To appear in the IEEE Symposium on Security & Privacy, 2022

Back to the Drawing Board: A Critical Evaluation of Poisoning Attacks on Production Federated Learning

Virat Shejwalkar*, Amir Houmansadr*, Peter Kairouz†, Daniel Ramage†
*University of Massachusetts Amherst †Google Research
*{vshejwalkar, amir}@cs.umass.edu, †{kairouz, dramage}@google.com

***Abstract*—While recent works have indicated that federated learning (FL) may be vulnerable to poisoning attacks by compromised clients, their real impact on production FL systems is not fully understood. In this work, we aim to develop a comprehensive systemization for poisoning attacks on FL by enumerating all possible threat models, variations of poisoning, and adversary capabilities. We specifically put our focus on untargeted poisoning attacks, as we argue that they are significantly relevant to production FL deployments.**

We present a critical analysis of untargeted poisoning attacks under practical, production FL environments by carefully characterizing the set of realistic threat models and adversarial capabilities. Our findings are rather surprising: contrary to the established belief, we show that FL is highly robust in practice even when using simple, low-cost defenses. We go even further and propose novel, state-of-the-art data and model poisoning attacks, and show via an extensive set of experiments across three benchmark datasets how (in)effective poisoning attacks are in the presence of simple defense mechanisms. We aim to correct previous misconceptions and offer concrete guidelines to conduct more accurate (and more realistic) research on this topic.

I. Introduction

Federated learning (FL) is an emerging learning paradigm in which data owners (called *clients*) collaborate in training a common machine learning model without sharing their private training data. In this setting, a central *server* (e.g., a service provider) repeatedly collects some updates that the clients compute using their local private data, aggregates the clients' updates using an *aggregation rule* (AGR), and finally uses the aggregated client updates to tune the jointly trained model (called the *global model*), which is broadcasted to a subset of the clients at the end of each FL training round. FL is increasingly adopted by various distributed platforms, in particular by Google's Gboard [1] for next word prediction, by Apple's Siri [49] for automatic speech recognition, and by WeBank [62] for credit risk predictions.

The threat of poisoning FL: A key feature that makes FL highly attractive in practice is that it allows training models in collaboration between *mutually untrusted* clients, e.g., Android users or competing banks. Unfortunately, this makes FL susceptible to a threat known as *poisoning*: a small fraction of FL clients, called *compromised clients*, who are either owned or controlled by an adversary, may act maliciously during the FL training process in order to corrupt the jointly trained global model. Specifically, the goal of the poisoning adversary is to attack FL by instructing its compromised clients to contribute *poisoned* model updates during FL training in order to *poison* the global model.

There are three major approaches to poisoning FL: *targeted*, *backdoor*, and *untargeted* poisoning; Figure 1 briefly illustrates them. The goal of this work is to understand the significance of poisoning attacks to *production FL systems* [11], [32], and to reevaluate the need for sophisticated techniques to defend against FL poisoning. **We choose to focus on untargeted FL poisoning** as we find it to be significantly relevant to production deployments: it can be used to impact a large population of FL clients and it can remain undetected for long duration. As we focus on untargeted poisoning, in the rest of this paper "poisoning" refers to untargeted poisoning, unless specified otherwise.

The literature on FL poisoning attacks and defenses: Recent works have presented various techniques (Section IV-A) to poison FL [5], [23], [55]. Their core idea is to generate poisoned updates (either by direct manipulation of model updates, called *model poisoning* [3], [5], [7], [23], [55], or through fabricating poisoned data, called *data poisoning* [59], [61]) that deviate maximally from a benign direction, e.g., the average of benign clients' updates, and at the same time *circumvent* the given robust AGR, i.e., by bypassing its detection criteria. To protect FL against such poisoning attacks, the literature has designed various *robust* aggregation rules [10], [15], [23], [41], [55], [70] (Section II-B) which aim to remove or attenuate the updates that are more likely to be malicious according to some criterion.

The gap between the literature and practice: The existing literature on poisoning attacks and defenses for FL makes *unrealistic assumptions* that do not hold in real-world FL deployments, e.g., assumptions about the percentages of compromised clients, total number of FL clients, and the types of FL systems [32]. For instance, state-of-the-art attacks [5], [23], [55] (defenses [10], [16], [68], [70]) assume adversaries who can compromise up to 25% (50%) of FL clients. For an app like Gboard with $\sim 1B$ installations [32], 25% compromised clients would mean an attacker controls *250 million Android*

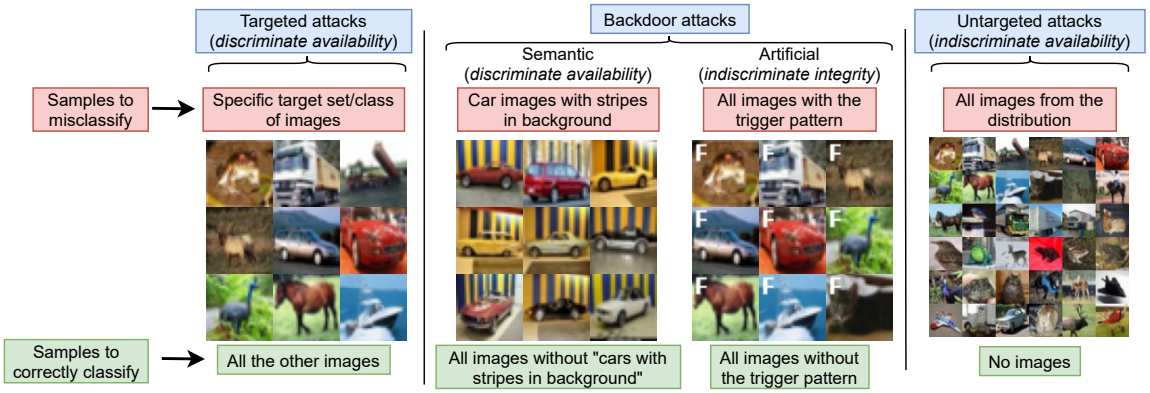


Figure 1: Classes of FL poisoning attacks and their objectives defined using the taxonomy in Section III-A1: *Targeted* attacks [7], [58] aim to misclassify only a specific set/classes of inputs (e.g., certain 10 samples from CIFAR10), *semantic backdoor* attacks [3], [61] aim to misclassify inputs with specific properties (e.g., cars with stripes in background), *artificial backdoor* attacks [67] aim to misclassify inputs with an artificial (visible or invisible) trigger pattern (e.g., shape of letter "F"), and *untargeted* attacks [23], [55] aim to reduce model accuracy on *arbitrary* inputs (e.g., the entire CIFAR10 distribution).

devices! We argue that, although interesting from theoretical perspectives, the assumptions in recent FL robustness works do not represent common real-world adversarial scenarios that account for the difficulty and cost of at-scale compromises.

Our contributions: In this work, we perform a critical analysis of the literature on FL robustness against (untargeted) poisoning under practical considerations. Our ultimate goal is to understand the significance of poisoning attacks and the need for sophisticated robust FL algorithms in production FL. Specifically, we make the following key contributions:

I. Systemization of FL poisoning threat models. We start by establishing a comprehensive systemization of threat models of FL poisoning. Specifically, we discuss three key dimensions of the poisoning threat to FL: The adversary's objective, knowledge, and capability. We discuss the practicality of all possible threat models obtained by combining these dimensions. As we will discuss, out of all possible combinations, only two threat models, i.e., *nobox offline data poisoning* and *whitebox online model poisoning*, are of practical value to production FL. We believe that prior works [5], [10], [23], [55] have neglected the crucial constraints of production FL systems on the parameters relevant to FL robustness. *Our work is the first to consider production FL environments* [11], [32] and provide practical ranges for various parameters of poisoning threat models. As a result, *our evaluations lead to conclusions that contradict the common beliefs in the literature*, e.g., we show that production FL even with the non-robust Average AGR is significantly more robust than previously thought.

