A Secure Hierarchical Federated Learning Using Dirichlet-based Trust Management

Grace Colette Tessa Masse*, Abderrahim Benslimane*, Vianney Kengne Tchendji†, and Ahmed H. Anwar‡,
*Laboratoire Informatique d'Avignon, Avignon University, Avignon, France
†Department of Mathematics and Computer Science, University of Dschang, Dschang, Cameroon
‡DEVCOM Army Research Laboratory, Adelphi, MD, USA

***Abstract*—Hierarchical Federated Learning (HFL) is a distributed machine learning training system in which a server works with several clients and edge nodes while maintaining data privacy. Distributed machine learning training systems are also known as Federated Learning, but HFL is a type of Federated Learning that utilizes a hierarchical network architecture to address computational issues when dealing with a high number of clients. However, HFL is vulnerable to attacks such as data poisoning, which may jeopardize the entire training process and result in misclassifications. As system defenders, we have to tackle this issue. Using a label-flipping attack, we investigate the effect of data poisoning attacks on HFL training. We propose a trust management-based strategy to mitigate data poisoning attacks, which assesses client trustworthiness using a Dirichlet distribution. We maintain a record of previous activities, allowing the server to enhance its knowledge based on client reliability. We demonstrate the proposed approach's effectiveness through improvements in model performance after removing malicious clients, using the MNIST dataset as a benchmark.**

***Index Terms*—Data poisoning, hierarchical federated learning, trust management model**

I. Introduction

Federated Learning (FL) is a large-scale distributed deep learning approach that enables a central server to collaborate with numerous Internet of Things (IoT) devices known as clients to train a machine learning model. This method preserves data privacy by letting clients update the global model without transmitting raw data to the server [1], in contrast to conventional Machine Learning (ML) [2], [3]. FL typically uses a single-server architecture with many clients, which may not be appropriate for particular processing needs such as bandwidth when the number of clients is large [4]. To get around FL's computational restrictions, Hierarchical Federated Learning (HFL) applies FL to a hierarchical network design. The issue of communication expenses, which significantly delayed training, is resolved by HFL [5]. Although HFL has many benefits, it shares FL's vulnerability to security risks [6]. These flaws are intended to hinder the functioning of the global FL model and take often the form of poisoning attacks [7]. Attacks that poison the model or the data are both possible. For instance, a data poisoning attack occurs when the attacker modifies the local data of the clients by injecting false or reversed information [8]. Clients with tainted data are regarded as malicious and unreliable. To protect the global model, they must be found and eliminated from the training procedure.

It can be challenging to identify malicious clients. Using a fixed decision threshold, it is decided whether or not a client parameter is correct for the good functioning of the training process [8], [9]. However, the performance of the whole model can be harmed if the threshold is set too high, though, as malicious clients may be able to avoid detection. In contrast, if the threshold is set too low, good clients can unintentionally be excluded, which would lower the model's quality. The efficiency of protection mechanisms against data poisoning attacks in HFL must therefore be improved by finding a balance between the decision threshold and dynamic client participation. Moreover, An approach that appears to hold promise for identifying malicious clients is quantifying various client behaviors in HFL by altering the decision threshold. As far as we are aware, this strategy has not yet been explored.

Our contribution is a new malicious client detection mechanism in HFL that is based on measuring distinct client behaviors by varying the decision threshold. In contrast to the literature such as Baseline [5], Multi-Krum [10], and FoolsGold [11], our method increases detection performance because we can be sure with a high probability that a client is honest before maintaining him in the training process or that a client is malicious before dropping him. Furthermore, our method is less vulnerable to the presence of malicious clients than existing solutions.

