Breakwater: Securing Federated Learning from Malicious Model Poisoning via Self-Debiasing

Yeawon You*, JinYi Yoon†, and HyungJune Lee*
*Department of Computer Science and Engineering, Ewha Womans University, Seoul, South Korea
†Department of Computer Science, Virginia Tech, Blacksburg, VA, USA
yeawon@ewhain.net, jinyiyoon@vt.edu, hyungjune.lee@ewha.ac.kr

***Abstract*— Deep learning models deployed on edge devices leverage locally collected data to extract intelligence, mitigating privacy concerns associated with external data sharing. Edge federated learning, an on-device learning paradigm, has emerged as a promising solution, allowing edge nodes to train models locally and share only the trained weights, preserving data privacy. However, it also poses critical challenges of network burden and potential model poisoning. We introduce a self-debiasing security framework *Breakwater* for multi-hop edge federated learning. We incorporate on-device malicious weight discriminator at each participant, enhancing security and robustness of the federated learning process. The framework strategically balances the benefits of participating nodes with timely defenses against potential malicious clients. Based on the discriminator, we further embed a self-debiasing mechanism that can determine whether to retain or discard the weight propagation from its child nodes. Our *Breakwater* framework identifies and filters out harmful weights, ensuring the integrity of the global model. Our work contributes to the ongoing discourse on federated learning security, presenting a solution that maintains efficiency while robustly defending against model poisoning threats. We demonstrate its efficacy in enhancing the reliability of the multi-hop edge federated learning process with recovery of up to 69 % in accuracy under attack, offering a path toward secure and cooperative distributed learning environments.**

I. Introduction

In the era of intelligent services, as deep learning heavily relies on the quantity and quality of data, a key source of intelligence has transitioned to the edge, which tends to be equipped with various sensors near users in real-world environments. However, it often suffers from privacy leakage. To tackle the problem, secured federated learning on edge devices emerges as a key research area with attention, enabling knowledge extraction from abundant data at edge nodes without privacy or resource concerns.

In typical federated learning, each edge node trains a model with its own local data and then sends the trained local model to a central server to aggregate the models and form a global model. However, the model aggregation might potentially impose an excessive burden on the central server and incur a bottleneck on the local network to the central server [10]. Similar to in-network aggregation [6], which brings communication and computational efficiency to edge sensor nodes with limited resources, the combination of multi-hop federated learning and the in-network aggregation paradigm [4] can make federated learning more efficient for edge nodes. To easily apply into the existing networks, a tree structure is generally applied to multi-hop federated learning [4]. From the leaf node, each participant successively sends the local weight to its parent node, and then the parent node aggregates the model from its child nodes up to the root node, which finally constructs a global model.

Federated learning with multiple edge participants has raised critical security threats such as model poisoning attacks. Each node crucially contributes to the whole learning process, offering new model updates from local data. On the contrary, the reliance on distributed nodes is adversely vulnerable to possible threats. When aggregating the models in multi-hop federated learning, an attacker can pretend to be a normal participant or intercept the benign model parameters in transit and instead send some manipulated models to the aggregation server to distort the global model. One of the prevalent model poisoning methods is to simply multiply the model parameters by an arbitrary scaling factor or add noise [1], and the global model becomes disrupted, incurring critical training failures. The presence of even a single or very few malicious participants holds the serious potential to jeopardize the entire framework.

