MUDGUARD: Taming Malicious Majorities in Federated Learning using Privacy-preserving Byzantine-robust Clustering

Rui Wang
r.wang-8@tudelft.nl
Delft University of Technology
The Netherlands

Xingkai Wang
starshine87@sjtu.edu.cn
Shanghai Jiao Tong University
China

Huanhuan Chen
h.chen-2@tudelft.nl
Delft University of Technology
The Netherlands

Jérémie Decouchant
J.Decouchant@tudelft.nl
Delft University of Technology
The Netherlands

Stjepan Picek
stjepan.picek@ru.nl
Radboud University
The Netherlands

Nikolaos Laoutaris
nikolaos.laoutaris@imdea.org
IMDEA Networks Institute
Spain

Kaitai Liang
kaitai.liang@tudelft.nl
Delft University of Technology
The Netherlands

Abstract

Byzantine-robust Federated Learning (FL) aims to counter malicious clients and train an accurate global model while maintaining an extremely low attack success rate. Most existing systems, however, are only robust when most of the clients are honest. `FLTrust` (NDSS '21) and `Zeno++` (ICML '20) do not make such an honest majority assumption but can only be applied to scenarios where the server is provided with an auxiliary dataset used to filter malicious updates. `FLAME` (USENIX '22) and `EIFFeL` (CCS '22) maintain the semi-honest majority assumption to guarantee robustness and the confidentiality of updates. It is, therefore, currently impossible to ensure Byzantine robustness and confidentiality of updates without assuming a semi-honest majority. To tackle this problem, we propose a novel Byzantine-robust and privacy-preserving FL system, called `MUDGUARD`, to capture malicious minority and majority for server and client sides, respectively. Our experimental results demonstrate that the accuracy of `MUDGUARD` is practically close to the FL baseline using FedAvg without attacks (≈0.8% gap on average). Meanwhile, the attack success rate is around 0%-5% even under an adaptive attack tailored to `MUDGUARD`. We further optimize our design by using binary secret sharing and polynomial transformation, leading to communication overhead and runtime decreases of 67%-89.17% and 66.05%-68.75%, respectively.

CCS Concepts

• Computing methodologies → Machine learning; **• Security and privacy → Privacy-preserving protocols**; **Distributed systems security**.

Keywords

Federated Learning; Privacy Preservation; Robustness

ACM Reference Format:
Rui Wang, Xingkai Wang, Huanhuan Chen, Jérémie Decouchant, Stjepan Picek, Nikolaos Laoutaris, and Kaitai Liang. 2025. MUDGUARD: Taming Malicious Majorities in Federated Learning using Privacy-preserving Byzantine-robust Clustering. In *Abstracts of the 2025 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS Abstracts '25), June 9–13, 2025, Stony Brook, NY, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3726854.3727296

1 Introduction

Thanks to its privacy properties, Federated Learning (FL) [9] has been widely applied in real-world applications, e.g., prediction of the future oxygen requirements of symptomatic patients with COVID-19 [4]. Despite its attractive benefits, FL is vulnerable to Byzantine attacks. For example, attackers may choose to deteriorate the testing accuracy of models in an untargeted attack. Alternatively, they might fool models into predicting an attack-chosen label without downgrading the testing accuracy in a targeted attack. Many research works [6, 15, 17] have proved the vulnerability of FL via well-designed attack methods, e.g., poisoning training data or manipulating updates. Other studies [1, 3, 8, 10, 11, 13, 18, 19] have been dedicated to strengthening FL assuming that a minority of the clients can be malicious and that the server is honest.

Beyond Byzantine attacks, FL could put clients at high risk of privacy breach [7, 20] even if clients' datasets are maintained locally. Several studies [11, 13, 14] have applied secure tools, e.g., Additive Homomorphic Encryption (AHE) [12], Differential Privacy (DP) [5, 16], and Secure Multiparty Computation (MPC), to protect clients' updates[1]. However, these works only guarantee security when all servers are (semi-)honest and when a minority of the clients are malicious.

To the best of our knowledge, there does not exist any FL system that is capable of withstanding the presence of a majority of

[1]Note AHE and MPC require onerous computation over ciphertexts so that the computational complexity could naturally increase.

Byzantine clients, as well as malicious servers, while also guaranteeing the confidentiality of updates. One may think that existing Byzantine-robust solutions could be trivially extended to address the above challenge. However, that is not the case because they either violate privacy preservation requirements or are only effective in the honest majority scenario. For example, `FLTrust` [3] and `Zeno++` [18] require an auxiliary dataset that is independently and identically distributed (iid) with the clients' training datasets to rectify malicious updates, which evidently violates the clients' privacy. As for `FLAME` [11], it clusters updates and considers the smallest cluster as a malicious group, which makes sense in the malicious minority context. However, in the case of a malicious majority, it is difficult to assert if a given large/small-size cluster is malicious. `EIFFeL` [13] shows similar infeasibility, since it combines existing Byzantine-robust methods (e.g., `FLTrust` [3]) with secure aggregation [2].