II. Introducing improved poisoning attacks. First, we overview all of the existing untargeted poisoning attacks on FL that consider the two aforementioned threat models (Section IV-A). Then, we design improved attacks for these threat models. ***1) Our improved data poisoning attacks:*** We present *the first* attacks that systematically consider the data poisoning threat model for FL (Section IV-B2). We build on the classic label flipping data poisoning attack [45], [64], [65] designed for centralized ML. Our data poisoning attacks rely on the observation that increasing the amount of label flipped data increases the loss and norm of the resulting updates, and therefore, can produce poisoned updates that can effectively reduce the global model's accuracy. However, using arbitrarily large amounts of label flipped data may result in updates that cannot circumvent the robustness criterion of the target AGR. Hence, to circumvent the target AGR, we propose to adjust the amount of label flipped data used. ***2) Our improved model poisoning attacks:*** We propose novel model poisoning attacks that outperform the state-of-the-art. Our attacks (Section IV-B3) use *gradient ascent* to fine-tune the global model and increase its loss on benign data. Then, they adjust the L_2-norm of the corresponding poisoned update in order to circumvent robustness criterion of the target AGR.

III. Analysis of FL robustness in practice. We extensively evaluate all existing poisoning attacks as well as our own improved attacks across three benchmark datasets, for various FL parameters, and for different types of FL deployments. We make several significant deductions about the state of FL poisoning literature for production FL. For production cross-device FL, which contains thousands to billions of clients, following are our key lessons:

(1) *For practical percentages of compromised clients (M), even the most basic, non-robust FL algorithm, i.e., Average AGR, converges with high accuracy, i.e., it is highly robust.* For instance, data poisoning with $M = 0.1\%$ (model poisoning with $M = 0.01\%$) reduces the global model accuracy of FEMNIST from 83.4% to 81.4% (73.4%), CIFAR10 from 86.6% to 85.1% (82.9%), and Purchase from 85.4% to 85.3% (76.4%). These findings directly contradict the claims of previous works [10], [41], [68] that Average AGR cannot converge even with a *single* compromised client.

(2) *Poisoning attacks have no impact on existing robust FL algorithms even with impractically high M's*: At M=1%, data or model poisoning attacks reduce the accuracy by only <1% for most of the settings.

(3) *Enforcing a limit on the size of the dataset contributed by each client can act as a highly effective (yet simple) defense*

against data poisoning attacks with no need to any of the state-of-the-art, sophisticated robust FL aggregation algorithms.

(4) While recent works have introduced sophisticated and theoretically robust AGRs that incur high computation and memory costs, *the simple and low-cost defenses, e.g., norm-bounding [58], provide an equivalent protection to FL against state-of-the-art poisoning attacks.*

For production cross-silo FL, which contains up to hundred clients [32], we show that *data poisoning attacks are completely ineffective, even against non-robust Average AGR*. We also argue that model poisoning attacks are unlikely to play a major risk to production cross-silo FL, where the clients involved are bound by contract and their software stacks are professionally maintained (e.g., in banks, hospitals, etc.).

Implications of our study: Numerous recent works have proposed sophisticated aggregation rules for FL with strong theoretical robustness guarantees [2], [10], [20], [21], [41], [50], [68], [70]. However, our work shows that, when it comes to production FL deployments, even simple, low-cost defenses can effectively protect FL against poisoning. We also believe that our systematization of practical poisoning threat models can steer the community towards practically significant research problems in FL robustness.

II. Background

A. Federated Learning (FL)

In FL [32], [33], [40], a service provider, called *server*, trains a *global model*, θ^g, on the private data of multiple collaborating clients without directly collecting their data. In the t^{th} FL round, the server selects n out of total N clients and shares the most recent global model, i.e., θ_g^t, with them. Then, a client k uses their local data D_k to fine-tune θ_g^t using stochastic gradient descent (SGD) for a fixed number of local epochs E and obtains updated model by θ_k^t. Then, the k^{th} client computes her FL *update* as the difference $\nabla_k^t = \theta_k^t - \theta_g^t$ and shares ∇_k^t with the server. The server then computes an aggregate of all client updates using some aggregation rule, f_{agg}, i.e., using $\nabla_{\mathsf{agg}}^t = f_{\mathsf{agr}}(\nabla_{\{k\in[n]\}}^t)$. Then, the server updates the global model of the $(t+1)^{th}$ round using SGD as $\theta_g^{t+1} \leftarrow \theta_g^t + \eta\nabla_{\mathsf{agg}}^t$; here η is the server's learning rate. Section III-B1 discusses salient features of production FL.

B. Existing Defenses Against Untargeted Poisoning

As we focus on untargeted poisoning, below we discuss defenses against untargeted poisoning in detail and defer the discussion for targeted/backdoor poisoning to Appendix A.

The literature has presented various directions towards making FL robust against Byzantine or compromised clients. Note that, from detection perspective, there is no difference between untargeted attacks (adversary deliberately corrupts model updates) and Byzantine failures (arbitrary system failures corrupt updates). The core approach is to replace FL's vanilla average aggregation rule (AGR) [40] with a **robust AGR** (also called a *defense*). Below, we introduce the types of robust AGRs designed to defend FL against untargeted poisoning attacks.

Dimension-wise filtering defenses separately filter potentially malicious values for each dimension of clients' updates. Example AGRs are Median [70], Trimmed-mean [70], and sign-SGD with majority voting [6].

Vector-wise filtering defenses aim at removing potentially poisoned client updates. They differ from dimension-wise filtering, as they attempt to remove entire malicious updates, as opposed to removing malicious values. Example AGRs include RFA [50], RSA [36], Krum [10], Multi-krum [10], Bulyan [41], and Divide-and-conquer (DnC) [55].

Vector-wise scaling defenses, e.g., Norm-bounding [58], reduce the impact of poisoned updates by scaling their norms.

Certified defenses [14], [66] provide certified accuracy for each test input when the number of compromised clients or perturbation to the test sample is below a certified threshold.

Knowledge transfer based defenses [15], [38] aim to reduce the dimensionality of the client updates, because theoretical robustness guaranty of most of robust AGRs is directly proportional to updates' dimensionality. Hence, they use knowledge transfer and, instead of sharing parameters of client models, share predictions of client models on some public data.

Personalization techniques, e.g. Ditto [37] and EWC [71], fine-tune the potentially corrupt global model on each client's private data to improve its performance for the client.

C. Defenses We Evaluate in Our Work

For a robust AGR to be usable in production FL, it needs to provide *high performing models at low compute and memory costs*. In Table I, we compare the performance and overhead implications of state-of-the-art defenses overviewed above. Each red cell demonstrates a hindrance to adoption in production FL. In particular, (1) SignSGD + majority voting, Krum, Bulyan, RSA, and certified defenses incur significant performance losses, (2) certified defenses incur high memory cost to clients, (3) knowledge transfer based defenses require public data and incur high performance losses in cross-device, non-iid FL settings, and (4) personalization techniques cannot improve performance if the global model is significantly corrupt. Hence, to be effective, they should be coupled with a robust AGR and rely completely on the robustness of the AGR against poisoning. Therefore, the evaluation of personalization techniques is orthogonal to our evaluation of robust AGRs.

As the focus of our work is production FL, for brevity and space limitations, we only choose representative AGRs (bold in Table I) from each class that offer practical performance and overheads. Below we introduce the selected defenses in detail. Note that more sophisticated (e.g., higher overhead) defenses may provide better robustness to poisoning, however this does not impact our main conclusions: we will show that even such simple defenses are enough to protect production FL.

