TESSERACT: GRADIENT FLIP SCORE TO SECURE FEDERATED LEARNING AGAINST MODEL POISONING ATTACKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Federated learning—multi-party, distributed learning in a decentralized environment—is vulnerable to model poisoning attacks, more so than centralized learning. This is because malicious clients can collude and send in carefully tailored model updates to make the global model inaccurate. This motivated the development of Byzantine-resilient federated learning algorithms, such as Krum, Bulyan, FABA, and FoolsGold. However, a recently developed untargeted model poisoning attack showed that all prior defenses can be bypassed. The attack uses the intuition that simply by changing the sign of the gradient updates that the optimizer is computing, for a set of malicious clients, a model can be diverted from the optima to increase the test error rate. In this work, we develop **TESSERACT**—a defense against this directed deviation attack, a state-of-the-art model poisoning attack. TESSERACT is based on our intuition that in federated learning, certain patterns of gradient flips are indicative of an attack. This intuition is remarkably stable across different learning algorithms, models, and datasets. TESSERACT assigns reputation scores to the participating clients based on their behavior during the training phase and then takes a weighted contribution of the clients. We show that TESSERACT provides robustness against even a white-box version of the attack.

1 INTRODUCTION

Federated learning (FL) Smith et al. (2017); Yang et al. (2019) offers a way for multiple clients on heterogeneous platforms to learn collaboratively without sharing their local data. The clients send their local gradients to the parameter server that aggregates the gradients and updates the global model for the local clients to download. FL can be attacked during the training phase by compromising a set of clients that then send maliciously crafted gradients. The attack can be targeted against particular data instances or can be untargeted. The latter brings down the overall accuracy by affecting *all* classes. To counter this threat, a set of approaches has been developed for countering Byzantine clients in FL, *e.g.*, Krum Blanchard et al. (2017), Bulyan Mhamdi et al. (2018), Trimmed Mean and Median Yin et al. (2018), FoolsGold Fung et al. (2020), and FABA Xia et al. (2019). In this work, we use the state-of-the-art (SOTA) untargeted model poisoning attack called a *directed deviation attack*, proposed recently in Fang et al. (2020). *This has been shown to bypass all existing Byzantine-robust aggregation techniques,* e.g.*, Krum, Bulyan, Trimmed mean, and Median.* In our experiments, this attack has been found to decrease the test accuracy from 90% to a low 9% when a DNN is trained using Krum on the MNIST dataset, distributed among 100 clients, 20 of which are under the attacker's control. We describe the relevant details of this attack in Section 2.2.

Our solution. We propose a novel defense called **TESSERACT** against untargeted model poisoning attacks (Figure 1), which uses a stateful model to reduce the contributions by suspicious clients to the global model update. We show that where all prior Byzantine-resilient federated learning approaches fail against the directed-deviation attack of Fang et al. (2020), which is the state-of-the-art untargeted model poisoning attack, TESSERACT is able to recover the test accuracy of the trained model. This benefit applies even when the attack knows the algorithm and all the parameters of our defense, *i.e.*, an adaptive white-box attack. TESSERACT is based on a simple intuition that for a sufficiently small learning rate, as the model approaches an optima in a benign setting, a large number of gradients do not flip their direction with large magnitudes, that is, a degree of inertia is maintained. Our intuition is supported from the analysis we show in Figure 2. We capture this quantitatively in a metric that we propose, called ***flip-score***, which is the sum of squares of gradient magnitudes of all parameter updates that suggest a flip in the gradient direction from the previous global update. We find that

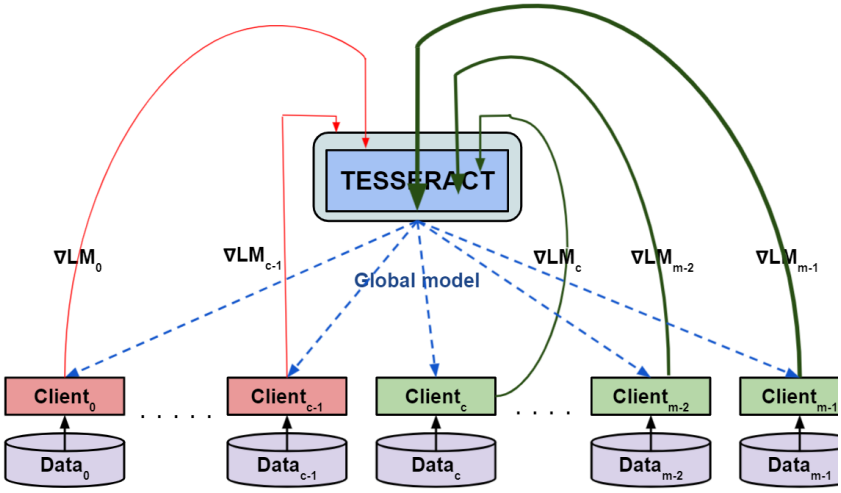
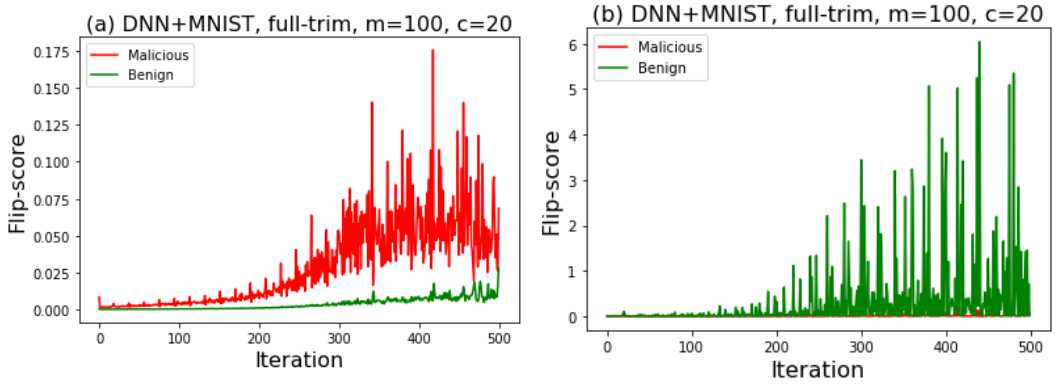