We propose a practical and secure Byzantine-robust FL system, `MUDGUARD`, that defends against malicious entities (i.e., malicious minority for servers and malicious majority for clients) with privacy preservation.

2 MUDGUARD Overview and Design

2.1 Overview

The first goal of this work is to maintain the Byzantine robustness such that malicious updates should be excluded properly. To do so, the servers must separate the malicious clients from the semi-honest clients. DBSCAN helps the servers to perform clustering. Since the main difference between the malicious and the benign is in the direction and magnitude of the updates, we use the adjusted cosine similarity of updates as feature extraction to obtain better clustering accuracy. Under the (semi-)honest majority, the clustering result directly links to the group size. However, for a dynamic malicious majority, we cannot judge if a cluster is malicious only based on its size. To address this issue, we propose *Model Segmentation.* Unlike traditional FL generating "a unique" global model, our proposed algorithm can yield multiple aggregation results. It does not require the servers to know whether a given group is malicious or not. Moreover, it only aggregates the updates within the same cluster and then returns the results to the corresponding clients. We thus guarantee that the semi-honest will not be aggregated with the malicious.

As far as fighting against inference attacks is concerned, we should protect the confidentiality of the updates. For this, we use SS to wrap the updates into a secret shared format in the sense that individual secret shares cannot reveal the underlying information of the updates. By doing so, we guarantee that the updates are secured from eavesdroppers, semi-honest, or even malicious servers and can further be used on secure multiplication, comparison, and aggregation via cryptographic tools. However, using Secret Sharing (SS) alone is not sufficient to defend against differential attacks. To thwart the attack, we apply DP to prevent the attackers from extracting benign updates from the semi-honest group. Since injecting noise has a negative influence on the accuracy of the training model, we enable clients to perform denoising before wrapping the results into shares. Note that this does not invalidate DP due to the post-processing nature [5]. We also consider the malicious minority servers and thus leverage the Homomorphic Hash Function (HHF) to prevent malicious servers from performing incorrect aggregation, e.g., merging the gradients from two different groups.

2.2 System Design

Assume client $i \in [n]$ holds a horizontally partitioned dataset $\mathcal{D}_i$ satisfying $\mathcal{D} = \bigcup_{i=1}^{n} \mathcal{D}_i$, at t-th round, `MUDGUARD` works as follows.

Protocol MUDGUARD

❶ *Local Training.* For each local minibatch, each client conducts SGD and takes gradients $\mathbf{g}_t^i$ as updates.

❷ *Noise Injection.* Each client adds noise into $\mathbf{g}_t^i$ to satisfy DP: $\widetilde{\mathbf{g}}_t^i \leftarrow \mathbf{g}_t^i / \max(1, \|\mathbf{g}_t^i\|_2/\Delta) + \mathcal{N}(0, \Delta^2\sigma^2)$.

❸ *Denoising.* To improve accuracy, each client denoises $\widetilde{\mathbf{g}}_t^i$ by $\hat{\mathbf{g}}_t^i \leftarrow \mathsf{KS}(\widetilde{\mathbf{g}}_t^i, \mathcal{N}) \cdot \widetilde{\mathbf{g}}_t^i$, where $\mathsf{KS}(\cdot)$ is the KS distance.

❹ *SS.* Each client splits $\overline{\mathbf{g}}_t^i \leftarrow \mathsf{ECD}(\mathsf{sign}(\hat{\mathbf{g}}_t^i))$ into S shares by binary SS with Tiny Oblivious Transfer (OT) and sends the shares to S servers: $[[\overline{\mathbf{g}}_t^i]] \xleftarrow{SS} \overline{\mathbf{g}}_t^i$. Besides, by running HHF, all clients broadcast $\mathsf{H}_{\delta,\phi}(\mathsf{sign}(\hat{\mathbf{g}}_t^i))$.

❺ *Feature Extraction.* After receiving n shares, each server locally computes a pairwise adjusted cosine similarity matrix by bit-XOR: $[[\mathsf{CosM}_{ij}]] \leftarrow [[\overline{\mathbf{g}}_t^i]] \oplus [[\overline{\mathbf{g}}_t^j]]$, $i, j \in [n]$. To further compute L_2 distance, all servers convert Boolean shares to arithmetic shares by correlated randomness.

