FEDCLEAN: A DEFENSE MECHANISM AGAINST PARAMETER POISONING ATTACKS IN FEDERATED LEARNING

Abhishek Kumar[⋆⋆] *Vivek Khimani*[‡] *Dimitris Chatzopoulos*[⋆⋆] *Pan Hui*[†⋆†]

[⋆] University of Helsinki, Finland
[‡] Drexel University, USA
[⋆⋆] University College Dublin, Ireland
[†] Hong Kong University of Science and Technology, Hong Kong SAR

ABSTRACT

In Federated learning (FL) systems, a centralized entity (server), instead of access to the training data, has access to model parameter updates computed by each participant independently and based solely on their samples. Unfortunately, FL is susceptible to model poisoning attacks, in which malicious or malfunctioning entities share polluted updates that can compromise the model's accuracy. In this study, we propose FedClean, an FL mechanism that is robust to model poisoning attacks. The accuracy of the models trained with the assistance of FedClean is close to the one where malicious entities do not participate.

Index Terms— Federated learning, model poisoning, active learning, reputation

1. INTRODUCTION

Federated learning (FL) is a machine learning (ML) technique for training models across multiple entities without requiring the exchange of locally stored data but only the exchange of parameters [1, 2]. FL is vunerable to model poisoning by adversaries that contribute poisoned parameters [3]. The majority of existing defense mechanisms against parameter poisoning are based either on the assumption that the data samples generated on every agent follow the same distribution [4] or on update aggregation techniques that focus on the similarity of updates with each other [5, 6].

In this work we propose FedClean, a defense mechanism for FL systems against poisoning attacks that is based on the principles of active learning [7]. It contains a reputation mechanism, which keeps track of each agent's credibility and assists in selecting trustworthy agents for model training, and a update quality control mechanism that detects malicious updates. We depict the performance of FedClean against four attacks , and compare with four benchmarks (FedAvg without malicious agents [1], RSA [6], Krum [8], and Coomed [9]).

*Corresponding author: abhishek.kumar@helsinki.fi

†Thanks to the Academy of Finland for funding.

2. PROBLEM STATEMENT

We consider an FL model $\mathcal{M}$ that is described by a vector of parameters $W_{\mathcal{M}} \in R^N$ and is trained for T rounds of length τ time slots via a training dataset, $\mathcal{D}$, which is distributed and privately stored among a set of K agents $\mathcal{A}$ (i.e., $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2 \cup \ldots \cup \mathcal{D}_K$). The dimensionality of the parameter space, N, depends on the type of the model (e.g., neural networks). The parameter vector $W_{\mathcal{M}}$ is obtained by distributed training and aggregation over the updates provided by selected agents. At each round t, a random subset of agents, $\mathcal{A}^t \subset \mathcal{A}$, is chosen by the server for synchronous aggregation [1] for a period of τ slots. Every agent $i \in \mathcal{A}^t$, minimizes the empirical loss using a loss function L over its own data shard $\mathcal{D}_i^{t \to t+\tau}$ that is stored privately and generated between t and $t+\tau$, by starting from the global weight vector $W_{\mathcal{M}}^t$ and running an algorithm such as stochastic gradient decent (SGD) [10] for τ slots. At the end of its run, each agent i obtains a local weight vector $W_{\mathcal{M}_i}^{t+\tau}$ using only her privately stored data samples and a mutually agreed loss function L, computes its local update

$$\delta_i^{t+\tau} = W_{\mathcal{M}_i}^{t+\tau} - W_{\mathcal{M}}^t \tag{1}$$

and sends it to the server. To obtain the global weight vector $W_{\mathcal{M}}^{t+\tau}$ for the next round, any aggregation mechanism can be used over the collected parameter updates $\Delta^{t+\tau} = \left\{\delta_i^{t+\tau}\right\}_{i \in \mathcal{A}^t}$. One of the most commonly used aggregation mechanisms is weighted averaging based aggregation:

$$W_{\mathcal{M}}^{t+\tau} = W_{\mathcal{M}}^t + \sum_{i \in \mathcal{A}^t} \delta_i^{t+\tau} \Gamma_i(t) \tag{2}$$

$$= W_{\mathcal{M}}^t + \sum_{i \in \mathcal{A}^t} \left(W_{\mathcal{M}_i}^{t+\tau} - W_{\mathcal{M}}^t\right) \Gamma_i(t), \tag{3}$$

where $\sum \Gamma_i(t) = 1$. The quality of $\mathcal{M}$ depends on $W_{\mathcal{M}}$ that depends on agents' updates during the training period. Assuming that a subset of $\mathtt{P}$ agents submit poisonous updates $\Delta_{\mathtt{P}}^t \subset \Delta^t$, the server can only observe each $\delta_i^t \in \Delta^t$ and decide whether to accept it or not. Although the server cannot

control the quality of each individual update, it can control the subset of the selected agents on each training iteration t, $\mathcal{A}^t$, and the way the submitted updates will be aggregated, $\Gamma_i(t)$.