Figure 1: ***TESSERACT's architecture where c out of m clients may be malicious and send carefully crafted values of their local models to throw the global model off convergence. TESSERACT weights the gradients, received from the clients, by their reputation scores before aggregation, (see varying and thicknesses of arrows from the clients.)***

our intuition and correspondingly the defense TESSERACT also holds for other attacks, *e.g.*, the targeted and untargeted label flipping attacks (Appendix § **??**). However, the other attacks are less damaging, consistent with the observation in Fang et al. (2020). Thus, in the main paper, we focus on the directed deviation model poisoning attack.



(a) Only Benign updates aggregated

(b) Only Malicious updates aggregated

Figure 2: ***The average flip-score of malicious and benign clients over time for a motivating experiment where a DNN is trained on MNIST for 500 iterations (80 benign, 20 malicious clients). In (a), only benign updates were aggregated using FEDSGD, and in (b), only malicious updates were aggregated, depicting the two extreme cases of federated learning in a malicious setting. These results show that when the global model update is benign, the malicious clients send gradients with high flip-score to deviate the model from convergence, but when the global model update itself was poisoned, the benign clients send high flip-score gradients for recovery whereas the malicious clients strive to maintain the direction of the already-poisoned model. Thus the intuition is too high flip-scores and too low flip-scores in a coordinated manner are red flags.***

In summary, TESSERACT makes the following contributions.

1. We use our simple intuition to detect malicious clients that attack federated learning using the SOTA attack model Fang et al. (2020). Our intuition is that certain patterns of flips of the signs

of gradients across multiple model parameters and across multiple clients are rare under benign conditions.

2. We use a stateful suspicion model to keep the history of every client's activity and use that as a weighting factor in the aggregation. We theoretically prove the convergence of TESSERACT that uses the weighted averaging and establish a convergence rate.
3. We evaluate TESSERACT on DNNs trained on MNIST and FEMNIST, ResNet-18 on CIFAR-10, and GRU on the Shakespeare dataset. We comparatively evaluate our defense against six baselines, including the most recent ones, FABA Xia et al. (2019), FoolsGold Fung et al. (2020), and FLTrust Cao et al. (2020) and show that TESSERACT remains robust even against an adaptive white-box attacker. While several of the existing defenses shine under specific configurations (combination of attacks and datasets/models), TESSERACT is the only one whose protection transfers well across configurations. We release the source code, the attack scripts, the trained models, and the test harness for the evaluation at the anonymized page https://www.dropbox.com/sh/h9ulw6y2f8rzv64/AADe1Sb9PhhCLqclzgZ4xvvJa?dl=0.

The rest of the paper is organized as follows. We describe in Sec 2 the threat model, the SOTA attack, and why all existing Byzantine-resilient federated learning approaches are susceptible. We present TESSERACT's design in Sec 3 and convergence analysis in Sec 4. We describe the baselines and the datasets in Sec 5, and evaluation in Sec 6.

2 BACKGROUND

Our simulation of federated learning consists of m clients, each with its own local data, but with the same model architecture and SGD optimizer, out of which c are malicious, as shown in Figure 1. The parameter server assumes c_{max} number of clients are malicious, and we have set $c = c_{max}$. The clients run one local iteration, send their gradients in unencrypted form to the server, which updates the global model for the clients to download in a synchronous manner.

2.1 BYZANTINE-RESILIENT FEDERATED LEARNING

Here we describe the leading defenses briefly, stymied by SOTA untargeted model poisoning attacks.