❻ *L_2 Distance Computation.* After conversion, deriving multiplicative SS, each server uses HE or OT to produce a triple, satisfying further multiplications. Therefore, each server takes $[[\mathsf{CosM}]]$ as the inputs of DBSCAN and then computes $[[\mathsf{EucM}]]$ by (a) pairwise subtraction: $[[vector_{ij}]] \leftarrow [[\mathsf{CosM}_i]] - [[\mathsf{CosM}_j]], i, j \in [n]$, (b) dot product: $[[x_{ij}]] \leftarrow [[vector_{ij}]] \cdot [[vector_{ij}]]$, and (c) approximated square root: $[[\mathsf{EucM}_{ij}]] \leftarrow 1 + \frac{[[x_{ij}]]-1}{2} - \frac{([[x_{ij}]]-1)^2}{8} + \frac{([[x_{ij}]]-1)^3}{16}$.

❼ *Element-wise Comparison.* By comparing each element of EucM with density parameter α, each server can derive shares of indicator matrix $[[\mathsf{IndM}]]$, $\{\mathsf{IndM}_{ij} = 1 \mid \mathsf{EucM}_{ij} \leq \alpha\}$.

❽ *Reconstruction.* All servers run a reconstruction algorithm to reveal IndM: $\mathsf{IndM} \xleftarrow{\mathsf{recon}} [[\mathsf{IndM}]]$ and broadcast it to the client side. By DBSCAN, one can derive cluster labels. Based on these labels, the clients learn about clustering information to perform aggregation verification in step ❿.

❾ *Model Segmentation.* The servers aggregate shares (based on the number of labels c) with the same labels after decoding: $\{[[\boldsymbol{G}_t^j]] \leftarrow \sum_{i \in c_j} \mathsf{DCD}([[\overline{\mathbf{g}}_t^i]]) \mid c_j = \{i \mid i \in [n]\}, j \in [c],\}$ and send to the corresponding clients.

❿ *Aggregation Verification.* After reconstructing aggregation, according to cluster labels, each client verifies aggregation by $\prod_{i \in c_j} \mathsf{H}_{\delta,\phi}(\mathsf{sign}(\hat{\mathbf{g}}_t^i)) \overset{?}{=} \mathsf{H}_{\delta,\phi}(\boldsymbol{G}_t^j)$. If the equation holds, clients accept the aggregation results; otherwise, reject and abort.

3 Evaluation

We set the baseline as a "no-attack-and-defense" FL, which means it excludes the use of any cryptographic tools as well as Byzantine-robust solutions but only trains with fully honest parties. This reaches the highest accuracy and fastest convergence speed for FL training. We then set #clients participating in the baseline training equal to the number of semi-honest clients in the malicious existence case. We conduct each experiment for 10 independent trials and further calculate the average to achieve smooth and precise accuracy performance. We evaluate MUDGUARD's accuracy and ASR by varying the total number of clients, the proportion of malicious clients, and the degree of non-iid; and further compare the performance with the baseline.

Under Gaussian Attack (GA), Adaptive Attack (AA), Backdoor Attack (BA), and Edge-case Attack (EA), the testing accuracy is on par with the baseline (with only a 0.008 gap on average) in MNIST. However, compared with the baseline, the results of MUDGUARD under LFA, Krum, and Trim attacks show slight drops (on average, 0.025 in MNIST). This is so because MUDGUARD has slow convergence and large fluctuation. This is incurred by two factors. To reduce the overheads of secure computations, we apply binary SS in SignSGD. SignSGD could cause negative impacts on clustering. Only taking the signs of the gradients can ignore the effect of the magnitudes of the malicious gradients. This makes the clustering a bit prone to inaccuracy. The other factor is that the LFA and Krum/Trim attacks either poison the training data and further poison updates or the local model to optimize the attacks. In the early stage of training, the malicious models do not perfectly fit the poisoned training data and local models yet. Thus, the semi-honest and malicious clients could be classified into the same cluster.

Semi-honest clients can obtain comparable accuracy to the baseline at the end of the training. The accuracy of the semi-honest group and the baseline sharply increases from 0.1 at epoch 0 to around 0.95 at epoch 25, then gradually converges to 0.97. In the GA, since the malicious group can only receive aggregation of noise, their accuracy always fluctuates around 0.1, equalling a random guess probability. As for LFA, the model accuracy gradually drops from 0.1 (at the beginning) to 0. This is because their models are trained on label-flipped datasets, while the labels of the testing set are not flipped. If the testing set is used to detect a poisoned model, the result should be flipped labels and failing to match the labels in the testing set, which results in 0. Since semi-honest and malicious clients can be classified into the same cluster at the beginning of the training, the accuracy of their models, w.r.t. malicious clients, is larger than 0.1 in some trials.

