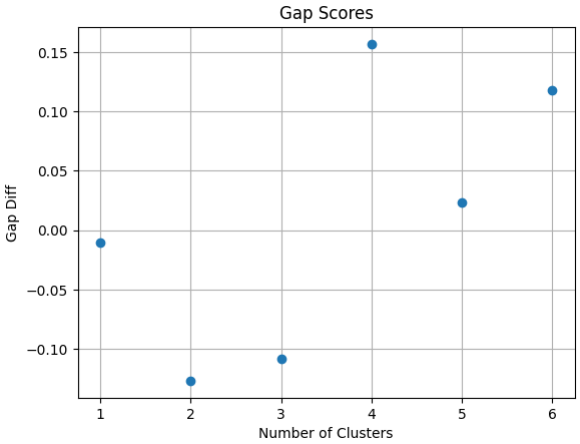

Fig. 2. Gap scores for each number of clusters. The largest Gap Difference represents the optimal number of clusters to cluster the clients.

VI. Conclusion

In this study, we design a novel clustering method to defend model poisoning attacks and detect malicious clients in Federated Learning. Our method improves FLDetector[8] and reduces its False Negative Rate (FNR). We utilize Gap statistics to determine the optimal number of clusters and then classify the cluster with the smallest average malicious score as honest and the rest as malicious. Our evaluation so far shows us promising results by reducing the False Negative Rate (FNR) and improving the Detection Accuracy (DACC). Even when FLDetector[8] correctly classifies all clients, the proposed method produces the same results.

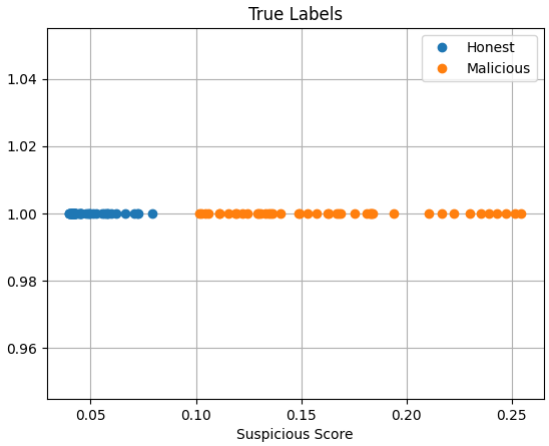

Fig. 3. True labels of clients during Gaussian attack.

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on October 22,2024 at 13:28:00 UTC from IEEE Xplore. Restrictions apply.

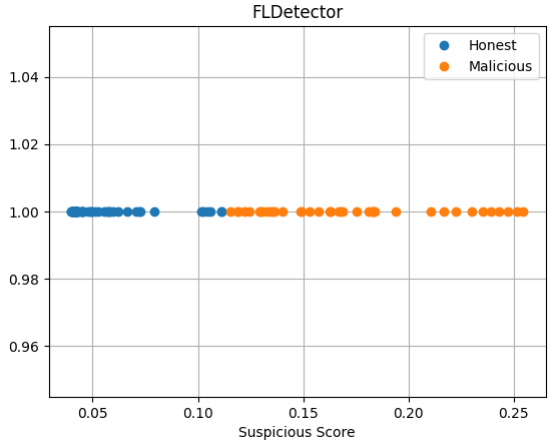

Fig. 4. Clients clustered with FLDetector.

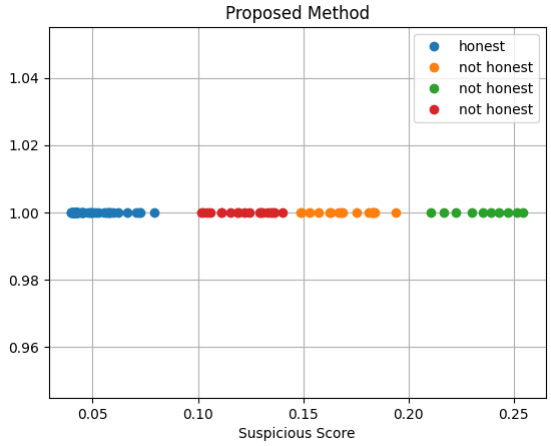

Fig. 5. Clients clustered with proposed method.