In this paper, we aim to leverage the hierarchical structure of HFL to introduce the Dirichlet trust management system to evaluate the trustworthiness of each client and assign a trust score accordingly [12]. The trust management system is based on Dirichlet distribution which is a Bayesian statistical method that estimates the likely future behavior of a client based on his past history [13], [14]. This enables the direct expression of incremental evaluations and their reflection on the generated confidence scores. By merging the prior trust score with the fresh evaluations, the Dirichlet trust management system determines the updated posterior trust score. We use it to propose a new scheme for detecting malicious clients in HFL in a hierarchical network architecture consisting of a server, edge node, and clients. Our system assumes that the server and edge node are trustworthy, but not the clients. We assign trust to a client depending on its behavior, reflected by its

979-8-3503-7099-7/24/$31.00 ©2024 IEEE

level of credibility. Specifically, each edge node evaluates credibility of its corresponding clients to detect the prese or absence of poisoning, by analyzing the performance of model parameters received from its clients. In HFL, client ticipation is dynamic, meaning that a client may switch e nodes from one iteration to another, giving the edge noc partial view of its client's behavior history. Thus, the edge n cannot decide to revoke a client. Therefore, we propose that server should handle this task by aggregating the behavio clients across different edge nodes, removing malicious clie and proposing updates to the decision threshold to edge nc that have more malicious clients.

The remainder of the paper is organized as follows. Sec II presents the system model and the adversary's assumpt In section III, we present the approach we propose. Numer results are presented in section IV, and conclusions are dr in section V.

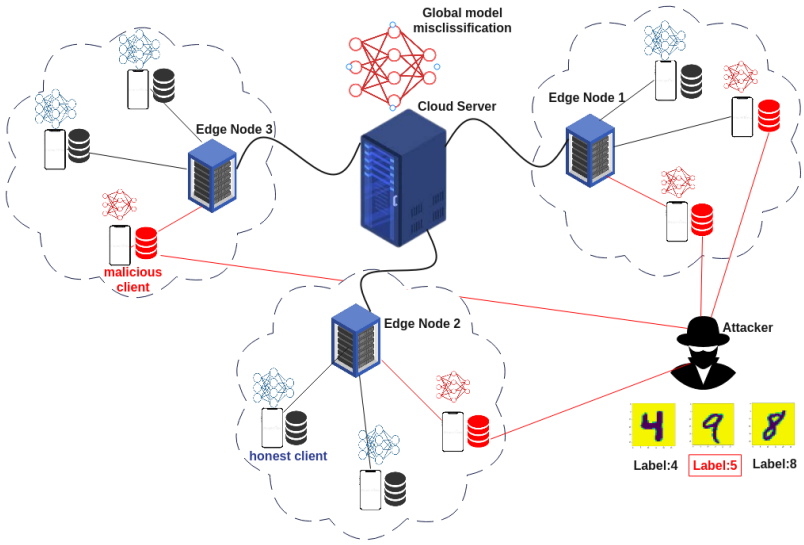

Fig. 1. Data poisoning attack in Hierarchical Federated Learning

II. System Model and Adversary Assumption

Our model relies on a Server-edge-Client architecture, where each edge node assesses the credibility of its clients based on the history of their behavior (malicious or not); adversarial activity results in revocation from the learning process. This system model uses the hierAVG algorithm [5], [15], which consists of clients (with learning models) reporting model parameters to their corresponding edge nodes, who in turn submit their model parameters to the server.

A. Problem definition

In data poisoning attacks, the data used to train models is tainted so that the trained model's outputs correspond with the attacker's predetermined targets [16]. We consider the label flipping assault, one of numerous data poisoning techniques because it has negative effects on HFL [9]. By changing the training data's actual label to a different label, a label-flipping attack is carried out. Gradients are then computed using this "poisoned" data. By taking advantage of the learning method, such assaults can dramatically reduce the performance of the classifier [17]. For instance, a hacker may pass off legitimate customer reviews as fake, spam emails as secure, and pneumonic X-ray results as expected [8].

As described in Figure 1 illustrating the HFL training process with poisoning, each client generates its local data through its activity and uses it to train the model locally. However, the attacker alters the data of some clients. At the end of each training phase, clients send their model weight updates (parameters) to the corresponding edge node for aggregation and model update at the edge node level. Unfortunately, poisoned clients have also sent their weight updates, resulting in a poisoned model on the edge node. Then, the edge nodes send their model weight updates to the server for final aggregation and update of the global model, which makes erroneous classifications as it is based on poisoned model weights coming from the edge nodes. The updated new global model is then sent back to all clients, who continue with the next training phase until the local model accuracy can no longer be improved.