The accuracy of the semi-honest group under these attacks converges slightly slower than the baseline. Label Flipping, Krum, and Trim attacks aim to either train poisoned data or optimize local poisoned models to deteriorate the global model's testing accuracy. Due to the attacks being relatively slow and not as direct as GA, malicious updates cannot deviate 100% from benign updates at the beginning of the training (which means that malicious and semi-honest clients could be clustered together). However, with more training rounds, the deviation becomes clearer. Thus, MUDGUARD separates the two groups easily.

Acknowledgments

This work was partly supported by the European Union's Horizon Europe Research and Innovation Program under Grant No. 101073920 (TENSOR), No. 101070052 (TANGO), No.101070627 (REWIRE) and No.101092912 (MLSysOps). Nikolaos Laoutaris was supported by the MLEDGE project (REGAGE22e00052829516), funded by the Ministry of Economic Affairs and Digital Transformation and the European UnionNextGenerationEU/PRTR.

References

[1] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. 2017. Machine learning with adversaries: Byzantine tolerant gradient descent. In *NIPS*. 118–128.
[2] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical secure aggregation for privacy-preserving machine learning. In *CCS*. 1175–1191.
[3] Xiaoyu Cao, Minghong Fang, Jia Liu, and Neil Zhenqiang Gong. 2021. FLTrust: Byzantine-robust Federated Learning via Trust Bootstrapping. In *NDSS*.
[4] Ittai Dayan, Holger R Roth, Aoxiao Zhong, Ahmed Harouni, Amilcare Gentili, Anas Z Abidin, Andrew Liu, Anthony Beardsworth Costa, Bradford J Wood, Chien-Sung Tsai, et al. 2021. Federated learning for predicting clinical outcomes in patients with COVID-19. *Nature medicine* (2021), 1735–1743.
[5] Cynthia Dwork. 2008. Differential privacy: A survey of results. In *TAMC*. 1–19.
[6] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. 2020. Local Model Poisoning Attacks to {Byzantine-Robust} Federated Learning. In *USENIX Security*. 1605–1622.
[7] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. 2020. Inverting Gradients - How easy is it to break privacy in federated learning?. In *NIPS*. 16937–16947.
[8] K. Kumari, P. Rieger, H. Fereidooni, M. Jadliwala, and A. Sadeghi. 2023. BayBFed: Bayesian Backdoor Defense for Federated Learning. In *IEEE Symposium on Security and Privacy (SP)*.
[9] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*. 1273–1282.
[10] El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Rouault. 2018. The hidden vulnerability of distributed learning in byzantium. In *ICML*. 3521–3530.
[11] Thien Duc Nguyen, Phillip Rieger, Huili Chen, Hossein Yalame, Helen Möllering, Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Shaza Zeitouni, Farinaz Koushanfar, Ahmad-Reza Sadeghi, and Thomas Schneider. 2022. FLAME: Taming Backdoors in Federated Learning. In *USENIX Security*.
[12] Pascal Paillier. 1999. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*. 223–238.
[13] Amrita Roy Chowdhury, Chuan Guo, Somesh Jha, and Laurens van der Maaten. 2022. EIFFeL: Ensuring Integrity for Federated Learning. In *CCS*. 2535–2549.
[14] Stacey Truex, Nathalie Baracaldo, Ali Anwar, Thomas Steinke, Heiko Ludwig, Rui Zhang, and Yi Zhou. 2019. A hybrid approach to privacy-preserving federated learning. In *AISec*. 1–11.
[15] Hongyi Wang, Kartik Sreenivasan, Shashank Rajput, Harit Vishwakarma, Saurabh Agarwal, Jy-yong Sohn, Kangwook Lee, and Dimitris Papailiopoulos. 2020. Attack of the Tails: Yes, You Really Can Backdoor Federated Learning. In *NIPS*. 15 pages.
[16] Nan Wu, Farhad Farokhi, David Smith, and Mohamed Ali Kaafar. 2020. The value of collaboration in convex machine learning with differential privacy. In *IEEE S&P*. 304–317.
[17] Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. 2020. DBA: Distributed Backdoor Attacks against Federated Learning. In *ICLR*.
[18] Cong Xie, Sanmi Koyejo, and Indranil Gupta. 2020. Zeno++: Robust fully asynchronous SGD. In *ICML*. 10495–10503.
[19] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. 2018. Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates. In *ICML*. 5650–5659.
[20] Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep leakage from gradients. *NIPS* (2019).