FedClean incorporates a reputation-based sampling strategy to select agents on each round and a classifier hosted in an oracle to detect poisonous updates.

3. FEDCLEAN DESIGN

FedClean protects the iterative training of an FL model, $\mathcal{M}$, from poisonous updates $\Delta_{\mathrm{p}}^t \subset \Delta^t$ using an agent reputation model, γ, and a hypersphere model $\mathcal{H}$ assisted by an Oracle $\mathcal{O}$. The ideal scenario for FedClean is to filter out any poisonous updates before being included in model training. However, it is difficult to design a solution which can filter out poisonous updates with 100% accuracy. So, FedClean introduces a two-layer defense mechanism to address model poisoning attacks in FL.

Reputation based Sampling Strategy: In order to determine a random subset of agents $\mathcal{A}^t \subset \mathcal{A}$ that will contribute on the next training round, we use two factors: *(i)* agents' reputation score, and *(ii)* an inclusion bias. We use a Bayesian framework [11, 12] to determine the probability of agent i to make an honest contribution in a given round and use a beta distribution $Beta(\alpha_i, \beta_i)$ to determine this probability. If there is no prior information about agent i, $\alpha_i = \beta_i = 1$ which means that the distribution is uniform. The reputation of agent i, γ_i, is $\gamma_i = \alpha_i/(\alpha_i + \beta_i)$. Whenever an agent, i, is sampled δ_i is examined by $\mathcal{H}$ and if the update is included for training, α_i increases by 1, and so i's reputation, otherwise β_j increases by 1, and i's reputation decreases.

Basing agent sampling entirely on reputation score has two pitfalls: *(i)* some agents who were chosen in initial training rounds and made honest contribution will be sampled more frequently, leaving many agents unexplored, and *(ii)* some agents may wrongly be penalized by the hypershapere classifier, or some agents may send malicious updates in given round, but may want to contribute honestly in future rounds for rewards. To tackle these pitfalls, we introduce an inclusion bias $b_i(t)$ for agent i at round t, which is calculated as follows

$$b_i(t) = b_i(t-1) + 1 \text{ if } i \notin \mathcal{A}^t \text{ or } 0 \text{ if } i \in \mathcal{A}^t \tag{4}$$

and increases the chances of non recently selected agents.

We combine the reputation model and the inclusion bias to propose an agent selection mechanism in which the probability of agent i to be selected at round t is:

$$P[i \in \mathcal{A}^t] = \chi \frac{\gamma_i}{\sum_{i \in \mathcal{A}} \gamma_i} + (1-\chi)\frac{b_i}{\sum_{i \in \mathcal{A}}(b_i)}, \tag{5}$$

where $\chi \in [0,1]$ is a trade-off factor. In essence, reputation-based agent sampling mechanism tries to sample agents often who are likely to make honest contribution, while being inclusive to agents with low reputation.

Peer Truth Serum aided Hypersphere based Update Selection: $\mathcal{H}$ is modeled as a hypersphere-based anomaly detection approach equipped with an active learning strategy to adjust an anomaly threshold similarly to support vector domain description (SVDD) [13]. The goal of the SVDD is to find a concise description of the normal updates such that malicious data can be easily identified. In the underlying one-class scenario, this translates to finding a minimal enclosing hypersphere of center $\mathcal{H}_c$ and radius $\mathcal{H}_R$ that contains the honest training updates.