Various defense mechanisms against model poisoning in model aggregation have been proposed by employing some aggregation or filtering rules. Traditionally, a central server that maintains a global model tends to run a geometric median aggregation to mitigate the impact of some outlier weights that can potentially be malicious [5]. Some research works predefine a maximum number of the Byzantine expected weights and discard the weights with significant out-of-distribution [2], [13], [14]. However, these statistical approaches require collecting the model weights of participants to a single node to operate optimally. It becomes particularly crucial in multi-hop federated learning, where the numerous model parameters from multiple participants should be transmitted across multiple hops, incurring excessive network and storage overhead for resource-constrained edge devices. Furthermore, in these

This work was supported by the Institute of Information & communications Technology Planning and Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2022-00155966, Artificial Intelligence Convergence Innovation Human Resources Development (Ewha Womans University) and No.2021-0-02068, Artificial Intelligence Innovation Hub). The corresponding author is HyungJune Lee.

approaches, a predefined number of weights also need to be discarded, exposing a new attack surface for attacks similar to [1]. Here, there is a need for a lightweight defense mechanism capable of distributed manner on devices.

Now, we raise an interesting question – *Is there a way to discriminate whether the new model update from a node indeed contributes or deteriorates for all? Then, how to self-debiase the intermediate model updates at each participant level for agile defense?* Different from typical federated learning, we aim to design a defense system in multi-hop federated learning by gradually performing malicious inspection along the aggregation path. Diagnosing the maliciousness of the child nodes recursively and dynamically branching out a sub-tree with malicious nodes can contribute to secure and effective federated learning.

In this work, we introduce *Breakwater*, a novel model poisoning attack-proof defense framework to gradually secure a global model in multi-hop federated learning. We embed an on-device malicious weight discriminator at each edge device, while reducing centralized overhead at the master node side. Each node judges the level of maliciousness of its child nodes and filters out poisonous weights that pose a threat to the integrity of the global model.

We implement a two-phase mechanism: 1) Anomaly Discriminator with neural networks; and 2) Self-debiasing Defense to maintain an attack-free participant network. It significantly enhances robustness and security in multi-hop edge federated learning, ensuring the dependability of the aggregated model despite the potential presence of malicious clients within the network. This decision-making process becomes pivotal between weighing the choice to retain a set of nodes for potential contribution within the framework or rather aggressively discarding suspicious nodes to safeguard the integrity of the global model. Our work finds an interesting trade-off between usefulness of new model update and harmfulness of possible model poisoning.

II. System Architecture

We consider the problem of anomaly detection and self-debiasing defense against model poisoning in multi-hop federated learning. Typical federated learning involves direct communication between a central server and local devices, whereas multi-hop federated learning requires multiple hops of communication between devices. Since the direct communication spans with one hop neighbors, it can potentially offer some advantages in terms of privacy, fault tolerance, and scalability. In general, a tree structure is a practical way to adapt to its underlying network topology and organize communication in an efficient multi-hop manner. In particular, we consider a complete binary tree structure to avoid duplicate weight reflection or traversal with in-network aggregation in the federated learning scenario.

In the tree-based multi-hop federated learning, each node initiates training with its own local data. To aggregate the respective trained model parameters (e.g., weights or biases), child nodes send their local models to their parent node, and then it merges the models from itself and from its two child nodes. This model aggregation is processed recursively from the leaf node to ensure that the intermediate node receives the aggregated models from its child nodes. The gradually accumulated model is propagated up to the root node, which is a master node who finally constructs a global model. The global model is disseminated to all other nodes down to the tree structure. Upon disseminating the global model down to all of nodes, each node restarts local training process, and this whole procedure is repeated until the global model converges.

A. Threat Model

An attack-proof federated learning system aims to diagnose and prevent the learning process from potential attackers. The attacker nodes participate in federated learning and try to make the global model collapse. On top of the multi-hop federated learning approach, it is assumed that malicious participants may employ an attack strategy to send incorrect or manipulated model parameters with the intent to *poison* the global model or give unauthorized insights into other participants.

This model poisoning attack is to intentionally inject some poisoned model parameters to be aggregated into the global model of federated learning. The presence of even a few malicious nodes may disrupt the global model, consequently contaminating all the other nodes within the whole system. The attackers can be involved in the learning process with or even without their own data. In particular, it is assumed that the attackers may multiply a constant value to their own trained models and propagate the manipulated models to connected nodes [3]. Through successive attacks, they can control the global model to be escalated or diminished. These contaminated parameters mislead final learning process and hinder a learning system from effectively extracting the features from local data at the client side.