VII. Acknowledgements

We would like to express our gratitude to NSERC for their financial support, which significantly contributed to the success of this paper. Additionally, we appreciate the internal support from the University of Windsor, which played a crucial role in the completion of this research.

References

[1] Mothukuri, V., Parizi, R. M., Pouriyeh, S., Huang, Y., Dehghantanha, A., & Srivastava, G. (2021). A survey on security and privacy of federated learning. Future Generation Computer Systems, 115, 619-640.

[2] Sagar, S., Li, C. S., Loke, S. W., & Choi, J. (2023). Poisoning Attacks and Defenses in Federated Learning: A Survey. arXiv preprint arXiv:2301.05795.

[3] Pfitzner, B., Steckhan, N., & Arnrich, B. (2021). Federated learning in a medical context: A systematic literature review. ACM Transactions on Internet Technology (TOIT), 21(2), 1-31.

[4] Fang, M., Cao, X., Jia, J., & Gong, N. Z. (2020, August). Local model poisoning attacks to byzantine-robust federated learning. In Proceedings of the 29th USENIX Conference on Security Symposium (pp. 1623-1640).

[5] Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., & Shmatikov, V. (2020, June). How to backdoor federated learning. In International Conference on Artificial Intelligence and Statistics (pp. 2938-2948). PMLR.

[6] Shejwalkar, V., & Houmansadr, A. (2021, January). Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning. In NDSS.

[7] Li, X., Qu, Z., Zhao, S., Tang, B., Lu, Z., & Liu, Y. (2021). Lomar: A local defense against poisoning attack on federated learning. IEEE Transactions on Dependable and Secure Computing.

[8] Zhang, Z., Cao, X., Jia, J., & Gong, N. Z. (2022, August). FLDetector: Defending federated learning against model poisoning attacks via detecting malicious clients. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (pp. 2545-2555).

[9] Cao, X., Zhang, Z., Jia, J., & Gong, N. Z. (2022). FLCert: Provably Secure Federated Learning Against Poisoning Attacks. IEEE Transactions on Information Forensics and Security, 17, 3691-3705.

[10] Tibshirani, R., Walther, G., & Hastie, T. (2001). Estimating the number of clusters in a data set via the gap statistic. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 63(2), 411-423.

[11] Xie, C., Huang, K., Chen, P. Y., & Li, B. (2019, September). Dba: Distributed backdoor attacks against federated learning. In International conference on learning representations.

[12] Blanchard, P., El Mhamdi, E. M., Guerraoui, R., & Stainer, J. (2017). Machine learning with adversaries: Byzantine tolerant gradient descent. Advances in neural information processing systems, 30.

[13] Yin, D., Chen, Y., Kannan, R., & Bartlett, P. (2018, July). Byzantine-robust distributed learning: Towards optimal statistical rates. In International Conference on Machine Learning (pp. 5650-5659). PMLR.

[14] LeCun, Y. (1998). The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/.

[15] Krizhevsky, A., & Hinton, G. (2009). Learning multiple layers of features from tiny images.

[16] Caldas, S., Duddu, S. M. K., Wu, P., Li, T., Konečný, J., McMahan, H. B., ... & Talwalkar, A. (2018). Leaf: A benchmark for federated settings. arXiv preprint arXiv:1812.01097.

[17] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[18] Byrd, R. H., Nocedal, J., & Schnabel, R. B. (1994). Representations of quasi-Newton matrices and their use in limited memory methods. Mathematical Programming, 63(1-3), 129-156.

[19] Lang, S. (1968). A first-second course in calculus. Addison-Wesley.

[20] McMahan, B., Moore, E., Ramage, D., Hampson, S., & y Arcas, B. A. (2017, April). Communication-efficient learning of deep networks from decentralized data. In Artificial intelligence and statistics (pp. 1273-1282). PMLR.

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on October 22,2024 at 13:28:00 UTC from IEEE Xplore. Restrictions apply.