$$f^{\mathcal{H}}(\delta_i) = ||\delta_i - \mathcal{H}_c||^2 - {\mathcal{H}_R}^2 \tag{6}$$

the boundary of the hypersphere is described by the set $\{\delta : f^{\mathcal{H}}(\delta_i) = 0 \cap \delta_i \in \Delta\}$. That is, the parameters of $f^{\mathcal{H}}$ are to be chosen such that $f^{\mathcal{H}}(\delta_i) \leq 0$ for honest updates and $f^{\mathcal{H}}(\delta_i) > 0$ for poisonous updates. Using $f^{\mathcal{H}}$, the server is able to classify agent updates, $\{\delta_i^{t+T}\}_{i \in \mathcal{A}^t}$, into honest and polluted and update of their reputation accordingly. In the examined setting, all the updates are unlabeled, since we do not have information whether they are honest or poisonous. We use a random sampling strategy to sample some of these unlabeled updates to receive their labels from an Oracle $\mathcal{O}$ in such way that having labeled information on those updates provide significantly high accuracy for the global model $\mathcal{M}$. After receiving labels from $\mathcal{O}$, center $\mathcal{H}_c$ and radius $\mathcal{H}_R$ of the hypersphere are determined from the following optimization problem (as adopted by [14]):

$$\begin{aligned} \min_{\mathcal{H}_c, \mathcal{H}_R, \mu, \xi} \quad & \mathcal{H}_R^2 - \kappa\mu + \eta_u \sum_{i=1}^{n} \xi_i + \eta_l \sum_{j=n+1}^{n+m} \xi_j \\ \text{s.t.} \quad & \forall_{i=1}^{n} : ||\phi(\delta_j) - \mathcal{H}_c||^2 \leq \mathcal{H}_R^2 + \xi_i \\ & \forall_{j=n}^{n+m} : y_j \left(\mathcal{H}_R^2 - ||\phi(\delta_j) - \mathcal{H}_c||^2\right) \leq -\mu + \xi_j \\ & \forall_{i=1}^{n} : \xi_i \geq 0, \text{and } \forall_{j=n+1}^{n+m} : \xi_j \geq 0 \end{aligned} \tag{7}$$

where μ is a margin, κ, η_u, and η_l are trade-off parameters which balance margin-maximization and the impact of unlabeled and labeled updates respectively, n is number of unlabeled updates, m is number of labeled updates, and y_j is the label given by Oracle as in Equation 8. The additional slack variables ξ_i are bound to labeled examples and allow for point-wise relaxations of margin violations by labeled example.

$$y_j = \begin{cases} +1, & \text{if Oracle O considers } \delta \text{ benign} \\ -1, & \text{if Oracle O considers } \delta \text{ malicious} \end{cases} \tag{8}$$

Hypersphere $\mathcal{H}$ decides which updates should be included on order to update global model $\mathcal{M}$. If the update δ_i lies within hypersphere, δ is included to update $\mathcal{M}$, and agent i's reputation increases as per reputation update mechanism. Similarly, if the update δ_i lies outside the hypersphere, δ is excluded to update $\mathcal{M}$, and agent i's reputation decreases reputation update mechanism. Before describing the calculation of $W_{\mathcal{M}}^{t+\tau}$

given $W_{\mathcal{M}}^{t}$ and $\Delta^{t+\tau}$ we need to highlight that the submitted updates $\{\delta_i^{t+\tau}\}$ are high dimensional and non-linear in nature. For example, the deep neural network model we use on FashionMNIST (fMNIST) dataset for evaluation purpose has total 3,382,346 parameters across different layers. The employed hypersphere based approach does not work in such high dimensions. For that, we flatten all these parameters into one row vector, and use a two-layer deep-forward auto-encoder function ϕ in order to retrieve its corresponding (compressed) representation in lower dimension, i.e. $\phi(\delta_i)$. We pick the size of the compressed vector to be 10,000. Given the submitted updates, FedClean integrates a classifier that processes these updates and determines if they should be considered in the calculation of $W_{\mathcal{M}}^{t+\tau}$.

Additionally, FedClean employs a second sampling strategy which objective is to select a number of unlabeled data points (or updates) to query Oracle $\mathcal{O}$ about it under the budget B (lesser the amount of queries is, better it is), such that having information regarding labels of these data points will provide significant boost to the accuracy (See Equation 8). This strategy samples updates lying either in previously unexplored region, or in uncertain and unexplored region preferred sampling strategy.