B. Adversary's assumption

In this work, it is assumed that the proportion of adversaries' clients to all other clients is never greater than fifty percent. Up to 40% of our clients were thought to be attackers. We also suppose that a client can change his corresponding edge node and behavior from one iteration to the next due to the client's dynamic participation. We take into account two cases. In the first scenario, the model is constantly under attack from the rogue client. That indicates that such a client updates the global model with tainted data at all times. In the second case, we presume that the malicious client will target the model roughly fifty percent of the time. In other words, from such a client, 50% of the updates are poisoned and 50% are not.

III. Proposed Approach

The poisoning of the global model is done through malicious clients who send incorrect updates, hence the need to detect and revoke them to secure the global model. In this section, we propose a robust trust management model for detecting and revoking malicious clients during training. This model adopts a Bayesian method to assess the trustworthiness of clients at each edge node and assist the server in identifying those who are malicious. Specifically, We propose that the Server calculate the likely future behaviors of each client based on their prior actions by utilizing the Dirichlet family of probability density functions. Clients' prior actions are computed based on the credibility judgment report of the related edge nodes. This technique enables the server to identify malicious clients so that they can be revoked.

A. Credibility Rating

In our model, the server initializes the global model and selects edge nodes to collaborate with. We assume that edge

nodes are initially trustworthy. The server also selects the clients to participate in training and distributes them to each edge node.

In the beginning, the server broadcasts the global model to the edge nodes, who in turn disseminate it to their clients. After local training, each client sends its updates to its corresponding edge node. The edge node evaluates the update's credibility before making a judgment about the client's behavior. Highly Trustworthy, Trustworthy, Untrustworthy, and Highly Untrustworthy are the four levels of credibility that are correlated with behavior judgment. Following the updates evaluation, the edge node transmits to the server a report describing its verdict of the credibility of each of its clients. For every client, the server builds an observation vector. The judgment history of every client is contained in this vector. The server then computes the Dirichlet probability based on the observation vector to determine which client to revoke.

Note that the level of credibility is established by the observable impact of the updates on the global model performance at the edge node level [8]. Good updates improve the global model while bad ones deteriorate it. There can be different performance metrics to observe updates' impact (accuracy, loss function, F1 score...). We choose to consider the global model accuracy. We assume that each client update after aggregation creates a deviation between the edge node accuracy before aggregation and the edge node accuracy after aggregation and we define k possible levels of deviation. We also define the function h that maps the levels of deviation to an interval $[0, 1]$ like $\forall i \in [1, k],\ h(i) \in [0, 1]$; where 0 denotes a benign deviation and 1 is a very bad deviation. In other words, h strictly preserves the order relation between deviations such that if deviation i is worse than j (meaning that if $i < j$), then $h(i) > h(j)$ as illustrated in Figure 2.

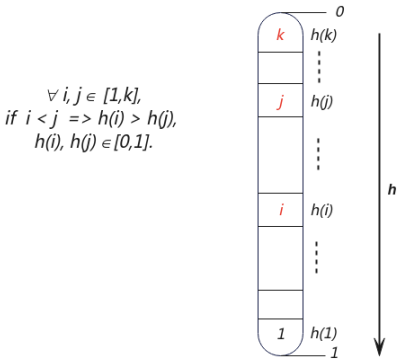

Fig. 2. k-levels of deviation mapping

Remember that client's update must satisfy the edge node evaluation in order to be credible. We define three criteria to assess the edge node's level of satisfaction:

- the current accuracy $a \in [0, 1]$ which represents the edge node accuracy before the client's update aggregation ;
- the calculated accuracy $o \in [0, 1]$ which represents the edge node accuracy after aggregation;
- the edge node's decision threshold $\rho \in [0, 1]$, which characterizes the value at which the edge node decides that an update is malicious at the time of observation.