B. Self-Debiasing against Model Poisoning Attacks

Under the model poisoning attacks in multi-hop federated learning, our goal is to diagnose and branch out possible malicious nodes to prevent a global model from being poisoned. In the best effort manner, each participant tries to detect whether its child nodes are malicious from the bottom layer. If a certain node is revealed to be highly likely malicious, it needs to be filtered out before aggregating the model. By recursively defending at each tree level, participants are repeatedly inspected, maintaining the global model more secured.

Given a tree-based network topology of participants, it is assumed that the network connection among nodes is stable. We assume that all of participants train a homogeneous model architecture to be simply aggregated by averaging the parameters based on *FedAvg* [9]. As illustrated in Fig. 1, we consider a dynamic self-debiasing defense mechanism in which a participant discards possible biases in model parameters by running malicious discriminator at each level and sends the sanitized model aggregate toward the central server.

III. Defense System with Breakwater

In a multi-hop federated learning framework with in-network aggregation, a participating node located at a certain

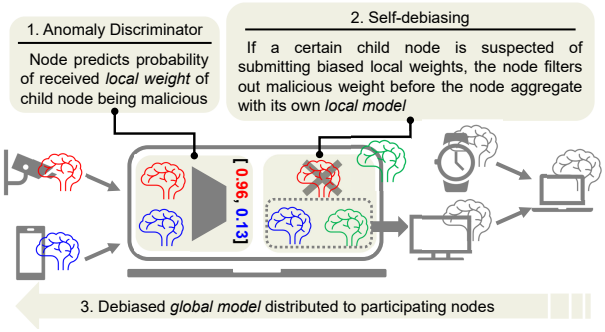


Fig. 1. Our self-debiasing defense architecture of *Breakwater*

tree level aggregates a model with the weights from itself and its child nodes. The intermediate model, denoted as $\bar{W}_t^i$, represents the weight aggregated using federated averaging such as *FedAvg* [9] at a node with depth i at epoch t as follows:

$$\bar{W}_t^i = FedAvg(W_t^i, W_t^{i+1(L)}, W_t^{i+1(R)}), \tag{1}$$

where $W_t^{i+1(L)}$ or $W_t^{i+1(R)}$ is the weight aggregate at the left or right child node with depth $i+1$. The final global model is denoted as $\bar{W}_t^1$, indicating the global weight aggregated at the root node with depth $i=1$ at epoch t. After the global model is calculated and is disseminated to all of the participating nodes, a local node calculates Δ_{t+1}^i from its local data, which is the gradient of the $t+1$ epoch from the node with depth i, as follows:

$$W_{t+1}^i \leftarrow \bar{W}_t^1 - \eta \cdot \Delta_{t+1}^i, \tag{2}$$

where $\bar{W}_t^1$ is the global model of previous epoch t.

We construct a layer-wise security framework called *Breakwater* at each node level by incorporating a malicious node discriminator and self-debiasing defense mechanism in federated learning. The discriminator at an intermediate node evaluates the potential maliciousness of the weights received from its left and right child nodes, $W_t^{i+1(L)}$ and $W_t^{i+1(R)}$, before aggregating them into $\bar{W}_t^i$ along with its own weight W_t^i. The discriminator outputs the probability of each child node to be malicious, providing valuable information for the parent node to perform self-debiasing and decide whether to aggregate or filter out the weight from a particular child node.

The defense policy within our system is contingent on the risk tolerance of the global model, with a focus on balancing fast convergence and acquiring diverse data from various nodes. Parent nodes play a pivotal role in running the defense policy. When the policy is too strict, even child nodes with a relatively low probability of being malicious can be discarded. The strategic approach ensures that the federated learning system can adapt its defense mechanisms based on the desired trade-off between convergence speed and security.