Formulation of Oracle: Oracle $\mathcal{O}$ randomly selects a subset of $\mathtt{S}$ agents in each iteration from the set of the agents that have not been selected for training. Each of these $\mathtt{S}$ agents, receives the same input as any agent in $\mathcal{A}^t$, calculates her update $\delta_{\mathtt{S}_i}^{t+\tau}$ and sends it to the server. So, by the end of each iteration, the server receives two sets of updates: $\Delta^{t+\tau}$ and $\Delta_{\mathtt{S}}^{t+\tau}$ while $\Delta^{t+\tau}$ includes a subset $\Delta_{\mathtt{P}}^{t+\tau}$ of poisonous updates. The goal of the oracle is to use $\Delta_{\mathtt{S}}^{t+\tau}$ to detect $\Delta_{\mathtt{P}}^{t+\tau}$ in $\Delta^{t+\tau}$. It is worth noting that since the $\mathtt{S}$ are not selected based on their reputation but uniformly at random (i.e., each of them has a probability of 1 over $K - \mathtt{S}$ to be selected), $\Delta_{\mathtt{S}}^{t+\tau}$ is not considered in the updated of the model but only for the detection of the poisonous updates as discussed below.

The server, after receiving $\Delta^{t+\tau}$ and $\Delta_{\mathtt{S}}^{t+\tau}$ calculates the cosine similarity of every update in $\Delta^{t+\tau}$ every update in $\Delta_{\mathtt{S}}^{t+\tau}$. If the similarity between an update in $\Delta^{t+\tau}$ and an update in $\Delta_{\mathtt{S}}^{t+\tau}$ is less than 0.5 the update in $\Delta^{t+\tau}$ is marked. If one update is marked by the majority of the updates in $\Delta_{\mathtt{S}}^{t+\tau}$, it is considered as poisonous and it is not included in the calculation of the new weights, as described in Equation 8.

Each of the $\mathtt{S}$ agents submits the produced update in the same way as the agents in $\mathcal{A}^t$ and gets paid a default token for their effort. If the test accuracy of the global model does not degrade after including those updates in training, they get an additional token. This ensures that the best strategy for a rational agent is to calculate their model update in non-malicious manner as proven by the authors of [15]. This approach is also known as peer truth serum. Since the member of Oracle set is always selected randomly, an attacker can not influence selection of the $\mathtt{S}$ agents.

Considering that the data each agent stores do not follow the same distribution it is possible for the cosine similarity between two updates to be low even if both of them are benign. For that $\mathtt{S}$ needs to be high enough. As depicted by the experiments in Section 4, the higher the number of $\mathtt{S}$ the faster the accuracy of the trained model converges to the highest possible value.

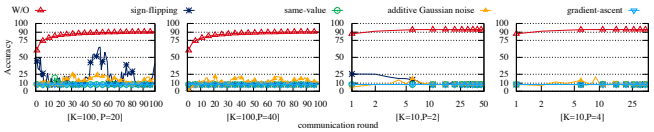

Fig. 1. Federated averaging under Sign-flipping (SF), same-value (SV), additive Gaussian noise (AGN), and gradient-ascent (GA) attacks without any defense mechanism.

Given the honest contributions $\{\delta_i^{t+\tau}\}$ we need the weights assigned to agents, $\{\Gamma_i(t)\}$ to calulate $W_{\mathcal{M}}^{t+\tau}$ as shown in equation 3. We use a variant of *Federated Averaging* (FedAvg) [1] for weight aggregation $\Gamma_i(t) = \gamma_i / \sum_{k \in \mathcal{A}_{\mathcal{H}}^t} \gamma_k$, where $\mathcal{A}_{\mathcal{H}}^t$ are the agents whose contributions were selected by $\mathcal{H}$ to update $\mathcal{M}$.

Convergence of Reputation-based FedAvg. As the underlying requirements for convergence of reputation-based FedAvg (i.e. FedClean) are the same as ones for FedAvg [1] which has been shown to be convergent, we claim that this aggregation mechanism also leads to the model convergence.

4. PERFORMANCE EVALUATION

We evaluate the robustness of FedClean against four attacks from malicious agents and compare them with four benchmark algorithms on a GeForce GTX 1080 Ti GPU computer. We use the fMNIST dataset, which has 60k training samples and 10k testing samples. We use a 3-layer Convolutional Neural Network (CNN) with dropout as the model architecture. With centralized training, this model achieves 91.7% accuracy on fMNIST. We experiment with two values for the number of the agents in two scenarios. In the first scenario we consider $K = 10$, $T = 50$. Here, we select all agents for training in each training round. In the second scenario, we consider $K = 100$, $T = 100$. Here, we use reputation-based agent sampling mechanism (Equation 5) to select 10 agents in each iteration for training. In both scenarios, we train the model until we reach an accuracy of 91% or reach T epochs. In both scenarios we examine two cases of malicious agents, one where they are 20% of the agents (i.e., $\mathtt{P} = 0.2K$) and one where they are 40% (i.e., $\mathtt{P} = 0.4K$). For the oracle we employ five agents on every iteration (i.e., $\mathtt{S} = 5$), so in the first scenario we have 15 agents in total since all the agents are considered on every iteration while in the second scenario $\mathtt{S} = 0.05K$.