This decision threshold is defined in our approach as the degree of tolerance of the edge node for taking decisions and making judgments. The decision threshold also impacts the judgment rate. If it is set too high, honest clients may be revoked, and if it is set too low, a malicious client may be kept in the training process. However, a fixed decision threshold does not adapt to the client behavior variation over edge nodes and communications. That is the reason why we chose to update his value after the reporting phase.

To quantitatively measure the credibility of an update, we defined a function $cred(a, o, \rho) \in [0, 1]$ to represent the level of satisfaction of an update based on the current accuracy, the distance from the calculated accuracy, and the edge node's decision threshold. Formally, this function is written as:

$$cred(a, o, \rho) = \begin{cases} 1 - \left(\frac{a-o}{max(C_1 o, 1-o)}\right)^{\frac{\rho}{C_2}} & if \quad a > o \\ 1 - \left(\frac{C_1(o-a)}{max(C_1 o, 1-o)}\right)^{\frac{\rho}{C_2}} & if \quad a \leq o \end{cases} \tag{1}$$

C_1 represents the cost of penalties for incorrect estimations. It is greater than 1 to reflect the fact that updates that deteriorate the model are more strongly penalized than those that improve it. C_2 reflects the sensitivity of credibility. The higher its value, the greater the sensitivity to the difference between the current accuracy and the calculated accuracy. Additionally, this equation ensures that during observations of updates when the edge node's decision threshold is low, they are more severely penalized in case of bad updates.

B. Dirichlet-based Model

In Bayesian statistics, we have a theoretical basis that allows us to measure uncertainty in a decision based on a collection of observations. We are interested in the distribution of the different levels of credibility of the updates from each client. Specifically, we want to take advantage of the information provided by this distribution to estimate the level of credibility that can be attributed to each client during future communications. In the case of binary credibility satisfactory, unsatisfactory, a Beta distribution can be used, as indicated in [7]. For credibility levels with multiple values, Dirichlet distributions are more appropriate.

A Dirichlet distribution [8] is based on initial beliefs about an unknown event, represented by a prior distribution. The initial beliefs combined with collected information can be represented by a posterior (or updated) distribution. The posterior distribution is well-suited to our trust management model because trust is updated based on the history of "client-edge node" interactions.

Let Θ be a discrete random variable denoting the level of credibility of an update. For k possible levels, Θ can take a value in the set $\mathcal{O} = \{\theta_1, \theta_2, \ldots, \theta_k\}$ corresponding to the level of satisfaction of the edge node for that update. Let $\vec{p} =$

$\Big(p(\theta_i)\Big)_{1\leq i\leq k}$ be the probability distribution vector of Θ, and $\vec{\alpha} = \big(\alpha(\theta_i) = \alpha_i\big)_{1\leq i\leq k}$ the vector of cumulative observations and prior beliefs about Θ. Then, we can model $\vec{p}$ using a posterior Dirichlet distribution as follows:

$$f(\vec{p}|\vec{\alpha}) = Dir(\vec{p}|\vec{\alpha}) = \frac{\Gamma(\sum_{i=1}^{k}\alpha(\theta_i))}{\prod_{i=1}^{k}\Gamma(\alpha(\theta_i))}\prod_{i=1}^{k}p(\theta_i)^{\alpha(\theta_i)-1}. \quad (2)$$

where

$$\begin{cases} p(\theta_1),...,p(\theta_k) \geq 0; \\ \sum_{i=1}^{k} p(\theta_i) = 1; \\ \alpha(\theta_1),...,\alpha(\theta_k) > 0. \end{cases} \quad (3)$$

The expectation probability value of the Θ random variable is defined as follows:

$$E(p(\theta_i)|\vec{\alpha}) = \frac{\alpha(\theta_i)}{\sum_{i=1}^{k}\alpha(\theta_i).} \quad (4)$$

C. Trustworthiness evaluation of client

After receiving and evaluating an update, an edge node assigns a credibility value to this update in accordance with equation 1. This credibility value is assigned to one of the satisfaction levels in the set $\mathcal{O} = \theta_1, \theta_2, ...\theta_k$ that has the closest value. Each satisfaction level θ_i also has a weight w_i. Let $\vec{p}^{\mu\varepsilon}$ be the probability that client μ provides updates to edge node ε with satisfaction level i, such that:

$\vec{p}^{\mu\varepsilon} = (p^{\mu\varepsilon_i})_{i=1,...,k} | \sum_{i=1}^{n_k} p^{\mu\varepsilon_i} = 1)$.