A. Anomaly Discriminator

For the discriminator, the data used for discrimination diverges from existing methods as it lacks access to a diverse range of weights trained across various nodes with local data. Instead, a node exclusively gathers the weights directly from its child nodes. While the limited input data pool poses a challenge for reliable results using conventional statistical methods, it aligns with our discriminator design.

Considering that model poisoning attacks often involve with multiplication by a certain dynamic or fixed value, the distribution of malicious weight parameters deviates and exhibits distinct tendencies compared to benign weight parameters. To address this, *Breakwater* performs a discriminating task that compares the weights from other nodes to one's own. This comparative analysis serves as an effective means of identifying potentially malicious weights from other nodes.

To enable a discriminator to learn the dynamics of model weight parameter distribution, it is essential to observe the history of weight values over time. This time series analysis provides the necessary context for the discriminator to comprehend the evolving patterns and fluctuations in model weight parameters. By considering temporal aspects of weight variations, the discriminator enhances its ability to discern between normal and abnormal weight behaviors, contributing to more robust defense within the federated learning framework.

Simultaneously, considering the necessity for this process to occur every epoch (or every n epoch) at each node in the aggregation phase, it becomes imperative that the process is lightweight and executed swiftly. Consequently, classifying weights from several lower nodes separately could be burdensome. To address this, we have devised a multi-label classifier capable of handling multiple weights from different nodes simultaneously, treating each as an independent case. This approach enables the discriminator to output the probability of each node being malicious efficiently.

By constructing a lightweight discriminator, it is advantageous to reduce the input dimension wherever possible. Recognizing that the latter layers' weight often contains crucial knowledge compared to the former ones since it directly contributes to the output, we opt to utilize only the weight parameters from the classifier layer. This selective approach ensures that the discriminator remains efficient while capturing essential information for the discriminating process. Discriminator D is defined as follows:

$$\begin{aligned}[p_t^{i+1(L)}, p_t^{i+1(R)}] = D(&W_t^i, W_{t-1}^i, W_{t-2}^i,\\ &W_t^{i+1(L)}, W_{t-1}^{i+1(L)}, W_{t-2}^{i+1(L)},\\ &W_t^{i+1(R)}, W_{t-1}^{i+1(R)}, W_{t-2}^{i+1(R)}),\end{aligned} \tag{3}$$

where $p_t^{i+1(L)}$ or $p_t^{i+1(R)}$ denotes the probability of left or right node at depth $i+1$ to be malicious at epoch t.

B. Defense Policy

We embed the discriminator's output to the defense mechanism to make a final decision on whether to discard the suspicious weight at epoch t, based on the probability p_t of left or right child node to be malicious. This decision is made by using a threshold, denoted as θ_i at depth i. In the context of a threshold value ranging between 0 and 1, a higher θ_i implies less stringent defense regarding the control of weight

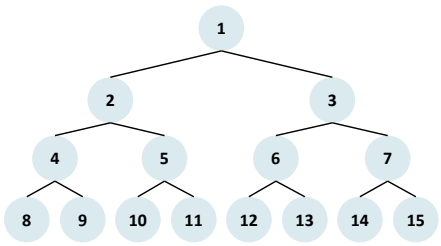


Fig. 2. A complete binary tree structure of 15 participant nodes

updates. This prioritizes aspects such as model learning speed over stability. Conversely, a lower θ_i signifies a stricter control mechanism, giving precedence to stability over generality. The choice of θ_i acts as a tuning parameter, allowing for the customization of the defense mechanism's behavior in response to the system's requirements and priorities. This adaptive approach ensures that the federated learning system can be tailored to strike a balance between model performance and stability based on the prevailing conditions and objectives with *Breakwater*.