1. The simplest aggregation technique is **FedSGD** McMahan et al. (2017) that does a simple weighted mean aggregation of the gradients, weighted by the number of data samples each client holds. FedSGD can be attacked by a single malicious client that can send boosted gradients.
2. **Trimmed mean and Median** Yin et al. (2018) aggregate the parameters independently, where one trims c_{max} each at the lower and higher extremes of every parameter and the other takes the median of every parameter update across the gradients received from all the clients. The full-Trim attack Fang et al. (2020) is specifically designed toward these aggregations rules.
3. **Krum** Blanchard et al. (2017) selects one local model as the next global model based on what the majority $(m - c_{max} - 2)$ of the clients have the least Euclidean distance from. The full-Krum attack Fang et al. (2020) is tailored to attack Krum.
4. **Bulyan** Mhamdi et al. (2018) combines the above approaches by running Krum iteratively to select a given number of models, and then running Trimmed Mean over the selected ones. The Full-Trim attack is also transferable to Bulyan.
5. **FABA** Xia et al. (2019) iteratively filters out models farthest away from the mean of the remaining models, c_{max} number of times before returning the mean of the remaining gradients.
6. **FoolsGold** Fung et al. (2020) was motivated to defend against poisoning attacks by Sybil clones, and thus, it finds clients with similar cosine similarity to be malicious, penalizes their reputation, and returns a weighted mean of the gradients, weighed by their reputation.
7. **FLTrust** Cao et al. (2020) bootstraps trust by assuming that the server has access to a clean validation dataset, albeit small, and returns a weighted mean of the gradients weighed by this trust. In our setting, we do not see a realistic method to access such as clean dataset, especially considering the non-iid nature of the local datasets at the clients.

2.2 THREAT MODEL: STATE-OF-THE-ART MODEL POISONING ATTACK

Our threat model consists of a scenario where an adversary occupies a fraction (less than half) of all clients participating in the collaborative learning process, either by joining the system through multiple aliases, or by compromising other clients. We assume that the adversary also has access to the gradient vector sent by the benign clients to the server, and knows what aggregation the server

is running. We claim to defend against untargeted model poisoning attacks in the above described setting, as they invariably generate a large flip-score in their attempt to push the global model away from the optima. We focus on the SOTA untargeted model poisoning attack Fang et al. (2020)—a directed deviation attack (DDA) bypassing *all known defenses* [1]. We have also evaluated TESSERACT on label-flipping attacks in Appendix **??**. In order to disrupt training, Fang *et al.* have formulated their attack to change the local models on the compromised worker devices. This change is done strategically (through solving a constrained optimization problem) such that the global model deviates the most toward the *inverse of the direction* along which the before-attack global model would have changed. Their intuition is that the deviations accumulated over multiple iterations would make the learned global model differ from the pre-attack one significantly.

TESSERACT defends against both variants of the DDA, one specialized to poison Krum (transferable to Bulyan, we call this the *Full-Krum attack*), the other specialized for Trimmed Mean (transferable to Median, we call this the *Full-Trim attack*), respectively. We assume a full-knowledge (white-box) attack where the attackers have access to the current benign gradients. They themselves compute the benign gradients on their local data as well, and thus estimate the benign direction as the average of all benign gradients. This value is stored in a vector s of size equal to the number model parameters.

$$s(t,\cdot) = sign(\underset{i}{sum}(\triangledown LM_i(t,\cdot)),$$

where $\triangledown LM_i(t,\cdot)$ is the gradient updates of client i.

Full-Krum attack: Having estimated s, the attackers send gradients in the opposite direction, all with a magnitude λ, due to computational constraints, with some added noise to appear different, still at a small Euclidean distance from each other. The upper bound of λ is computed in every iteration as a function of $m, c, |P|, GM(t,\cdot), \triangledown LM_i(t+1,\cdot)$, where c out m participating clients are malicious, $|P|$ is the number of parameters in the global model GM. λ is then iteratively decreased until the attackers make sure (using a local estimate) that the parameter server would have chosen the attacked model, had it been using Krum.

Full-Trim attack: The Trim attack while following the same fundamental principle of flipping the gradient direction, attempts to skew the distribution of every parameter j toward the inverse of the direction that $s(t,j)$ suggests in order to attack a mean-like aggregation. It does so by randomly sampling gradient magnitudes from a range that has been computed by the attackers, guaranteed to skew the gradient distribution of every parameter, without appearing as obvious outliers (which would have been caught by a method such as Trimmed Mean). Therefore, the attacked gradients here look more diverse than those in the Full-Krum attack. This attack smartly takes advantage of the fact that none of the existing aggregation techniques looks at the change in gradient direction to identify malicious gradients in a robust way.

3 DESIGN

TESSERACT uses a reputation-based scheme to compute the aggregation weights of the participating clients. Reputation-based schemes have been a standard practice in the literature - Kang et al. (2019) and Zhang et al. (2012) make use of reputation score as an incentive mechanism for clients to remain in the system. Fung et al. (2020) and Awan et al. (2021) compute reputation score from the pairwise cosine similarity of the gradients between clients, Cao et al. (2021) does that by computing the cosine similarity of the client gradients with reference to trusted benign gradients on a root dataset in order to defend against poisoning attacks. We compute reputation-score in a different way by using flip-score of the local gradients, as described below. TESSERACT assumes that a maximum of $c_{max}(< \frac{m}{2})$ clients can be malicious. It penalizes $2c_{max}$ clients and rewards the rest in every iteration by an amount $\mathcal{W}(i,t)$ based on their flip-score (described below) and updates their reputation score, where i is the client ID and t is time. We present the pseudocode in Algorithm 1.

$$\mathcal{W}(i,t) = \begin{cases} -(1-\frac{2c_{max}}{m}), & if \quad penalized. \\ \frac{2c_{max}}{m}, & if \quad rewarded. \end{cases}$$

These values make sure that that the expectation of the reputation score of a client is zero if their flip-scores belong to a uniform random distribution (see Appendix **??**). Also, this makes recovery difficult as the penalty value is higher if $c_{max} < \frac{m}{2}$ and allows us to be conservative. The reputation