1) Average: In non-adversarial FL settings, dimension-wise Average [40] is an effective AGR. Due to its efficiency, Average is the only AGR implemented by FL applications in practice [1], [39], [49], [62].

2) Norm-bounding: This AGR [58] bounds the L_2 norm of all submitted client updates to a fixed threshold, with the

Table I: Comparing state-of-the-art aggregation rules (AGRs) in terms of accuracy, computation/memory cost, and theoretical guarantees. We show results for CIFAR10 with 1,000 clients. Red cells show limitations of the corresponding AGR.

Type of aggregation rule (AGR)	Example AGR	Accuracy in non-iid FL	Computation at server	Memory cost to client	Theoretical robustness based on
Non-robust	**Average** [40]	86.6	$\mathcal{O}(d)$	$\mathcal{O}(d)$	None
Dimension-wise filtering	Median [70]	84.2	$\mathcal{O}(dn\log n)$	$\mathcal{O}(d)$	convergence
	Trimmed-mean [70]	86.6	$\mathcal{O}(dn\log n)$		convergence
	Sign-SGD + majority voting [6]	35.1	$\mathcal{O}(d)$		convergence
Vector-wise scaling	**Norm-bound** [58]	86.6	$\mathcal{O}(d)$	$\mathcal{O}(d)$	Not established
Vector-wise filtering	Krum [10]	46.9	$\mathcal{O}(dn^2)$	$\mathcal{O}(d)$	convergence
	Multi-krum [10]	86.2	$\mathcal{O}(dn^2)$		convergence
	Bulyan [41]	81.1	$\mathcal{O}(dn^2)$		convergence
	RFA [50]	84.6	$\mathcal{O}(dn^2)$		convergence
	RSA [36]	35.6	$\mathcal{O}(d)$		convergence
	DnC [55]	86.1	$\mathcal{O}(d)$		filtering
Certification	Emsemble [14]	74.2	$\mathcal{O}(d)$	$\mathcal{O}(Md)$	Certification
	CRFL [66]	64.1			Certification
Knowledge transfer	Cronus [15]	Needs public data	$\mathcal{O}(d)$	$\mathcal{O}(d)$	filtering
Personalization	Ditto [37]	86.6	$\mathcal{O}(d)$	$\mathcal{O}(d)$	None (depends on server's AGR)
	EWC [71]				

intuition that the effective poisoned updates should have high norms. For a threshold τ and an update ∇, if the norm, $\|\nabla\|_2 > \tau$, ∇ is scaled by $\frac{\tau}{\|\nabla\|_2}$, otherwise the update is not changed. The final aggregate is an average of all the updates, scaled or otherwise.

3) Multi-krum: Blanchard et al. [10] proposed Multi-krum AGR as a modification to their own Krum AGR [10]. Multi-krum selects an update using Krum and adds it to a *selection set*, S. Multi-krum repeats this for the *remaining updates* (which remain after removing the update that Krum selects) until S has c updates such that $n - c > 2m + 2$, where n is the number of selected clients and m is the number of compromised clients in a given round. Finally, Multi-krum averages the updates in S.

4) Trimmed-mean: Trimmed-mean [68], [70] aggregates each dimension of input updates separately. It sorts the values of the j^{th}-dimension of all updates. Then it removes m (i.e., the number of compromised clients) of the largest and smallest values of that dimension, and computes the average of the rest of the values as its aggregate of the dimension j.

III. Systemization of FL Poisoning Threat Models

We discuss the key dimensions of the threat models of poisoning attacks on FL, and argue that only two combinations of these dimensions are of practical interest for production FL. Note that, there exist taxonomies of FL poisoning attacks [27], [31] which provide comprehensive overviews of the poisoning attacks in existing literature. In contrast, our work aims to provide a systematic framework to model the existing and future poisoning threats to federated learning.

A. Dimensions of Poisoning Threat to FL

In this section, we build on previous systemization efforts for adversarial ML [4], [9], [29], [43] and present three key dimensions for the threat model of FL poisoning, as shown in Table II.

1) Adversary's Objective: Inspired by [9], we define three attributes of the adversary's objectives.

Security violation: The adversary may aim to cause an *integrity* violation, i.e., to evade detection without disrupting normal service operations, or an *availability* violation, i.e., to compromise the service for legitimate users.

Attack specificity: The attack is *discriminate* if it aims to cause misclassification of a specific set/class of samples; it is *indiscriminate* otherwise.

Error specificity: This attribute is especially relevant in multi-class classification settings. It is *specific* if the attacker's goal is to have a sample misclassified as a specific class; the attack is *generic* if the attacker does not care about the wrong label assigned to the misclassified samples.

Adversary objectives in different classes of poisoning: Here, based on the above taxonomy, we discuss the adversary's objective for different types of poisoning attacks (Figure 1).

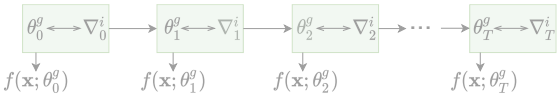
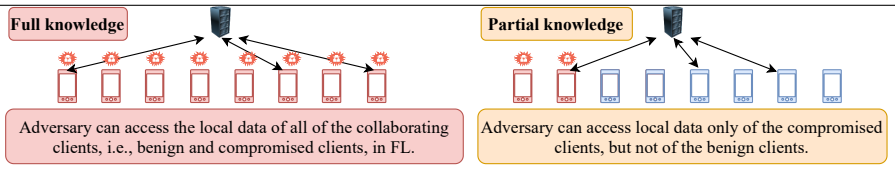
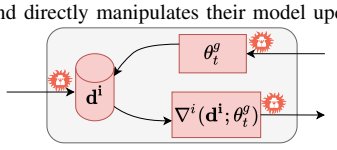
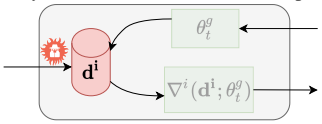
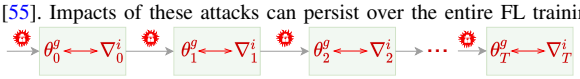
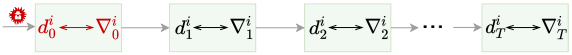
Targeted attacks [7], [59] aim to misclassify specific sets/-classes of input, hence they are "discriminate." Such discriminate attacks can be either used for "integrity" or "availability" violations, depending on how the poisoned data is used.

Semantic backdoor attacks [3], [61] have the same goal as the targeted attacks, but the targeted inputs should have specific properties, e.g., a pixel pattern or a word sequence. Hence, these are "discriminate," "availability" or "integrity" attacks.

Artificial backdoor attacks [67] aim to misclassify *any* input containing a backdoor *trigger*, hence these attacks are "indiscriminate" attacks. Note that, such test inputs should be modified to have the backdoor trigger and only the adversary or a malicious client know the trigger. Hence, these attacks aim to evade the detection, i.e., cause an integrity violation. Hence, these are "integrity indiscriminate" attacks.

Untargeted attacks [5], [23], [55] aim to misclassify any test input, i.e., they are "indiscriminate" attacks. But, test inputs need not be modified in order to misclassify. Hence, these are "availability" attacks.

Table II: The key dimensions of the threat models of poisoning attacks on FL. Each combination of these dimensions constitutes a threat model (Table IV). However, we argue in Section III-B2 that only two of these combinations are practical threat models.