In terms of the defense policy, recognizing an attacker's pattern systematically is also crucial, given their tendency to have specific attack patterns. To achieve this, the former probability of a certain node being malicious needs to be observed over subsequent time intervals. This consideration is computed by applying an exponentially weighted moving average as follows:

$$\bar{p}_t^i = (1-\alpha)\cdot \bar{p}_{t-1}^i + \alpha \cdot p_t^i, \tag{4}$$

where $\bar{p}_t^i$ indicates the final smoothed probability considered for a certain node at depth i to be malicious at time t, factoring in the former probabilities with exponent α. Taking into account the potential impact of attacks (with a larger impact near the root due to in-network aggregation), the θ_i value can be fine-tuned for each depth of the node.

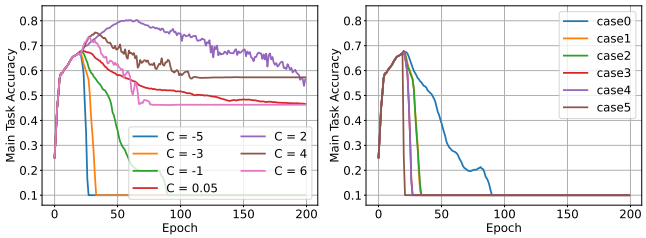
Based on the determined θ_i, the weight at the node with depth i is discarded if the following condition holds true:

$$\bar{p}_t^i \geq \theta_i. \tag{5}$$

Before aggregating the received weight with its own, each node runs the defense policy every time to ensure the safety of the global model. Since our *breakwater* runs at every certain epoch, even after discarding the weight of a certain node in a particular epoch, it is possible to rejoin the learning process afterward.

IV. Evaluation

We implemented *Breakwater* using *TensorFlow* 2.11. As shown in Fig. 2, we validated our system on the federated learning architecture with a complete binary tree topology of 15 nodes. For the main task of federated learning, each participant node trains the model of LeNet5 [8] using the dataset of Fashion-MNIST [12]. We mainly used the training data divided into each different 666 data for each node, and in heterogeneous settings, the training data is divided using



(a) Attack with various C value with attacker nodes of 14 and 15 (b) Attack with different attack nodes using $C = -1$

Fig. 3. Attack with different attack parameters

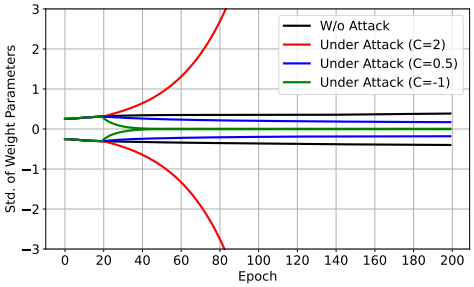


Fig. 4. A weight distribution of attacker nodes with different attack level C over epochs

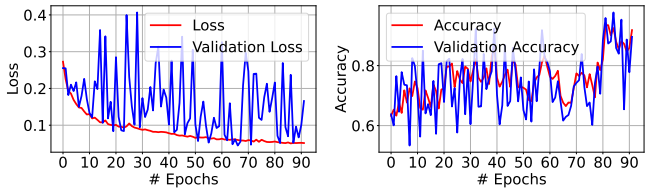
Dirichlet distribution [11]. Regarding the attack simulation, we multiplied a constant value (C) as done in [3] to the benign weights from some arbitrary attacking nodes within the tree.

Our anomaly discriminator used a lightweight learning model of MobileNet-V3 [7] with an activation function of the sigmoid. As the models closer to the outputs have summarized information, we only extract the last layer (i.e., classifier) of the main task model and pretrain the benign and attack scenarios using various C values. The value for θ_i is identical to θ across all of levels in the tree for experiments. For α, we mainly used 1, which means we fully reflect the instantaneous maliciousness, unless otherwise noted.