[1]There is no overlap between the authors of this current submission and of the SOTA attack Fang et al. (2020).

score of a client i is initialized and updated as follows -

$$RS(i,0) = 0.$$
$$RS(i,t) = \mu_d RS(i,t-1) + \mathcal{W}(i,t), \quad t > 0.$$

where $0 \leq \mu_d \leq 1.0$ is the decay parameter, and RS is the reputation score. A low μ_d gives more importance to the present flip-score and a high μ_d gives significant importance to the past performance of a client. This decay operation also helps cap the maximum and minimum reputation score (with $\mu_d < 1$, for more details, see Appendix **??**). We normalize the reputation score using softmax to do a weighted mean aggregation, but a user can use this to directly filter out the suspicious gradients and use an aggregation rule of one's choice. We halve the reputation score of every client if any of the values grows so large that the softmax operation causes an overflow, as the suspicion score is a cumulative quantity, and can potentially keep increasing with time if a client does not behave maliciously.

Flip-score. We compare the present gradients $\nabla LM_i(t+1,\cdot)$ sent by local model i with the gradient direction of the global model at time t, $s_g(t,\cdot) = sign(GM(t,\cdot) - GM(t-1,\cdot))$. We define flip-score as the sum of square of the gradient magnitudes of all parameters that experience a change in their gradient direction, that is,

$$FS_i(t+1) = \sum_{j=0}^{|P|-1} (\nabla LM_i(t+1,j))^2 (sign(\nabla LM_i(t+1,j)) \neq s_g(t,j)), \tag{1}$$

where $|P|$ is the total number of parameters in the model being trained. A low flip-score, thus suggests that the gradient updates are approximately in the same direction as the previous iteration, and a high flip-score suggests a deviation from the previous update – either a large number of parameters have flipped direction, or a small number of parameters have flipped direction with large magnitudes, or both. As observed in Figure 2, if the previous global update was benign, a malicious client will tend to have a high flip-score, but if the previous update itself was poisoned, the flip-score of benign clients will be high and those of malicious clients will be low. We, therefore penalize c_{max} number of clients on either end of the current flip-score distribution, as also done by Yin et al. (2018), but in the context of gradient values per parameter. This allows our system to reduce false negative detections, making it robust even if the previous global update was malicious. Since we allow redemption and make sure that in a random penalization scheme, the expectation of reputation score in a benign setting approaches zero, we do not unnecessarily penalize benign clients with low flip-score. Also, since we do not use any hard threshold for detecting attacks, and identify only the extreme ends of flip-score distribution in every iteration as malicious, we do not prevent low flip-score moves. At the time of convergence, when most of the clients will favor low flip-score moves, such moves will be allowed even after ignoring the gradients at the extreme ends.

Algorithm 1 Federated learning with TESSERACT

Output: Global model $GM(t+1,\cdot)$
Input: Local model updates $w = \nabla LM_i(t+1,\cdot)$
Parameters: m, c_{max}, μ_d

0 : Initialize reputation $RS_i(0) = 0$ for every client i
1 : Initialize global direction $s_g(0)$ to a zero vector
2 : for every client i **compute flip-score**:
3 : $\quad FS_i(t+1) = \sum_{j=0}^{|P|-1} (\nabla LM_i(t+1,j))^2 (sign(\nabla LM_i(t+1,j)) \neq s_g(t,j))$
4 : Penalize c_{max} clients on either end of FS spectrum as: $RS(i,t+1) = \mu_d RS(i,t) - (1 - \frac{2c_{max}}{m})$
6 : Reward the rest of the clients as: $RS(i,t+1) = \mu_d RS(i,t) + \frac{2c_{max}}{m}$
7 : Normalize reputation weights: $W_R = \frac{e^{RS}}{\sum e^{RS}}$
8 : Aggregate gradients: $\nabla GM(t+1,\cdot) = w^T W_R$
9 : Update global direction: $s_g(t+1,\cdot) = sign(\nabla GM(t+1,\cdot))$
10: Update global model and broadcast: $GM(t+1,\cdot) = GM(t,\cdot) + \nabla GM(t+1,\cdot)$

4 CONVERGENCE ANALYSIS

We make the following assumptions on LM_k, part of which are adapted from Li et al. (2020).

1. Assumption #1: LM_k are all L-smooth, that is, for all v and w, $LM_k(v) \leq LM_k(w) + (v - w)^T \nabla LM_k(w) + \frac{L}{2}\|v - w\|_2^2$.
2. Assumption #2: LM_k are all μ-strongly convex, that is, for all v and w, $LM_k(v) \geq LM_k(w) + (v - w)^T \nabla LM_k(w) + \frac{\mu}{2}\|v - w\|_2^2$.
3. Assumption #3: Let ξ_t^k be sampled uniformly at random from the local data of the $k - th$ client, then the variance of stochastic gradients of each client is bounded, that is, $\mathbb{E}\,\|\nabla LM_k(\mathbf{w}_t^k, \xi_t^k) - \nabla LM_k(\mathbf{w}_t^k)\|^2 \leq \sigma_k^2$ for $k = 1, 2, ...m$.
4. Assumption #4: The expected squared norm of stochastic gradients is uniformly bounded, that is, $\mathbb{E}\,\|\nabla LM_k(\mathbf{w}_t^k, \xi_t^k)\|^2 \leq G^2$ for $k = 1, ...m$, and $t = 0, ..T - 1$.
5. Assumption #5: Within K iterations, the reputation score of malicious clients drop at least by δ_{mal}, and reputation score of benign clients increase at least by δ_{ben}, that is, $|RS_{mal}^t - RS_{mal}^{t-K}| \geq \delta_{mal}$ and $|RS_{ben}^t - RS_{ben}^{t-K}| \geq \delta_{ben}$, for $t = 0, ..T - 1$.