Dimension	Attribute	Values	Description
Objective of the adversary	*Security violation*	Integrity	Misclassify a (adversarially crafted) test input in order to evade detection.
		Availability	Misclassify an unmodified test input to cause service disruption for benign users.
	Attack specificity	Discriminate	Misclassify a small and/or specific set of inputs at the test time.
		Indiscriminate	Misclassify all or most of inputs at the test time.
	Error specificity	Specific	Misclassify a given modified/pristine test input to a specific class.
		Generic	Misclassify a given modified/pristine test input to any class.
Knowledge of the adversary	*Knowledge of the global model*	Whitebox	Adversary can access the global model parameters as well as its predictions, e.g., in the model poisoning case. 
		Nobox	Adversary cannot access parameters or predictions of global model, e.g., in the data poisoning case. 
	Knowledge of the data from the distribution of benign clients' data	Full	
		Partial	
Capabilities of the adversary	*Capabilities in terms of access to client devices*	Model poison	Adversary breaks into the compromised clients (e.g., by circumventing security protocols of operating systems such as Android) and directly manipulates their model updates. 
		Data poison	Adversary can only manipulate local data of the compromised clients; the clients use this data to compute their updates. Adversary does not break into the compromised clients. 
	Capabilities in terms of frequency of the attack (Attack mode)	Online	Adversary repeatedly and adaptively poisons the compromised clients during FL, e.g., model poisoning attacks [7], [23], [55]. Impacts of these attacks can persist over the entire FL training. 
		Offline	Adversary poisons the compromised clients only once at the beginning of FL, e.g., baseline label flipping attacks [23], [61]. Impact of such attacks may quickly fade away. 

Finally, the error specificity of each of these attacks can be either "specific" or "generic."

Focus of our work: In this work, *we focus on untargeted attacks*, i.e., indiscriminate availability attacks with generic error specificity, for the following reasons.

Untargeted attacks pose a great threat to production FL: Untargeted attacks are designed to impact all clients and all test inputs. For instance, FL on FEMNIST achieves an 85% [52] accuracy in a benign setting, and untargeted attacks reduce the accuracy to, e.g., $[78, 82]\%$ depending on the percentages of compromised clients. Such an accuracy drop is significant for production FL, as *a malicious service provider can gain advantage over their competitors by causing such small, yet noticeable, accuracy reductions in the competing services* and *such small accuracy reductions can impact most clients and data from all classes in arbitrary fashion.*

Untargeted attacks can go undetected for long duration: As discussed above, the untargeted attack aims at reducing the overall accuracy of the global model, even by only a few percentage points. Such a small reduction in accuracy is hard to detect in practical settings due to the absence of reliable benchmarks for the target application. For instance, the affected service provider will never know that they could have achieved an 85% accuracy and will believe that $[78, 82]\%$ is the highest achievable accuracy.

Constructing untargeted attacks is more challenging: Untargeted attacks aim to solve a more challenging problem, which is affecting arbitrary test inputs. However, while there exist several defenses to protect FL against untargeted poisoning [10], [41], [55], [70], *these attacks are not studied under production FL environments* (as discussed later on).

2) Adversary's Knowledge: Below we elaborate on two dimensions of adversary's knowledge: knowledge of the global

model and knowledge of the data from the benign distribution.

Knowledge of the global model: This can be *nobox* or *whitebox*. In the nobox case, the adversary does not know the model architecture, parameters, or outputs, and is the most practical setting in FL [32], e.g., the data poisoning adversary has nobox knowledge of the global model. In the whitebox case, the adversary knows the global model parameters and outputs, whenever the server selects at least one compromised client. The model poisoning adversary always has whitebox knowledge of the global model. As we will explain in Section III-B3, this is a relatively less practical setting in FL, as it assumes complete control of the compromised devices.

Knowledge of the data from benign distribution: This can be *full* or *partial*. In full knowledge case, the adversary can access the benign local data of compromised as well as benign clients. In partial knowledge case, the adversary can access the benign local data only of the compromised clients. We only consider the partial knowledge case, because accessing the data of all the clients is impractical in production FL.

3) Adversary's Capability: Below, we elaborate on the the adversary's capability in terms of *access to client devices* and *frequency of attack*, i.e., the *attack mode*.

Capability in terms of access to client devices: Based on the FL stages (part of FL pipeline on client device) that the adversary poisons, there can be a *model poisoning adversary* or a *data poisoning adversary*. The model poisoning adversary can break into a compromised device (e.g., by circumventing the security protocols of operating systems such as Android) and can *directly manipulate* the poisoned updates [5], [10], [23], [41], [50], [55]. This adversary can craft highly effective poisoned updates, but due to unreasonable amount of access to client devices, it can compromise very small percentages of FL clients [24], [32].

On the other hand, a data poisoning adversary cannot break into a compromised device and can only poison its local dataset. The compromised clients use their local poisoned datasets to compute their poisoned updates, hence this adversary *indirectly manipulates* the poisoned updates. Due to the indirect manipulation, these updates may have less poisoning impact than the model poisoning updates. But, due to the limited access required to the compromised clients, this adversary can compromise relatively large percentages of FL clients [24], [32].

Capability in terms of attack frequency (Attack mode): The mode of poisoning attacks on FL can be either *offline* or *online*. In the offline mode, the adversary poisons the compromised clients only once before the start of FL training, e.g., the baseline label flip attack [23] flips the labels of data of compromised clients once before the FL training starts. In the online mode, the adversary *repeatedly* and *adaptively* poisons the compromised clients, e.g., existing model poisoning attacks [5], [55] repeatedly poison the updates of compromised clients selected by the server.

Finally, we assume that the compromised clients can collude to exchange their local data and model updates in order to increase impacts of their attacks.

B. Practical Considerations for Poisoning Threat Models

1) Salient Features of Production Federated Learning: Production FL can be either **cross-device** or **cross-silo** [32]. In *cross-device FL*, the number of clients (N) is large (from few thousands to billions) and only a small fraction of them is chosen in each FL training round, i.e., $n \ll N$. In cross-device FL, clients' devices are highly resource constrained, and therefore, they can process only a limited amounts of data in an FL round. Also, as the devices have highly unreliable network connections, it is expected that a small fraction of the selected devices may drop out in any given FL round. Note that, this equally impacts both benign and compromised clients and does not affect the robustness; this is similar to how the choice of n has no impact on the robustness (Section V-C3). In *cross-silo FL*, N is moderate (up to 100) and all clients are selected in each round, i.e., $n = N$. Clients are large corporations, e.g., banks, and have devices with ample resources. Hence, they can process very large amounts of data and client drop-outs do not happen.

In both FL types, the on-device model used for inference and the on-device model being trained are different. Hence, an adversary cannot gain any insight into the training-model by querying the inference-model, i.e., nobox access (Table II), and must break into the device, i.e., get whitebox access (Table II).

Finally, we assume that production systems are adequately protected against standard attack vectors and vulnerabilities such as Sybil attacks. For instance, if the adversary manages to operate millions of fake accounts [22], we argue that the service provider should prioritize improving their security attestation protocols instead of deploying FL. Section III-B3 also explains that the cost of operating a large scale, persistent botnet in modern operating systems, e.g., Android, is non-trivial. Please refer to [32] for more details on production FL.

2) Understanding the practicality of threat models: For our goal of untargeted poisoning with the partial knowledge of the benign data, we can combine the rest of the dimensions in Table II and obtain eight possible threat models (Table IV). We argue that only T4 (nobox offline data poison) and T5 (whitebox online model poison) are of practical value, and below, justify why other models are less relevant in practice: **(1)** With model poisoning capability, the adversary has whitebox access by default, hence, T1 and T2 in Table IV are not valid. **(2)** In cross-device FL, only a few selected clients get the most recent global model in each round. Hence, to gain whitebox access to the model, the adversary needs to control (i.e., break into) a large number of devices (so that in most FL rounds, the FL server picks at least one of them), which is impractical in practice as we explain in Section III-B3. With whitebox access, the adversary can mount the stronger online model poisoning attacks (MPAs) instead of data poisoning attacks (DPAs). Therefore, T3, T7, and T8 are not reasonable threat models, as they combine whitebox access with either offline attacks or DPAs. **(3)** Under T6 (nobox online data poison), the adversary mounts an online attack, i.e., they *adaptively* poison

Table III: Practical ranges of FL parameters based on the literature and discussions on FL production systems [11], [24], [32] and the ranges used in *untargeted* FL poisoning and robust AGRs literature [5], [10], [23], [41], [55]. MPA means model poisoning attack and DPA means data poisoning attack. Red (green) cells denote impractical (practical) ranges.