A. Observations

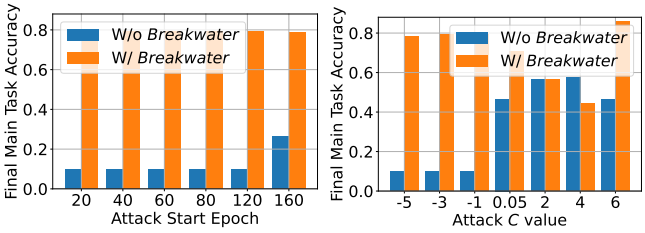
We first investigated how attacks affect the learning model by varying the attack level C and the attack node in Fig. 3. As shown in Fig. 3(a), we applied different attack levels C with positive values but larger than 1 ($C > 1$), positive values but smaller than 1 ($0 < C < 1$), and negative values ($C < 0$) from epoch 20. The main task accuracy shows a noticeable pattern that can be used for distinguishing purposes. Taking a closer look at the weight distribution regarding the C values in Fig. 4, the attack with large values of $C = 2$ leads to an increase in the weight deviations significantly, while the small positive value ($C = 1$) and the negative value ($C = -1$) rather sharply decrease the weight variances. This finding stresses out a necessity to monitor the changes in weight dynamics as a key indicator for identifying and understanding adversarial attacks in the federated learning settings.

We further examined the impact of attacks with various attack nodes with respect to different numbers of attack nodes, the position, and the depth of the attack node, as illustrated



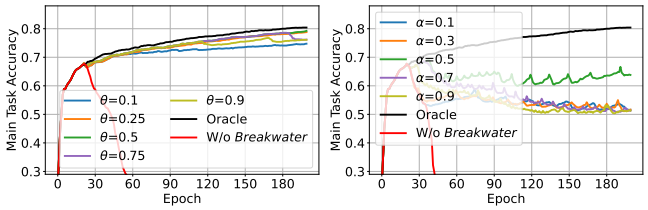
(a) Discriminator loss (b) Discriminator accuracy

Fig. 5. Global model performance under attack over epoch



(a) Defense for various starting epoch of attacks (b) Defense for various attack level C

Fig. 6. Performance of discriminator under various attack scenarios



(a) Various θ values (b) Various α values

Fig. 7. Effects of different defense parameters

in Fig. 3(b). Based on Fig. 2, the attack cases indicate the attack nodes as: case 0) 15; case 1) 14 and 15; case 2) 13 and 15; case 3) 13, 14, and 15; case 4) 7; and case 5) 3, respectively. If there are more number of attackers from cases 0, 1, to 3, it means that there are more harmful weights, and thus the global model loses its ability faster. Interestingly, as the model weights at a higher level are reflected more in the global model, the main task accuracy degrades earlier from cases 0, 4, to 5. In the same context, if the same number of attacker nodes are located at the same level as cases 1 and 2, it shows similar performance degradation. It implies that when the attacker is at a higher level, the impact of the attack can be more severe, highlighting the need for strict defense measures for nodes at the higher levels.

B. Performance of Anomaly Discriminator

Our anomaly discriminator is trained and validated using the classifier of the main task model trained with differently sampled data. Since a complete binary tree structure is used for experiments, our discriminator generates two different outputs that indicate the probability of maliciousness for the left child and the right child nodes, respectively. We have pretrained the discriminator using the balanced number of data for attacks on the left child node, attacks on the right child node, attacks on both child nodes, and non-attack of both child nodes. To prevent the overfitting problem, we have adopted an early stopping method with a patience of 5. As shown in Fig. 5, the discriminator converges after 90 training epochs with a reasonable categorical accuracy of 85 % and higher.

We validated our anomaly discriminator at different attack epochs of 5, 25, 50, 75, 100, and 125, respectively and various C values from 0.5 to 2. As shown in Fig. 6, the discriminator exhibits decent performance against various attack scenarios. Although we only trained the discriminator using C values of 0.5 and 2.0, our discriminator can even notice the unseen attack types with C of -5, -3, -1, 0.05, 4, and 6, implying the extensibility of our discriminator in unrevealed attacks.