To express the trust score as a single value, we assign a weight value w_i to each rating level i, which is calculated as: $w_i = \frac{i-1}{k-1}$ or $w_i = \frac{i+1}{k+1}$.

We model $\vec{p}^{\mu\varepsilon}$ using equation 2. Let $\Theta^{\mu\varepsilon}$ be the random variable that represents the weighted average of the probability of each satisfaction level in $\vec{p}^{\mu\varepsilon}$.

$$\Theta^{\mu\varepsilon} = \sum_{i=1}^{k} p^{\mu\varepsilon} w_i \quad (5)$$

After receiving reports from the edge nodes, the server aggregates them and calculates the trust score of the client as:

$$\mathcal{T}^{\mu\varepsilon} = E(\Theta^{\mu\varepsilon}) = \sum_{i=1}^{k} E(\Theta_i^{\mu\varepsilon}) w_i = \frac{\sum_{i=1}^{k} w_i \alpha(\theta_i)^{\mu\varepsilon}}{\sum_{i=1}^{k}\alpha(\theta_i)^{\mu\varepsilon}} \quad (6)$$

where $\alpha(\theta_i)^{\mu\varepsilon}$ is the cumulative evidence that μ has sent an update to ε with satisfaction level θ_i. Clients with a low trust score are revoked, and edge nodes with the most poisoned clients are advised to improve their decision threshold by one step d.

IV. Performance evaluation and discussion

The details of the performance evaluation to support our analysis and proposed approach for minimizing poisoning attacks are presented in the following sections. We specifically offer details on the dataset used, the classification model architecture, and the numerical results.

A. Dataset

In this work, we utilize the MNIST dataset [18], which is extensively utilized for classification tasks. It contains 60,000 training samples and 10,000 test samples with ten-digit classes and is made up of handwritten English digits (0-9). The data is in CSV format, and 60% of the training examples are poisoned or have inverted labels for experimentation, while the remaining 40% are not poisoned. The dataset's original labels are inverted at random to yield poisoned labels. The dataset does not initially conform to the conventional data distribution, but we transform it by subtracting each value from the mean and dividing it by the standard deviation. To make it easier to detect malicious clients, we partition the dataset equally among clients as data shards in an i.i.d way. We present the numerical results and classification model architecture in the following subsections.

B. Learning Model and Experimental Setup

The classification model in our work is a 3-layer MLP with an input layer, a hidden layer with 200 neurons, and an output layer with 10 prediction probabilities for the digits. We use the Convolutional Neural Network (CNN) with 21840 trainable parameters as in [2]. After shuffling, we distribute the dataset among 50 clients and we share equally the 50 clients among 5 edge nodes. To simulate an assault using the label-flipping approach, we distribute batches of data with label-flipped samples to 20 clients (40% of the 50 clients), while the remaining 30 clients receive batches of data with the original label. As a result, there are 20 poisoned clients and 30 honest clients. Our approach experiment's purpose is to identify those poisoned clients.

At first, each edge node has a copy of the designed categorization model. The data from the local clients are then converted into TensorFlow dataset objects and batch-processed. At each communication round, the server broadcasts its global model to edge nodes which sends it to the participating clients to begin the training process. The training settings are as follows: $C = 50$, $B = 16$, $e = 10$, $lr = 0.01$, $d = 0.1$ and $lr - decay = 0.995$ for the number of communication rounds (R), batch size (B), number of epochs for each local client (e), decision threshold(d), learning rate (lr) and learning rate decay ($lr - decay$). The server collects clients' behavior reports from each edge node throughout 100 communication rounds. After updating its parameters from each client, the edge node determines the loss and accuracy of the global model at each communication round, using categorical cross-entropy as the loss function. The report collector then uses the collective report from each client to calculate the trust score model.