Parameters/Settings	**What we argue to be practical**	**Used in previous *untargeted* works**
FL type + Attack type	Cross-silo + DPAs Cross-device + {MPAs, DPAs}	Cross-silo + MPAs
Total number of FL clients, N	Order of $[10^3, 10^{10}]$ for cross-device $[2, 100]$ for cross-silo	[50, 100]
Number of clients chosen per round, n	Small fraction of N for cross-device All for cross-silo	All
% of compromised clients, M	$M \leq$0.1% for DPAs $M \leq$0.01% for MPAs	$[20, 50]\%$
Average size of benign clients' data, $\lvert D\rvert_{\text{avg}}$	$[50, 1000]$ for cross-device Not applicable to cross-silo	Not studied for cross-device $[50, 1000]$ for cross-silo
Maximum size of local poisoning data	Up to $100 \times \lvert D\rvert_{\text{avg}}$ for DPAs Not applicable to MPAs	$\sim \lvert D\rvert_{\text{avg}}$

Table IV: The eight possible threat models for *untargeted poisoning attacks* on FL. T3-T8 are valid, but only T4 and T5 represent practical FL deployments (Section III-C).

	Capability $\in \{\texttt{MP}, \texttt{DP}\}$	Knowledge $\in \{\texttt{Nb}, \texttt{Wb}\}$	Attack mode $\in \{\texttt{Off}, \texttt{On}\}$
T1	Model poison	Nobox	Offline
T2	Model poison	Nobox	Online
T3	Model poison	Whitebox	Offline
T4	Model poison	Whitebox	Online
T5	Data poison	Nobox	Offline
T6	Data poison	Nobox	Online
T7	Data poison	Whitebox	Offline
T8	Data poison	Whitebox	Online

the local data of compromised clients. But, as the adversary has no knowledge of the (current) global model due to nobox access, they cannot generate new poisoning data adaptively. Hence, the combination of nobox and online is not practical.

3) Practical Ranges of FL Parameters: We argue that the literature on untargeted poisoning [5], [10], [23], [41], [55] rarely evaluates their proposed attacks/defenses for the production FL settings, primarily due to their motivation to perform worse-case analyses. But, we show that such analyses lead to conclusions that do not apply to production FL.

Table III demonstrates the stark differences between the parameter ranges used in the untargeted poisoning literature and their practical ranges, which we have obtained from recent surveys [11], [32] and discussion among FL experts [24]. This is due to the more challenging nature of untargeted poisoning in FL. We attribute this to the difficulty of establishing successful untargeted attacks for practical settings, as we will also show in our evaluations.

Contrary to what production FL settings encounter, previous works commonly evaluate robustness using very high percentages of compromised clients and/or using model poisoning attacks on cross-silo FL (Table III). However, we use small percentages of compromised clients $M \leq 1$, for cross-device FL, use large numbers of clients $N \in [1,000, 34,000]$ and use $n \in [25, 50] \ll N$ in each round; we use N=n=50.

In particular, consider the percentages of compromised clients; state-of-the-art attacks [5], [23], [55] (defenses [10], [16], [68], [70]) assume adversaries who can compromise up to 25% (50%) of FL clients. The cost of creating and operating a compromised client botnet at scale (which includes breaking into devices) is non-trivial. To create the botnet, the adversary would need to either buy many physical devices (∼$25 each) and root them (for state-of-the-art model poisoning attacks [5], [23], [55]), pay for access to large but undetected botnets with remote administrative access, or develop an entirely new botnet via compromising a popular app/sdk to exploit unpatched security holes and gain persistence. To operate the botnet, the adversary must avoid detection by antimalware services [28] as well as dynamic anti-abuse services (such as Android's SafetyNet [53]). With a botnet in place, the adversary may further need to pay for a skilled engineering team to keep malicious FL code in sync with the target FL-enabled app and to reverse-engineer frequently-shifting ML workloads. Such an engineering team could instead change apps' behaviors to mimic the effect of a compromised FL-trained model, they might use their privileged access to steal login credentials for account hijacking, or they might participate in ad/click fraud or bank fraud or ransomware for financial gain. More plausible scenarios for an adversary reaching double-digit client percentages—such as an app insider—likely enable attacker-controlled FL servers, thereby removing them from the literature's standard threat model.

For data poisoning attacks, we assume that compromised clients can have a limited amount of poisoned data D_p. Because, in cross-device FL, the devices with low processing powers (e.g., smart phones and watches) can process limited D_p in the short duration of FL rounds. However, in cross-silo FL, silos can inspect D_p and remove D_p with sizes much larger than the average size of clients' data $\lvert D\rvert_{\text{avg}}$. Hence, we argue that $\lvert D_p\rvert$ should be up to $100 \times \lvert D\rvert_{\text{avg}}$. We discuss rest of the parameters from Table III in the corresponding sections.

C. Threat Models in Practice

Here we discuss the two threat models of practical interest.

1) Nobox Offline Data Poisoning (T4): In this setting, the adversary does not know the architecture, parameters, or outputs of the global model. The adversary knows the server's AGR, but may or may not know the global model architecture; we evaluate both cases. We assume that the adversary knows

the benign data of the compromised clients and mounts offline data poisoning attacks (DPAs).

This adversary does not require any access to the internals (e.g., FL binaries, memory) of compromised devices, and therefore, can compromise large percentages of production FL clients, e.g., on order of up to 0.1% [24], [32]. However, the poisoning impact of the corresponding poisoned updates is very limited. This is partly because arbitrarily poisoned updates (e.g., of model poisoning attacks (MPAs) [5], [23], [55]) need not map to the valid data domain. For instance, consider the standard *max* function: $f(x,y)$=max(x,y). Gradient of this function with respect to either x or y is always 0 or 1 [19]. Hence, a DPA cannot have a poisoned update with an arbitrary value for gradients of the parameters. But an MPA can, because it can directly assign any arbitrary value to the parameters' gradients.

2) Whitebox Online Model Poisoning (T5): The adversary knows the parameters and predictions of the global model whenever the server selects at least one compromised client. We assume that the adversary knows the server's aggregation rule and the benign data on the compromised devices. The adversary mounts online MPAs.

Unlike data poisoning adversary, this adversary breaks into the compromised devices, which is extremely costly as discussed in Section III-B3. Hence, in practice, a model poisoning adversary can compromise very small percentages of FL clients, e.g., on order of up to 0.01% [24], [32]. However, due to their ability to directly manipulate the model updates, in theory, a model poisoning adversary can craft highly poisonous updates. We can justify this claim from the example of a zero-value parameter discussed in Section III-C1.

IV. Exploring the Space of FL Poisoning Attacks

A. Existing FL Poisoning Attacks

1) Data Poisoning Attacks (DPAs): DPAs have been studied mainly for centralized ML [17], [44], [64], [65], [69], and no prior work has studied untargeted DPAs that are tailored to FL settings. Fang et al. [23] show the possibility of applying simple label flipping attacks to FL, where each compromised client flips the labels of their data from true label $y \in [0, C-1]$ to false label $(C-1-y)$ if C is even and to false label $(C-y)$ if C is odd, where C is the number of classes.

2) Model Poisoning Attacks (MPAs): These consider our whitebox online model poisoning threat model (T4) from Section III-C2).