C. Performance of Breakwater

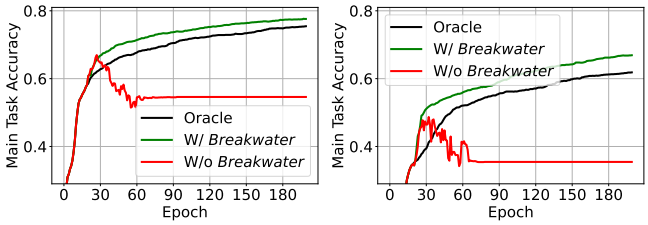
Given the probabilistic maliciousness of two child nodes from the discriminator, *Breakwater* interprets the risk level using a certain threshold θ and a weight α on a historical anomaly. As illustrated in Fig. 7(a), the lower θ value (e.g., 0.1) tends to branch out the nodes too readily, which means the removal of benign nodes and thus the loss of local data, leading to relatively low performance. In contrast, a large θ value (e.g., 0.9) allows too many opportunities for malicious nodes, resulting in model weight contamination. It means that we can control the strictness of our defense system by adjusting the value of θ. In our experimental settings, the θ value of 0.5 seems effective, and we used it as default.

In Fig. 7(b), we varied α, which is a weight of the former discriminator output. A higher α value implies that defense decisions are less influenced by historical maliciousness. In case of $\alpha = 1$, we use only the current state to detect anomaly. To evaluate the impact of historical behavior, we assume that the attacker engaged in the attack using three different C values, -0.5, -1, and -3, iteratively. Under this stealthy attack scenario controlling the impact of the attack, the global model is able to manage to secure the task performance with around 60 % accuracy. Specifically, the α value of 0.5 turns out to be the most effective, indicating that partially reflecting the anomaly history may assist anomaly detection. It can also be tuned considering potential attack behaviors.

D. Stability under Dynamic Federated Learning Scenarios

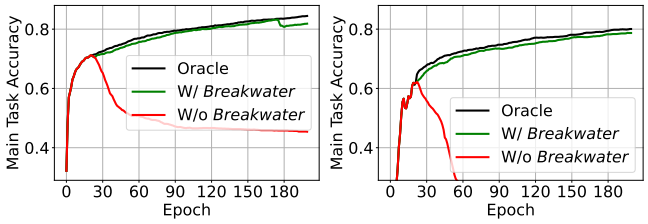
Lastly, we examined *Breakwater* under dynamic learning scenarios with heterogeneous data settings. We compared the performance of *Breakwater* with two baselines: 1) *Oracle*: an ideal defense case of discarding the malicious node immediately when the attacks occur, perfectly showing us an upper-bound under attack; and 2) *w/o Breakwater*: no defense method at all against the attacks.

Non-iid data. To simulate non-independent and identically distributed (iid) data, we used Dirichlet distribution [11] to divide the training data for heterogeneous participants. $Dir(\beta)$ refers to the Dirichlet distribution, where β is a concentration parameter, which is always larger than 0. The smaller β value means more heterogeneous. We validated the performance of *Breakwater* under attack by multiplying $C = 4$ to the weight starting from epoch 20 at two leaf nodes 13 and 15. As shown in Fig. 8, *Breakwater* secured the model from the accuracy



(a) $\beta = 0.5$ (b) $\beta = 0.1$

Fig. 8. Performance using non-iid data for each participant



(a) More data on node located in higher level (b) Data in-flow while training

Fig. 9. Performance under different number of data

drop. In Fig. 8(a), the gap between the *Oracle* and *Breakwater* is trivial, when the data is distributed using $\beta = 0.5$, indicating a less biased distribution. Interestingly, in Fig. 8(b), when using $\beta = 0.1$ of extremely heterogeneous data distribution, the performance with *Breakwater* even outperforms the *Oracle*. It is due to the adverse effect of the attack, where we used C value larger than 0. At the first few epochs, the attack rather helps learning similar to using the larger learning rate, before discarding the node with the potential intelligence extracted from the local data. It means that each participant has crucial data for the whole training in realistic heterogeneous learning scenarios, and the gradual and probabilistic expelling from *Breakwater* can not only defend, but also utilize the attack.