Assumptions #1 and #2 are standard and apply to l_2-norm regularized linear regression, logistic regression, and softmax classifier. Assumptions #3 and #4 have been also made by Zhang et al. (2012); Stich (2018); Stich et al. (2018); Yu et al. (2019). In our problem setting, Assumption #3 claims that the gradient with a subset of local data is bounded from the gradient with whole batch for both malicious and benign clients. Assumption #5 claims that after every K iterations, our algorithm will have better ability to distinguish the malicious clients.
Following the Theorem 1 in Li et al. (2020), after simplifying for our case in which the clients communicate with the parameter server in every iteration, we derive this convergence (details in Appendix). Let GM^* and LM_k^* be the minimum value of GM and LM_k respectively, then:

$$\mathbb{E}[GM(\mathbf{w})_T] - GM^* \leq \frac{2}{\mu^2} \cdot \frac{L}{\gamma + T}(\sum_{k=1}^{m} p_k^2 \sigma_k^2 + 6L\Gamma + 8G^2 + 8G^2 \sum_{k=1}^{c} p_{k,0} + \frac{\mu^2}{4}\|w_0 - w^*\|^2).$$

where L, μ, G are defined above, $\gamma = max\{8\frac{L}{\mu}, 1\}$, $\eta_t = \frac{2}{\mu(\gamma+t)}$, T is the total number of iterations, c is the number of malicious clients, $p_{k,0}$ is the initial weight for malicious clients, and $\Gamma = GM^* - \sum_{k=1}^{m} p_k LM_k^*$, that effectively quantifies the degree of non-iidness according to Li et al. (2020). This shows that a weighted mean aggregation is guaranteed to converge in federated learning. The converge speed is $O(\frac{1}{T})$. The increase of number of malicious clients c increases the bound.

5 Implementation

We have simulated the federated learning on a single machine with a Tesla P100 PCIe GPU with 16GB memory, using PyTorch, with as many clients as can be handled by our machine. *The data was distributed with a non-IID bias of 0.5 (default), except for Shakespeare where the data was distributed sequentially among the clients and FEMNIST where the data was distributed by the writer.* In our simulation, all clients run one local iteration on a batch of its local data before communicating with the parameter server in a synchronous manner. The clients sample their local data in a round-robin manner, send their local gradients to the parameter server, and download the updated global model before running the next local iteration. The malicious clients attack every iteration of training, we assume $c_{max} = c$.

Baselines and datasets. The baseline aggregation rules used are Krum, Bulyan, Trimmed Mean, and Median. We also compare Tesseract with the recent defense techniques of FABA Xia et al. (2019), FoolsGold Fung et al. (2020), and FLTrust Cao et al. (2020). We evaluate Tesseract on 4 different datasets (Table 1). The DNN trained on MNIST has 2 conv layers with 30 and 50 channels respectively, each followed by a 2×2 maxpool layer, then by a fully connected layer of size 200, and an output layer of size 10. We use a constant learning rate, except for CIFAR-10 where we start with zero, reach the peak at one-fourth of the total iterations, and slowly get down to zero again. The CNN trained on the FEMNIST dataset follows the same network architecture as Caldas et al. (2019).

6 Evaluation

6.1 Macro Experiments

In Table 2, we compare the test accuracy achieved by various aggregation techniques in benign and malicious conditions. For Shakespeare, which is an NLP dataset, we report the test loss. The final reported test loss value does not capture the training dynamics, which can be observed in Figure 3 for the more damaging Full-Krum attack. We have not shown the test loss curve for Krum aggregation because of the large loss values here, but have pushed the results to Appendix A.3 in Figure **?**. We

Table 1: Datasets with the number of classes (n_c) and training samples (n_s), the models and model parameters (P), training rounds (n_r), batch_size (b), learning rate (lr), total number of clients (m), number of malicious ones (c), and decay parameter (μ) used in* TESSERACT. ** variable learning rate, peaks at 0.1.

Dataset	n_c	n_s	Model	P	n_r	b	lr	m	c	μ
MNIST	10	60k	DNN	0.27M	500	32	0.01	100	20	0.99
CIFAR-10	10	50k	ResNet-18	5.2M	2000	128	0.1^*	10	2	0.99
Shakespeare	100	-	GRU	0.14M	2000	100	0.01	10	2	0.99
FEMNIST	62	805k	DNN	6.6M	2000	32	0.1	35	7	0.99

Table 2: Attack impact - Test accuracy for tailored attacks (Full-Krum; Full-Trim), on different datasets with $c/m = 0.2$. For the Shakespeare dataset, test loss has been reported. We verify the damaging impact of the Full-Trim attack on mean-like aggregations (FedSGD, Trimmed mean, Median) and Full-Krum on Krum-like aggregations (Krum, Bulyan). We also observe that the existing defenses—FABA, FoolsGold, and FLTrust—seem to defend against this attack in some cases, and fail in others, whereas* TESSERACT *consistently shines in all cases.