Little Is Enough (LIE) attack [5] adds small amounts of noise to each dimension of the average of the benign updates. Specifically, the adversary computes the average (∇^b) and the standard deviation (σ) of the available benign updates; then computes a coefficient z based on the number of benign and compromised clients; and finally computes the poisoned update as $\nabla' = \nabla^b + z\sigma$. [5] shows that such noises easily evade the detection by robust AGRs as well as effectively poison the global model.

Static Optimization (STAT-OPT) attack [23] proposes a general FL poisoning framework and then tailors it to specific AGRs. STAT-OPT computes the average (∇^b) of the available benign updates and computes a *static malicious direction*, $\omega = -sign(\nabla^b)$; the final poisoned update, ∇', is $-\gamma\omega$ and the attack finds a suboptimal γ that circumvents the target AGR; for details please refer to [23]. Unlike LIE, STAT-OPT attacks carefully tailor themselves to the target AGR, and hence, perform better.

Dynamic Optimization (DYN-OPT) attack [55] proposes a general FL poisoning framework and then tailors it to specific FL settings. DYN-OPT computes an average of the available benign updates, ∇^b, and perturbs it in a *dynamic, data-dependent malicious direction* ω to compute the final poisoned update $\nabla' = \nabla^b + \gamma\omega$. DYN-OPT finds the largest γ that successfully circumvents the target AGR. DYN-OPT is much stronger, because unlike STAT-OPT, it finds the largest γ and uses a dataset tailored ω.

B. Our Improved FL Poisoning Attacks

We first present a general optimization problem to model FL poisoning attacks. Then we use it to design improved poisoning attacks on state-of-the-art AGRs from Section II-B.

1) Formulating FL Poisoning as an Optimization Problem: Our optimization problem for poisoning attacks is based on that of [55]. Specifically, we aim to craft poisoned updates (via data or model poisoning) which will increase the overall distance between the poisoned aggregate (computed using poisoned and benign updates) and the benign aggregate (computed using only benign updates). This can be formalized as follows:

$$\underset{\nabla' \in R^d}{\text{argmax}} \quad \|\nabla^b - \nabla^p\| \tag{1}$$
$$\ldots \nabla^b = f_{\mathsf{avg}}(\nabla_{i\in\{[n']\}}), \quad \nabla^p = f_{\mathsf{agr}}(\nabla'_{\{i\in[m]\}}, \nabla_{\{i\in[n']\}})$$

where, m is the number of compromised clients selected in the given round, f_{agr} is the target AGR, f_{avg} is the Average AGR, $\nabla_{\{i\in[n']\}}$ are the benign updates available to the adversary (e.g., updates computed using the benign data of compromised clients), ∇^b is a reference benign aggregate, and $\nabla'_{\{i\in[m]\}}$ are m replicas of the poisoned update, ∇', of our attack. ∇^p is the final poisoned aggregate.

Although our optimization problem in (1) is same as [55], *two key differences from [55] are:* **(1)** We are the first to use (1) to construct systematic data poisoning attacks on FL. **(2)** Our model poisoning attacks not only tailor the optimization in (1) to the given AGR (as in [55]), but also to the given dataset and global model, by using stochastic gradient ascent algorithm (Section IV-B3); this boosts the efficacy of our attack.

2) Our Data Poisoning Attacks (DPAs): We formulate a general DPA optimization problem using (1) as follows:

$$\underset{D_p \subset \mathcal{D}}{\text{argmax}} \quad \|\nabla^b - \nabla^p\| \tag{2}$$
$$\ldots \nabla^b \text{ and } \nabla^p \text{ as in (1) and } \nabla' = A(D_p, \theta^g) - \theta^g$$

where $\mathcal{D}$ is the entire input space and D_p is the poisoning data used to compute the poisoned update ∇' using a training algorithm A, e.g., mini-batch SGD, and global model θ^g. The

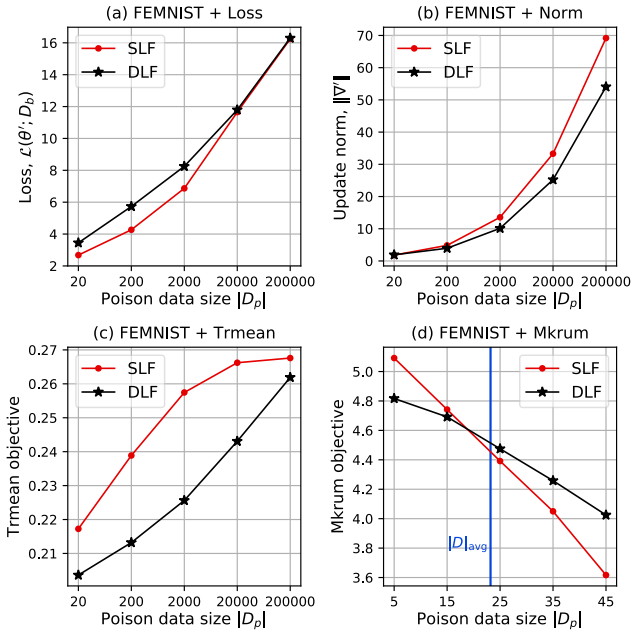


Figure 2: Effect of varying the sizes of poisoned data, D_p, on the objectives of DPAs (Section IV-B2) on various AGRs. We compute D_p by flipping the labels of benign data.

rest of the notations are the same as in (1). To solve (2), we find D_p such that when θ^g is fine-tuned using D_p, the resulting model θ' will have high cross-entropy loss on some benign data D_b (e.g., that of compromised clients), i.e., high $L(D_b; \theta')$, and the corresponding update $\nabla' = \theta' - \theta^g$ will circumvent the target AGR. Our intuition is that, when the global model is updated using such ∇', it will have high loss on benign data [8], [30], [43].

Sun et al. [57] propose DPAs on federated multi-task learning where each client learns a different task. Hence, their attacks are orthogonal to our work. On the other hand, as [23] demonstrates, backgradient optimization based DPAs [43] are computationally very expensive (*∼10 days* to compute poison for a subset of MNIST task) yet ineffective.

Instead, because the central server has no visibility into the clients' data or their sizes, we propose to use an appropriate amount of label flipped data as D_p for each of the compromised clients. Our intuition behind this approach is the same as before: the larger the amount of label flipped data used to compute θ', the larger the $L(D_p; \theta')$ and $\|\nabla'\|$, and therefore, the higher the deviation in (2). We validate this intuition using FEMNIST dataset in Figure 2 for various AGRs. For instance, Figures 2 (a) and (b) show that increasing $|D_p|$ monotonically increases update's loss and norm, respectively, and hence, can effectively poison the Average AGR [10], [41].

In our work, we propose two label flipping (LF) strategies: **static LF (SLF)** and **dynamic LF (DLF)**. In SLF, for a sample $(\mathbf{x}, y)$, the adversary flips labels in a static fashion as in Section IV-A1. On the other hand, in DLF, the adversary computes a surrogate model $\hat{\theta}$, an estimate of θ^g, e.g., using the available benign data, and flips y to the least probable label with respect to $\hat{\theta}$, i.e., to $argmin\ \hat{\theta}(\mathbf{x})$. We observe that

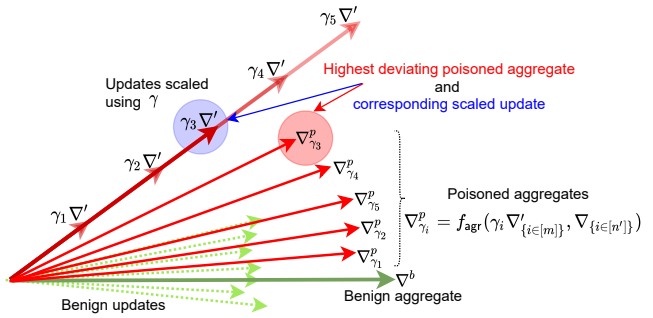


Figure 3: Schematic of our PGA attack: PGA first computes a *poisoned update* ∇' using stochastic gradient ascent (SGA). Then, f_{project} finds the scaling factor γ that maximizes the deviation between benign aggregate ∇^b and poisoned aggregate ∇^p_γ. Robust aggregations easily discard the scaled poisoned updates, $\gamma\nabla'$, with very high γ (e.g., $\gamma_{\{4,5\}}$), while those with very small γ (e.g., $\gamma_{\{1,2\}}$) have no impact.

the impacts of the two LF strategies are dataset dependent. Therefore, for each dataset, we experiment with both of the strategies and, when appropriate, present the best results. We now specify our DPA for the AGRs in Section II-B.