C. Results and Discussions

This section describes our approach's outcomes and how it effectively eliminates malicious clients with poisoned data. We provide a full explanation of our methodology's results.

Figures 3 and 4, which show average client update outcomes, indicate the global model's accuracy after averaging

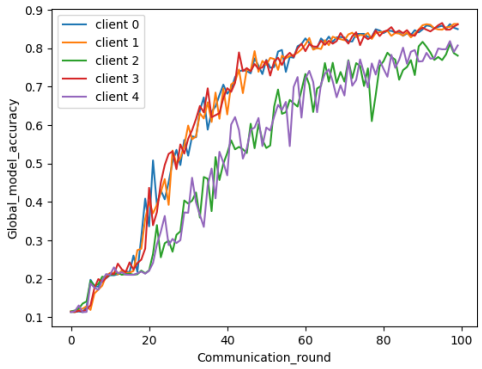

Fig. 3. Accuracy on the global model by local client updates. 5 data held by poisoned clients are label-flipped datasets

clients' updates on each communication round. The number of polluted data points is the only distinction between the two statistics. With regard to Figure 3, it illustrates accuracy when 50% of the data points on two clients (clients 1 and 3) are poisoned, while Figure 4 demonstrates the outcome when 100% of the data points are poisoned.

Our observations show that the poisoned clients are close to acting similarly to the non-poisoned clients from the 100th communication cycle in the first attack scenario (Figure 3). This similarity, however, is not present in the second attack scenario (Fig. 4). In the first scenario, only a fifth of the data pieces are poisoned, but clients rapidly begin acting honorably, making defense more challenging. For the remainder of the analysis, we concentrate on the first attack scenario.

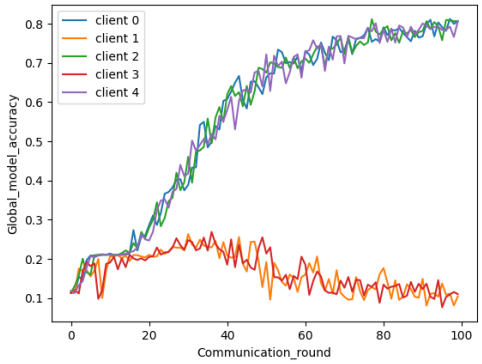

Fig. 4. Accuracy on the global model by local client updates. All of the data held by poisoned clients are label-flipped datasets

There are 50 clients; 20 are poisoned, leaving 30 honest clients. While edge node only has a partial view, the server oversees each client's learning process. Equation 6 is used to determine a trust score for each of the 50 clients by taking into account the edge node credibility judgment that the server stores for each client as a report vector. Figure 5 shows the clients' trust scores, with some clients having favorable ratings and others having negative ratings. Clients whose trust score is lower than 0 are revoked by the server.