Attack	**Defense**	**Test accuracy (%) / Test loss (only for Shakespeare)**			
		MNIST+ DNN	CIFAR-10+ ResNet-18	Shakespeare+ GRU	FEMNIST+ DNN
None	FedSGD	92.45	**71.17**	**1.62**	83.60
	TESSERACT	**92.52**	66.92	1.64	83.58
	FABA	91.77	69.94	1.76	82.69
	FoolsGold	91.20	70.71	1.63	**83.80**
	FLTrust	87.70	68.08	**1.62**	82.72
Full-Krum	FedSGD	82.97	39.68	1.62	29.87
	Krum	8.92	9.81	11.98	5.62
	Bulyan	10.14	13.24	9.23	9.91
	TESSERACT	**87.73**	61.26	1.64	**80.19**
	FABA	86.99	55.96	1.75	55.61
	FoolsGold	47.12	42.28	**1.63**	0.07
	FLTrust	82.50	**65.25**	1.67	79.53
Full-Trim	FedSGD	65.25	47.32	1.74	32.34
	Trim	36.36	55.25	3.28	13.03
	Median	28.37	50.54	3.30	45.6
	TESSERACT	90.55	67.65	1.66	82.51
	FABA	**91.84**	67.31	**1.64**	79.66
	FoolsGold	91.61	**69.24**	1.66	**83.09**
	FLTrust	34.20	64.23	1.68	79.28

see that TESSERACT is the winner or 2nd place finisher in 7 of the 12 cells (benign + two attacks $\times$ 4 datasets). This on the surface appears to be not very promising, till one looks deeper. The baseline protocols that finish first in one configuration fare disastrously in other configurations, indicating that they are tailored to specific attacks or datasets (whether by conscious design or as an artefact of their design). For example, FoolsGold does creditably for the Full-Trim attack but is vulnerable against the Full-Krum attack. Averaging across the configurations, it appears FABA is the closest competitor to TESSERACT. We observe that FABA, although it performed well for $c/m = 0.2$, failed to defend when the number of attackers grew to $c/m = 0.3$, evident from the results in Figure 3(b). We have used F-MNIST, Ch-MNIST, and Breast cancer Wisconsin Dataset here to show the effect on diverse datasets. As the number of attackers increases, the mean starts to shift more toward them. One false positive detection by FABA can cause it to trim out a benign local update, which causes the mean to shift further toward the malicious updates iteratively, and fail. On the other hand, TESSERACT is guaranteed to defend against the attack as long as $c_{max} \geq c$ and $c < \frac{m}{2}$. We find empirically (result not shown) that FABA degrades fast, and much faster than TESSERACT, when the fraction of malicious clients increases. Where FABA fails at $c/m = 0.3$, we show in Figure **??**(a) that TESSERACT remains stable until $c/m = 0.45$. We also show in Figures **??**(b) and (c) that TESSERACT remains robust across a wide range of non-IID bias, that is 0.1 to 0.8. TESSERACT and FABA require an estimate of the upper bound of number of malicious clients, c_{max} to be known.

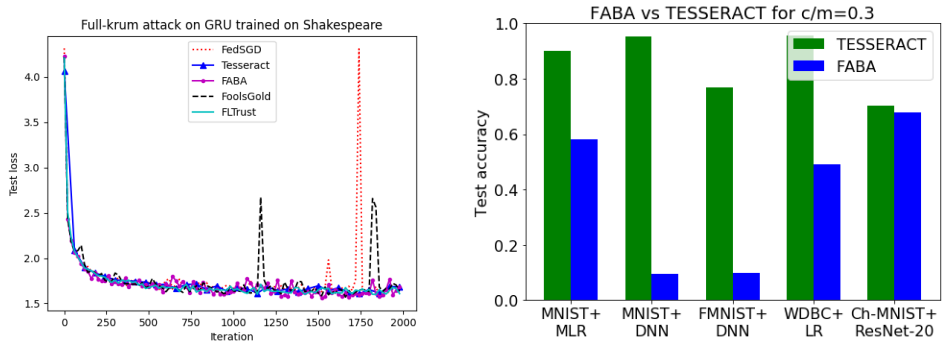


Figure 3: Left shows the test loss curve comparing* TESSERACT *with the benchmark aggregation algorithms against the Full-Krum attack. We see that the attack generates sporadic spikes in training, best handled by* TESSERACT*, evident from its smooth test loss curve. Right shows the comparison of FABA and* TESSERACT *across diverse datasets for $c/m = 0.3$. FABA begins to fail with a higher fraction of malicious clients, while* TESSERACT *remains robust.