Realistic data distribution. Lastly, we assess the performance of *Breakwater* in more realistic settings under two scenarios involving different data sizes. In Fig. 9(a), we distributed more data to the node at a higher level, while all child nodes have half the number of local data. The attack node 15 (in Fig. 2) has C of -1. Although the different number of training data among nodes may lead to different learning speeds, our defense system successfully detects and prunes malicious nodes.

We also validated when the participating nodes keep collecting the training data, and thus the data is in-flowing in the middle of training. In Fig. 9(b), nodes 13 and 15 initiate an attack starting from epoch 20 with a C value of 4. In this context, as the model dynamically trains on data evolving over time, *Breakwater* can effectively identify malicious nodes. Moreover, by allowing the malicious node to participate for a few epochs after the attack commences, we can learn from the newly collected data before ultimately expelling the malicious node. It highlights that *Breakwater* is outstanding in situations with diverse data distributions even for unseen data in training sets, showing robustness and generalization.

V. Conclusions

We have presented *Breakwater*, an on-device self-debiasing framework to enhance security for multi-hop federated learning with edge devices. Our model poisoning attack-proof system incorporates a lightweight anomaly discriminator, ensuring learning protection in a distributed manner. After computing the maliciousness of the neighboring participants, we gradually branch out some potential suspicious attackers.

This work offers a lightweight self-debiasing model sharing mechanism in distributed edge networks in multi-hop federated learning. As our proposed system *Breakwater* is easily adaptable on underlying edge networks by using a simple tree-based structure, it has high potential to fully leverage model updates from other edge nodes, with a trade-off between model addition and model poisoning. As for future work, beyond detecting and expelling attackers, it would be interesting to seek some substitute participants and reorganize the aggregation paths to achieve more stable learning. Moreover, we may further optimize computational overhead of the decision-making process for more resource-constrained edge devices.

References

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

[2] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. *Advances in neural information processing systems*, 30, 2017.

[3] Lingjiao Chen, Hongyi Wang, Zachary Charles, and Dimitris Papailiopoulos. Draco: Byzantine-resilient distributed training via redundant gradients. In *International Conference on Machine Learning*, pages 903–912. PMLR, 2018.

[4] Xianhao Chen, Guangyu Zhu, Yiqin Deng, and Yuguang Fang. Federated learning over multihop wireless networks with in-network aggregation. *IEEE Trans. on Wireless Communications*, 21(6):4622–4634, 2022.

[5] Michael B Cohen, Yin Tat Lee, Gary Miller, Jakub Pachocki, and Aaron Sidford. Geometric median in nearly linear time. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 9–21, 2016.

[6] Elena Fasolo, Michele Rossi, Jorg Widmer, and Michele Zorzi. In-network aggregation techniques for wireless sensor networks: a survey. *IEEE Wireless Communications*, 14(2):70–87, 2007.

[7] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF ICCV*, pages 1314–1324, 2019.

[8] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[9] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, pages 1273–1282, 2017.

[10] Pinyarash Pinyoanuntapong, Prabhu Janakaraj, Pu Wang, Minwoo Lee, and Chen Chen. Fedair: Towards multi-hop federated learning over-the-air. In *2020 IEEE 21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pages 1–5, 2020.

[11] Gerd Ronning. Maximum likelihood estimation of dirichlet distributions. *Journal of statistical computation and simulation*, 32(4):215–221, 1989.

[12] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[13] Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. Generalized byzantine-tolerant sgd. *arXiv preprint arXiv:1802.10116*, 2018.

[14] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learning*, pages 5650–5659. PMLR, 2018.