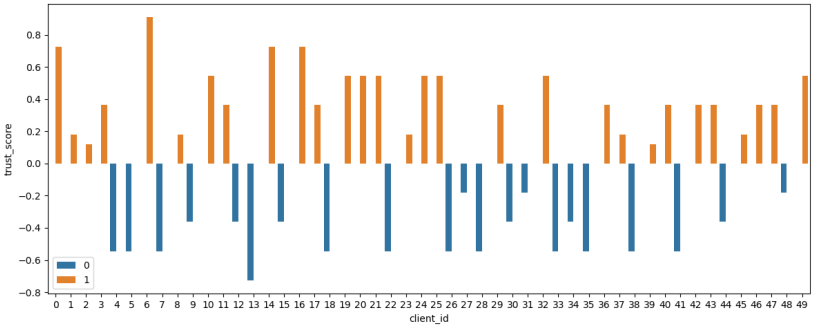

Fig. 5. Trust score of 50 Clients

Table I illustrates changes in the edge node judgment rate and the fluctuation of threshold decision values. It can be observed that with a threshold decision of 0, the attack detection rate is 1.05, indicating that the system has identified and eliminated 21 clients as malicious (as shown in Figure 5), with only one honest client wrongly identified as an attacker. This is considered more acceptable than undetected attacking clients. On the other hand, choosing a threshold score of -0.7 would result in the detection of fewer attacking clients, with an attack detection rate of 0.80. Therefore, rejecting all clients with a trust score lower than 0 is a reasonable decision.

Judgement Ratio	0.80	0.92	1.05	1.05	1.05	1.12	1.20
Decision threshold	-0.7	-0.5	-0.3	0	0.1	0.3	0.5

TABLE I
EDGE NODE JUDGEMENT RATIO WITH THE VARIATION OF THE DECISION THRESHOLD

The HFL server revokes all clients whose trust score is below 0. In our experiment, 21 clients are identified as attackers, so the server removes them from the training process. After removing the identified clients, we evaluate the performance of the MNIST classification model with the remaining clients. The model's accuracy improves significantly when the server removes the reported model parameters from malicious clients, as shown in Figure 6. Note that accuracy refers to the accuracy of the global model updated after averaging the weights of each client at each communication round. We observed that after removing the attacker clients, the classification model's accuracy increased from 80.2% to 92,8% over 100 communication rounds, as shown in Figure 6

Therefore, by using our proposed trust management-based approach, we can detect data poisoning attacks and improve the classification model accuracy by around 12.6%, as shown in Figure 6. We show the comparative analysis of our proposed approach with baseline simple averaging [5], HFL with Multi-Krum aggregation [10], and FoolsGold [11] in Table II. Multi-Krum aggregation can tolerate the presence of up to 31% attackers and fails above this threshold. On the contrary, our

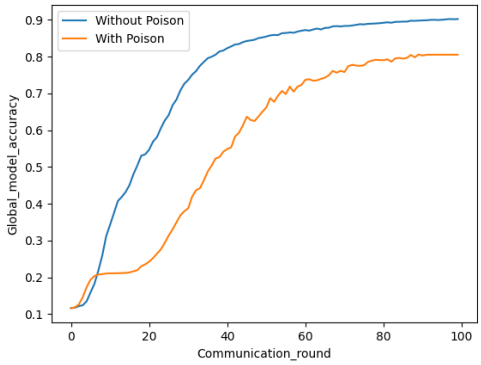

Fig. 6. Accuracy of the global model with attacker vs without attacker

Poison clients(%)	Baseline	MultiKrum	FoolsGold	Our approach
0%	0	0	0	0
13%	0.34	0.4	0.17	0.23
31%	0.93	0.98	0	0
40%	0.95	0.97	0	0

TABLE II
THE RATE OF ATTACK WAS MEASURED FOR DIFFERENT APPROACHES AS THE PERCENTAGE OF ATTACKERS VARIED

approach is defensive even with 40% attackers. Similarly, the attack rate increases significantly with the growth of the attacker percentages on both Baseline and Multi-Krum approaches. Our approach outperforms these two approaches however FoolsGold shows good performance at 13% of attackers. Our future work will look at the performance of attacker percentages over 40% and evaluate other benchmark datasets.

V. CONCLUSION

Our method identifies malicious clients and excludes them from the training process by applying a Dirichlet trust management and score elimination. This strategy assures that only outstanding models are utilized for classification algorithms and contributes to protecting clients' private information against poisoning attacks. Last but not least, it should be mentioned that our Dirichlet trust-based management methodology can be used in various HFL scenarios where malevolent clients may jeopardize the training process using a data poisoning attack. In order to increase the precision of fraudulent client detection, our approach can be expanded to take into account additional elements such as clients' histories of involvement level.