Table 3: Fraction of malicious or benign clients allotted weights above 1e-4 averaged over 500 iterations

Defense	Mal/ Ben	Benign	Full-trim	Full-krum
FoolsGold	n_{ben}	0.29	0.30	0.10
	n_{mal}	-	0.00	0.64
FLTrust	n_{ben}	0.48	0.45	0.49
	n_{mal}	-	0.52	0.63
TESSERACT	n_{ben}	0.75	0.75	0.63
	n_{mal}	-	0.00	0.08

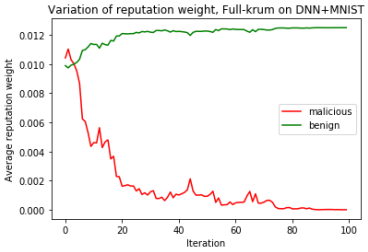


Figure 4: Clients' reputation weights against time with* TESSERACT*, with higher weights to benign clients, as malicious ones tend to 0.

FoolsGold and FLTrust, on the other hand, make use of cosine similarity among clients, and with a trusted cross-check dataset at the server, respectively, to identify suspicious clients. We have found that both of these techniques unnecessarily penalize many benign clients and assign them a zero weight in order to conservatively defend against an attack, as can be seen in Table 3. This can have a significantly negative impact on a practical system where one wishes to learn from data that the different clients hold locally. On the other hand, TESSERACT allows all clients to contribute to the global model update that do not have a large negative reputation score. Figure 4 shows the evolution of the average reputation score of malicious and benign clients. When benign and malicious clients start from the same region, the weights of malicious clients decrease with time tending to zero, while those of the benign clients increase with time.

6.2 ADAPTIVE ATTACK

Having shown the performance of TESSERACT against the above attacks, we proceed to analyze an adaptive attack scenario, *i.e.*, one where the attacker has full knowledge of TESSERACT, including the dynamic value of the cutoff flip-scores. Thus, at iteration $t+1$, the attacker knows $FS_{low}(t)$ and $FS_{high}(t)$ beyond which the clients were penalized at iteration t. The Adaptive-Krum attack first computes the target malicious gradients at $t+1$, and if its flip-score goes above $FS_{high}(t)$, it undoes the attack on its parameters that would have had low magnitude updates (without attack), by replacing 5% of the attacked parameters at a time with their benign values until the flip-score is brought down to ensure stealth. Since the stealthy attack will be less powerful than the original intended attack, the global model will only be partially poisoned, and the attackers are not expected to occupy the lower spectrum of the flip-score. This has been verified in our experiments as well. All the malicious clients send these attacked parameters with some added randomness in order to support each other. We observe a trade-off between stealth and attack impact in this case, as can be seen in Figure 5. The Adaptive-Trim attack is a smarter and collaborative attack. Its target is to

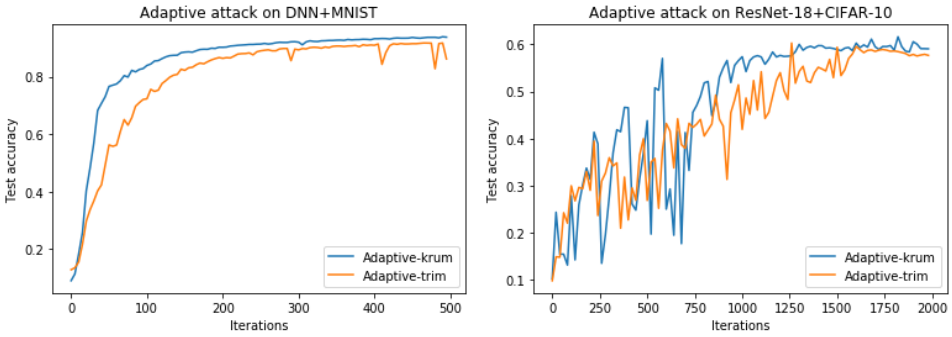


Figure 5: Performance of* TESSERACT *against adaptive whitebox attacks, specifically designed to attack* TESSERACT*, evaluated on MNIST and CIFAR-10. We observe a significant improvement in test accuracy, compared to the base case impact of Full-Krum on Krum and Full-Trim on FedSGD, as reported in Table 2.

generate attacked gradients v_i, $i = 1, 2, ..c-1$. It first computes the Full-Trim target attack u_i for every malicious client i. Then it initializes v_0 to u_0 and modifies it until v_0 generates a flip-score that is less than $FS_{high}(t)$, Client $i = 1$ updates its target attack from u_1 to $v_1 = u_1 + (u_0 - v_0)$ in order to compensate for a sub-optimal attack created by $i = 0$ because of the flip-score-evasion constraint. Thus, a client may not necessarily find a solution if the target grows and the constraints are hard to solve for. The attacker hopes that $\sum_{i=0}^{c-1} u_i = \sum_{i=0}^{c-1} v_i$. We have also modified the above constraint with a weighted sum weighed by the actual reputation score and evaluated it in Figure **??** in Appendix **??**. The attack is defined formally in Appendix **??**. The performance of TESSERACT against this attack is shown in Figure 5. We find that the adaptive attacks are not very effective against TESSERACT. The benign accuracy for the two datasets are 92.45% and 71.17%. This happens due to multiple reasons: 1) the attack loses in strength while trying to gain in stealth, 2) the attackers need not be allotted equal reputation weights, so the weighted sum of the attacked gradients v do not match with the weighted sum of the target attack u, 3) the flip-score distribution is dynamic, as can be seen in Figure 2, and changes from time t to $t + 1$, and when it decreases in consecutive iterations by a significant amount, the attackers can still be penalized and blocked from attacking.