Average: To satisfy the attack objective in (2) for Average AGR, we produce updates with large loss and norm [10], [41] using very large amounts of label flipped data (Figures 2-(a,b)).

To obtain large $|D_p|$, we combine the benign data of all compromised clients and flip their labels using either SLF or DLF strategy (simply SLF/DLF). To increase $|D_p|$ further, we add Gaussian noise to existing feature vectors of $|D_p|$ to obtain new feature vectors and flip their labels using SLF or DLF.

Norm-bounding: To attack Norm-bounding AGR (Section II-C2), we use large $|D_p|$ to generate poisoned updates that incur high losses on benign data (as we show in Figure 2-(b)). As our evaluations will show, even if their norms are bounded, such poisoned updates remain far from benign updates and have high poisoning impacts. This leads to effective attacks, but only at high percentages of compromised clients (e.g., M=10%). Due to space restrictions, we provide the details of our DPAs on Mkrum and Trmean in Appendix B1.

3) Our Model Poisoning Attacks (MPAs): We use (1) as the general optimization problem for our MPAs. To solve this optimization, we craft a poisoned model θ' with high $L(D_b; \theta')$ while ensuring that the corresponding poisoned update, ∇', circumvents the target AGR.

Model poisoning adversary can directly manipulate the compromised clients' updates (Section III-C2). Hence, first, our attack uses the stochastic gradient ascent (SGA) algorithm (instead of SGD) and fine-tunes θ^g to increase (instead of decreasing) the loss on some benign data, D_b, to obtain a malicious θ'. But, in order to ensure that the corresponding poisoned update, i.e., $\nabla' = \theta' - \theta^g$, circumvents the target AGR, we *project* the update on a ball of radius τ around origin, i.e., scale the update to have a norm $\|\nabla'\| \leq \tau$, where τ is the average of norms of the available benign updates. Hence, we call our attack **projected gradient ascent (PGA)**. To perform stochastic gradient ascent, we increase the loss on

batch b of data by *using the opposite of a benign gradient direction*, i.e., $-\nabla_\theta \mathcal{L}(\theta; b)$.

Algorithm 1 (Appendix B) gives the overview of our MPA. The adversary first computes τ (line 2), an average of the norms of some benign updates available to her ($\nabla_{\{i\in[n']\}}$). Then, the adversary fine-tunes θ^g using D_p and SGA to compute a poisoned update ∇'; our attack computes ∇' for any AGR in the same manner. Finally, the adversary uses f_{project} function to appropriately project ∇' in order to circumvent the robustness criteria of the target AGR, f_{agr}.

Algorithm 2 (Appendix B) describes f_{project}: It computes $\nabla^b = f_{\text{avg}}(\nabla_{\{i\in[n']\}})$. Then, it finds a scaling factor γ for ∇' that maximizes the distance between the benign aggregate ∇^b and the poisoned aggregate $\nabla^p = f_{\text{agr}}(\gamma\nabla'_{\{i\in[m]\}}, \nabla_{\{i\in[n']\}})$. Note that, there can be many ways to optimize γ [55], but we empirically observe that simply searching for γ in a pre-specified range (e.g., $[1, \Gamma]$ with $\Gamma \in \mathbb{R}^+$) yields strong attacks (line 6). Figure 3 depicts the idea of f_{project} algorithm.

Due to the modular nature of our attacks, one can attack any given AGR by plugging its algorithm in Algorithm 2. This is unlike Sun et al. [58], who propose a similar *targeted* attack which only works against norm-bounding AGR.

Furthermore, to reduce computation, below we tailor f_{project} to the state-of-the-art AGRs from Section II-B; note that, the adversary obtains a poisoned update, ∇', using Algorithm 1 before tailoring f_{project} to the target AGR.

Average: Average does not impose any robustness constraints, therefore, we simplify f_{project} by scaling ∇' by an arbitrarily large constant, e.g., 10^{20}. If the server selects a compromised client, such poisoned update suffices to completely poison θ^g.

Norm-bounding: Following the Kirchoff's law, we assume that the attacker knows the norm-bounding threshold, τ, and therefore, f_{project} scales ∇' by $\frac{\tau}{\|\nabla'\|}$, so that the norm of the final ∇' will be τ. We provide the details of our MPAs on Mkrum and Trmean in Appendix B2.

V. Analysis of FL Robustness in Practice

In this section, we evaluate state-of-the-art data (DPAs) and model poisoning attacks (MPAs) against non-robust and robust FL algorithms (Section II-B), under practical threat models from Section III-C. We start by analyzing cross-device FL (Sections V-A to V-C), as it is barely studied in previous works and is more susceptible to poisoning. Then, we will analyze cross-silo FL in Section V-D.

Experimental setup: Please refer to Appendix C.

Attack impact metric: A_θ denotes the maximum accuracy that the global model achieves over all FL training rounds, without any attack. A^*_θ for an attack denotes the maximum accuracy of the model under the given attack. We define *attack impact*, I_θ, as *the reduction in the accuracy of the global model due to the attack*, hence for a given attack, $I_\theta = A_\theta - A^*_\theta$.

A. Evaluating Non-robust FL (Cross-device)

We study Average AGR due to its practical significance and widespread use. Previous works [5], [10], [23], [41], [55], [70] have argued that even a single compromised client can prevent the convergence of FL with Average AGR. However, our results contradict those of previous works: we show that this established belief about Average AGR is *incorrect* for production cross-device FL.

Figure 4a shows the attack impacts (I_θ) of various DPAs and MPAs. Note that, for the Average AGR, all MPAs [5], [23], [55], including ours, are the same and craft arbitrarily large updates in a malicious direction. Hence, we show a single line for MPAs in Figure 4a.

We see that *for cross-device FL, when percentages of compromised clients (M) are in practical ranges (Table III), I_θ's of all the attacks are very low, i.e., the final θ^g converges with high accuracy*. For FEMNIST, I_θ of MPAs at M=0.01% is ∼2% and I_θ of DPAs at 0.1% is ∼5%. In other words, compared to the no attack accuracy (82.3%), the attacks reduce the accuracy by just 2% and 5%. Similarly, we observe very low I_θ's for the Purchase and CIFAR10 datasets.

Note that, here we use very large local poisoned data (D_p) for our DPAs, as DPAs on Average AGR become stronger with higher $|D_p|$ (Section IV-B2); $|D_p|$'s are 20,000, 50,000, and 20,000 for FEMNIST, CIFAR10, and Purchase, respectively. However, as we will show in Section V-C1, *under practical $|D_p|$, I_θ's of DPAs are negligible even with M=10%*.

The inherent robustness of cross-device FL is due to its client sampling procedure. In an FL round, the server selects a very small fraction of all FL clients. Hence, in many FL rounds no compromised clients are chosen when M ($< 1\%$) is in practical ranges.

> **(Takeaway V-A)** Contrary to the common belief, production cross-device FL with (the naive) Average AGR converges with high accuracy even in the presence of untargeted poisoning attacks.