ACKNOWLEDGMENT

Research was sponsored by the Army Research Office and was accomplished under Grant Number W911NF-21-1-0326. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Office or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

REFERENCES

[1] Zhilin Wang, Qiao Kang, Xinyi Zhang, and Qin Hu. Defense strategies toward model poisoning attacks in federated learning: A survey. *arXiv preprint arXiv:2202.06414*, 2022.

[2] Shuiguang Deng, Hailiang Zhao, Weijia Fang, Jianwei Yin, Schahram Dustdar, and Albert Y Zomaya. Edge intelligence: The confluence of edge computing and artificial intelligence. *IEEE Internet of Things Journal*, 7(8):7457–7469, 2020.

[3] Seraphin B Calo, Maroun Touna, Dinesh C Verma, and Alan Cullen. Edge computing architecture for applying ai to iot. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 3012–3016. IEEE, 2017.

[4] Deepti Gupta, Olumide Kayode, Smriti Bhatt, Maanak Gupta, and Ali Saman Tosun. Hierarchical federated learning based anomaly detection using digital twins for smart healthcare. In *2021 IEEE 7th International Conference on Collaboration and Internet Computing (CIC)*, pages 16–25. IEEE, 2021.

[5] Lumin Liu, Jun Zhang, SH Song, and Khaled B Letaief. Client-edge-cloud hierarchical federated learning. In *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2020.

[6] Jingwei Sun, Ang Li, Louis DiValentin, Amin Hassanzadeh, Yiran Chen, and Hai Li. Fl-wbc: Enhancing robustness against model poisoning attacks in federated learning from a client perspective. *Advances in Neural Information Processing Systems*, 34:12613–12624, 2021.

[7] Krishna Yadav and BB Gupta. Clustering based rewarding algorithm to detect adversaries in federated machine learning based iot environment. In *2021 IEEE International Conference on Consumer Electronics (ICCE)*, pages 1–6. IEEE, 2021.

[8] Aashma Uprety and Danda B Rawat. Mitigating poisoning attack in federated learning. In *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 01–07. IEEE, 2021.

[9] Ronald Doku and Danda B Rawat. Mitigating data poisoning attacks on a federated learning-edge computing network. In *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–6. IEEE, 2021.

[10] Junyu Shi, Wei Wan, Shengshan Hu, Jianrong Lu, and Leo Yu Zhang. Challenges and approaches for mitigating byzantine attacks in federated learning. In *2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 139–146. IEEE, 2022.

[11] Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. The limitations of federated learning in sybil settings. In *RAID*, pages 301–316, 2020.

[12] Weidong Fang, Wuxiong Zhang, Lianhai Shan, Xiaohong Ji, and Guoqing Jia. Ddtms: Dirichlet-distribution-based trust management scheme in internet of things. *Electronics*, 8(7):744, 2019.

[13] Audun Josang and Jochen Haller. Dirichlet reputation systems. In *The Second International Conference on Availability, Reliability and Security (ARES'07)*, pages 112–119. IEEE, 2007.

[14] Yingkun Wen, Yan Huo, Tao Jing, and Qinghe Gao. A reputation framework with multiple-threshold energy detection in wireless cooperative systems. In *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2020.

[15] Liu Wentao, Xu Xiaolong, Li Dejuan, Qi Lianyong, Dai Fei, Dou Wanchun, and Ni Qiang. Privacy preservation for federated learning with robust aggregation in edge computing. *IEEE INTERNET OF THINGS JOURNAL*, 2022.

[16] Gilad Baruch, Moran Baruch, and Yoav Goldberg. A little is enough: Circumventing defenses for distributed learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[17] Prajjwal Gupta, Krishna Yadav, BB Gupta, Mamoun Alazab, and Thippa Reddy Gadekallu. A novel data poisoning attack in federated learning based on inverted loss function. *Computers & Security*, page 103270, 2023.

[18] Keyang Cheng, Rabia Tahir, Lubamba Kasangu Eric, and Maozhen Li. An analysis of generative adversarial networks and variants for image synthesis on mnist dataset. *Multimedia Tools and Applications*, 79:13725–13752, 2020.