7 DISCUSSION AND CONCLUSION

We have presented TESSERACT, a secure parameter server for federated learning. TESSERACT uses a stateful algorithm to allocate reputation scores to the participating clients to lower the contribution of maliciously behaving clients. We define malicious behavior using a metric, flip-score, which when very high or low, captures attacks that try to divert the global model away from convergence. Even if the attack succeeds in one iteration, we block it in the subsequent iterations to expedite recovery, by also flagging the low flip-score values. This makes TESSERACT a robust defense, no matter when it is instantiated, although we recommend enabling TESSERACT right from the start. TESSERACT can also be used to just filter out clients based on their reputation, so that an aggregation of the user's choice can be used. We also add the fairness attribute to TESSERACT by allowing clients to redeem themselves and contribute to the global model if they act benign. This is done using a user-defined $decay$ parameter to control the speed of redemption. All of the above evaluation is limited to a synchronous setting with no gradient encryption, as gradient encryption is less relevant in a cross-device scenario due to its high computational overhead Zhang et al. (2020). We evaluate the benefits of TESSERACT compared to the fundamental FL aggregation FedSGD and state-of-the-art defenses, namely Krum, Bulyan, FABA, FoolsGold, and FLTrust. We evaluate using full knowledge untargeted model poisoning attacks that have recently been found to be most damaging against FL. We find that different existing defenses shine under specific combinations of attacks and datasets/models. However, TESSERACT provides transferable defense with accuracy competitive with individual winners under all configurations. Further, TESSERACT holds up better than its closest competitor, FABA, when the fraction of malicious clients increases (beyond 20%). Finally, an adaptive white-box attacker with access to all internals of TESSERACT, including dynamically determined threshold parameters, cannot bypass its defense.

8 REPRODUCIBILITY STATEMENT

All the details for the datasets and models used along with the implementation of the adaptive attack is given in Section 5. To help with the reproducibility of the experiments, we make available all of our code, the trained models, the training hyperparameter values, and the raw results from our testing. These are all to be found at the anonymized link https://www.dropbox.com/sh/h9ulw6y2f8rzv64/AADe1Sb9PhhCLqclzgZ4xvvJa?dl=0. The code has also been uploaded as a supplementary file in the current submission.

REFERENCES

Sana Awan, Bo Luo, and Fengjun Li. Contra: Defending against poisoning attacks in federated learning. In *European Symposium on Research in Computer Security*, pp. 455–475. Springer, 2021.

Peva Blanchard, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems*, pp. 119–129, 2017. Krum paper.

Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečný, H. Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2019.

Xiaoyu Cao, Minghong Fang, Jia Liu, and Neil Zhenqiang Gong. Fltrust: Byzantine-robust federated learning via trust bootstrapping. *arXiv preprint arXiv:2012.13995*, 2020.

Xiaoyu Cao, Minghong Fang, Jia Liu, and Neil Zhenqiang Gong. Fltrust: Byzantine-robust federated learning via trust bootstrapping. *Network and Distributed System Security Symposium*, pp. 1–18, 2021.

Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Local model poisoning attacks to byzantine-robust federated learning. In *29th USENIX Security Symposium (USENIX Security 20)*, Boston, MA, August 2020. URL https://www.usenix.org/conference/usenixsecurity20/presentation/fang.

Clement Fung, Chris J. M. Yoon, and Ivan Beschastnikh. The limitations of federated learning in sybil settings. In *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*, pp. 301–316, San Sebastian, October 2020. USENIX Association. ISBN 978-1-939133-18-2. URL https://www.usenix.org/conference/raid2020/presentation/fung.

Jiawen Kang, Zehui Xiong, Dusit Niyato, Shengli Xie, and Junshan Zhang. Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory. *IEEE Internet of Things Journal*, 6(6):10700–10714, 2019.

Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HJxNAnVtDS.

H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, 2017.

El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Rouault. The hidden vulnerability of distributed learning in byzantium. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pp. 3521–3530, 2018. Bulyan paper.

Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. In *Advances in Neural Information Processing Systems*, pp. 4424–4434, 2017.

Sebastian U Stich. Local sgd converges fast and communicates little. *arXiv preprint arXiv:1805.09767*, 2018.

Sebastian U Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified sgd with memory. *arXiv preprint arXiv:1809.07599*, 2018.

Qi Xia, Zeyi Tao, Zijiang Hao, and Qun Li. Faba: An algorithm for fast aggregation against byzantine attacks in distributed neural networks. In *IJCAI*, pp. 4824–4830, 2019.

Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.

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

Hao Yu, Sen Yang, and Shenghuo Zhu. Parallel restarted sgd with faster convergence and less communication: Demystifying why model averaging works for deep learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 5693–5700, 2019.

Chengliang Zhang, Suyi Li, Junzhe Xia, Wei Wang, Feng Yan, and Yang Liu. Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning. In *2020 {USENIX} Annual Technical Conference ({USENIX}{ATC} 20)*, pp. 493–506, 2020.

Yuchen Zhang, John C Duchi, and Martin J Wainwright. Communication-efficient algorithms for statistical optimization. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pp. 6792–6792. IEEE, 2012.