Impact of Reputation. As described by Equation 5, the probability of an agent to be selected on a round depends on her

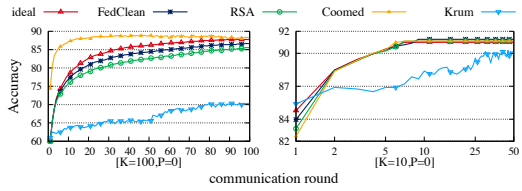

Fig. 2. Cost of resilience (i.e., accuracy under no attack).

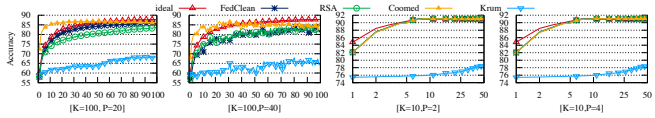

Fig. 3. Sign-flipping attack.

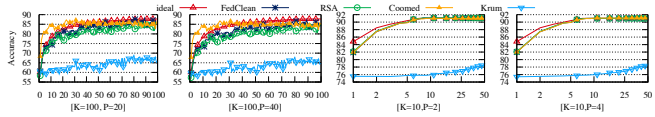

Fig. 4. Same value attack.

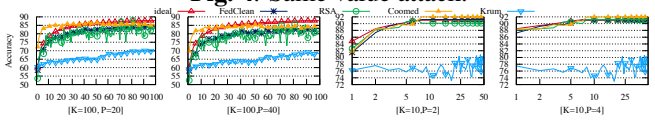

Fig. 5. Additive Gaussian noise attack.

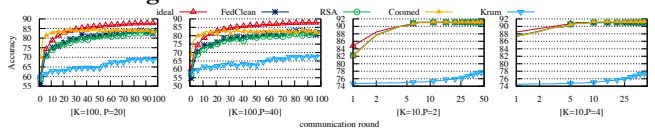

Fig. 6. Gradient ascent attack.

Fig. 7. Accuracy under SF, SV, AGN, and GA attacks.

serviceableness in the previous rounds and an inclusion bias factor.

Robustness of FedClean. Figure 1 shows that FedAvg (line labeled with "W/O" for the case of not being under an attack) is vulnerable and needs a defense mechanism. Figure 2 shows that FedClean shows overhead of defense when $\mathtt{P} = 0$. Next we assess how FedClean can make FedAvg more robust against four attacks adopted from [6, 16].

1) *Sign-flipping attack (SF).* Malicious agents flip the sign of their contribution and enlargen the magnitude. As demonstrated in Figure 3, for $K = 100$, FedClean performs marginally better than RSA, and slightly worse than Coomed and much better than Krum.

2) *Same Value Attack (SV).* Malicious agents perform same-value attacks by setting their updates as $c\mathbf{1}$, where $\mathbf{1}$ is an all-one vector and c is a constant which is set to 100. As demonstrated in Figure 4, FedClean performs marginally better than RSA and Coomed, and significantly better than Krum.

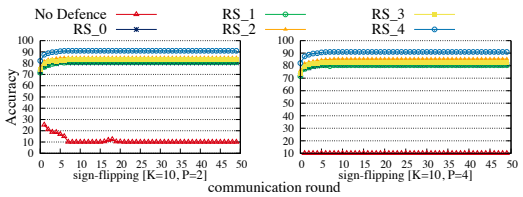

Fig. 8. Accuracy of FedClean with different sampling strategies under sign-flipping attacks.

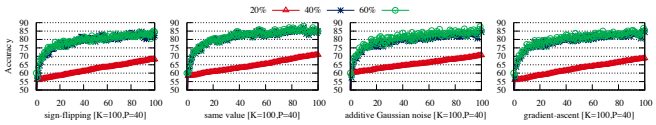

Fig. 9. FedClean Performance with different level of Oracle's budget.

3) *Additive Gaussian Noise Attack (AGA).* Malicious agents calculate $\{\delta_i^{t+\tau}\}$ as expected but add Gaussian Noise $\mathcal{N}(0, 100)$ before sending it to the server. As demonstrated in Figure 5, FedClean, RSA, and Coomed are close to FedAvg, and outperform Krum in both settings. FedClean performs marginally better than RSA, significantly better than Krum, and slightly worse than Coomed.

4) *Gradient Ascent Attack (GA).* Malicious agents run gradient ascent, instead of gradient descent during local training [16]. As demonstrated in Figure 6, FedClean performs marginally better than Coomed, significantly better than Krum, and slightly worse than RSA.

The Impact of the Oracle in FedClean's Performance. Figure 9 depicts the performance gain caused when Oracle provides label for 20%, 40%, 60% of the total sampled updates in the given iteration. With the budget being 40%, which we used in all the experiments unless stated differently, the accuracy of FedClean is close to other benchmarks and significantly higher than Krum and FedAvg (under attack). Higher is the fraction of the updates being examined by Oracle, faster is the model convergence to the maximum accuracy.

5. CONCLUSION

In this work, we propose *FedClean*, a mechanism for defending FL server against model poisoning attacks. The server's defense functionality is enhanced by *(i)* a reputation mechanism that keeps track of the credibility of each agent and assists on selecting trustworthy agents on each round to train the model, and *(ii)* a peer truth serum-based quality control mechanism that categorizes the contributed updates into honest and malicious.

6. REFERENCES

[1] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, Aarti Singh and Jerry Zhu, Eds., Fort Lauderdale, FL, USA, 20–22 Apr 2017, vol. 54 of *Proceedings of Machine Learning Research*, pp. 1273–1282, PMLR.

[2] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 2, pp. 12:1–12:19, Jan. 2019.

[3] Yann Fraboni, Richard Vidal, and Marco Lorenzi, "Free-rider attacks on model aggregation in federated learning," in *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, Arindam Banerjee and Kenji Fukumizu, Eds. 13–15 Apr 2021, vol. 130 of *Proceedings of Machine Learning Research*, pp. 1846–1854, PMLR.

[4] Shiqi Shen, Shruti Tople, and Prateek Saxena, "Auror: defending against poisoning attacks in collaborative deep learning systems," in *Proceedings of the 32nd Annual Conference on Computer Security Applications*, New York, NY, USA, 2016, ACSAC '16, p. 508–519, Association for Computing Machinery.

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

[6] Liping Li, Wei Xu, Tianyi Chen, Georgios B Giannakis, and Qing Ling, "Rsa: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019, vol. 33, pp. 1544–1551.

[7] Burr Settles, "Active learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 6, no. 1, pp. 1–114, 2012.

[8] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. 2017, vol. 30, Curran Associates, Inc.

[9] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *Proceedings of the 35th International Conference on Machine Learning*, Jennifer Dy and Andreas Krause, Eds. 10–15 Jul 2018, vol. 80 of *Proceedings of Machine Learning Research*, pp. 5650–5659, PMLR.

[10] Léon Bottou and Olivier Bousquet, "The tradeoffs of large scale learning," in *Advances in Neural Information Processing Systems*, J. Platt, D. Koller, Y. Singer, and S. Roweis, Eds. 2008, vol. 20, pp. 161–168, Curran Associates, Inc.

[11] Sonja Buchegger and Jean-Yves Le Boudec, "A robust reputation system for mobile ad-hoc networks," *P2PEcon*, 2004.

[12] Dimitris Chatzopoulos, Mahdieh Ahmadi, Sokol Kosta, and Pan Hui, "Flopcoin: A cryptocurrency for computation offloading," *IEEE Transactions on Mobile Computing*, vol. 17, no. 5, pp. 1062–1075, May 2018.

[13] Ke Wang and Salvatore J Stolfo, "Anomalous payload-based network intrusion detection," in *International Workshop on Recent Advances in Intrusion Detection*, 2004, pp. 203–222.

[14] Nico Görnitz, Marius Kloft, Konrad Rieck, and Ulf Brefeld, "Active learning for network intrusion detection," in *Proceedings of the 2nd ACM Workshop on Security and Artificial Intelligence*, New York, NY, USA, 2009, AISec '09, p. 47–54, Association for Computing Machinery.

[15] Goran Radanovic, Boi Faltings, and Radu Jurca, "Incentives for effort in crowdsourcing using the peer truth serum," *ACM Transactions on Intelligent Systems and Technology*, vol. 7, no. 4, Mar. 2016.

[16] Suyi Li, Yong Cheng, Yang Liu, Wei Wang, and Tianjian Chen, "Abnormal client behavior detection in federated learning," *CoRR*, vol. abs/1910.09933, 2019.