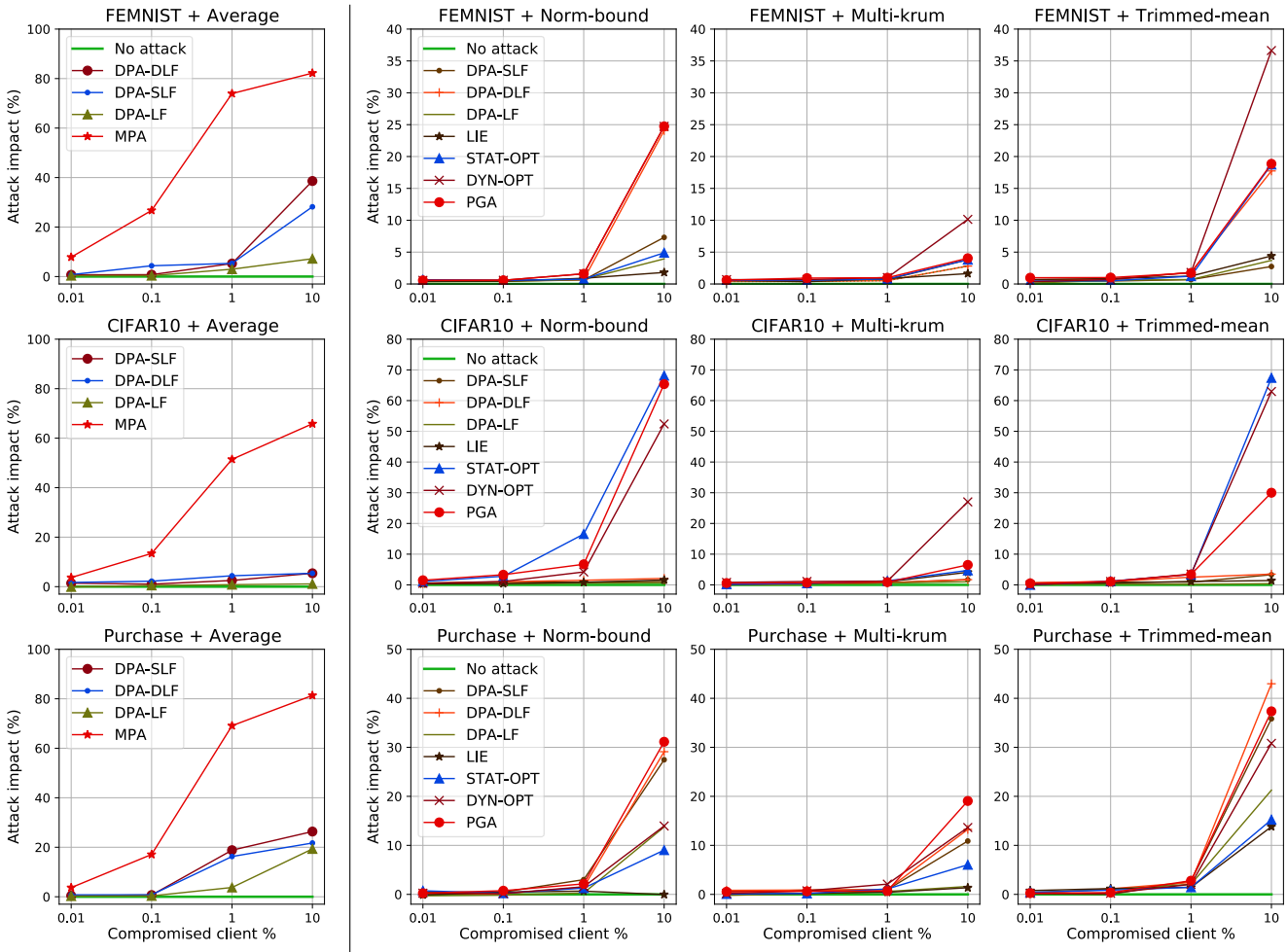
B. Evaluating Robust FL (Cross-device)

In this section, contrary to previous works, we study the robustness of robust AGRs for cross-device FL when percentages of compromised clients (M) are in practical ranges. Figure 4b shows the poisoning impact (I_θ) of DPAs and MPAs for Norm-bounding (Normb), Multi-krum (Mkrum), and Trimmed-mean (Trmean) AGRs. Below, we discuss three **key takeaways**:

1) **Cross-device FL with robust AGRs is highly robust in practice:** I_θ of attacks on robust AGRs are negligible in practice, i.e., when $M \leq 0.1\%$ for DPAs and $M \leq 0.01\%$ for MPAs. For instance, $I_\theta \leq 1\%$ for all of state-of-the-art attacks on all the three datasets, i.e., the attacks reduce the accuracy of θ^g by less that 1 percent.

We also run FL with a robust AGR for a very large number (5,000) of rounds to investigate if the strongest of MPAs against the AGR with $M = 0.1\%$ can break the AGR after long rounds of continuous and slow poisoning. Figure 6 shows the results: Mkrum and Trmean remain completely unaffected (in fact accuracy of the global model increases), while accuracy due to Normb reduces by <5%.

In summary, state-of-the-art poisoning attacks [5], [23], [55] demonstrate that the robust AGRs are significantly less robust



(a) Non-robust FL

(b) Robust FL

Figure 4: (4a) Attack impacts (I_θ) of state-of-the-art data (DPA-DLF/SLF) and model (MPA) poisoning attacks on cross-device FL with average AGR. *I_θ's are significantly low for practical percentages of compromised clients ($M \leq 0.1\%$).* (4b) I_θ of various poisoning attacks (Section IV) on robust AGRs (Section II-B). *These AGRs are highly robust for practical M values.*

than their theoretical guarantees. On the other hand, our findings show that these AGRs are more than sufficient to protect, more practical, production cross-device FL against untargeted poisoning. This is due to the peculiar client sampling of cross-device FL, as discussed in Section V-A.

(Takeaway V-B1) Cross-device FL with robust AGRs is highly robust to state-of-the-art poisoning attacks under production FL environments (M <0.1%, $n \ll N$).

2) **Investigating simple and efficient robustness checks is necessary:** Most of the state-of-the-art robust AGRs with strong theoretical guarantees [10], [41], [68], [70] have complex robustness checks on their inputs, which incur high computation and storage overheads. For instance, to process n updates of length d, the computational complexity of Mkrum is $\mathcal{O}(dn^2)$ and that of Trmean is $\mathcal{O}(dn\log n)$. Therefore, in production FL systems where n can be up to $5,000$ [11], [32], the computation cost prohibits the use of such robust AGRs.

On the other hand, Norm-bounding only checks for the norm of its inputs and has computation complexity of $\mathcal{O}(d)$, same as Average. Figure 4b shows that a simple and efficient AGR, Norm-bounding, protects cross-device FL against state-of-the-art poisoning attacks similarly to the theoretically robust (and expensive) AGRs, under practical M. For instance, for all the datasets with $M \leq 1\%$, $I_\theta < 1\%$ for all of the AGRs (Figure 4b). Our evaluation highlights that simple robust AGRs, e.g., Norm-bounding, can effectively protect cross-device FL in practice, and calls for further investigation and invention of such low-cost robust AGRs.

(Takeaway V-B2) Even the simple, low-cost Norm-bounding AGR is enough to protect production FL against untargeted poisoning, questioning the need for the more sophisticated (and costlier) AGRs.

3) **Thorough empirical assessment of robustness is inevitable:** Theoretically robust AGRs claim robustness to poisoning attacks at high M's, e.g., in theory, Mkrum [10] and Trmean [70] are robust for $M \leq 25\%$. But, we observe that, even at the theoretically claimed values of M, these robust AGRs do not exhibit high robustness; in fact, simple AGRs, e.g., Norm-bounding, are equally robust. Note in Figure 4b

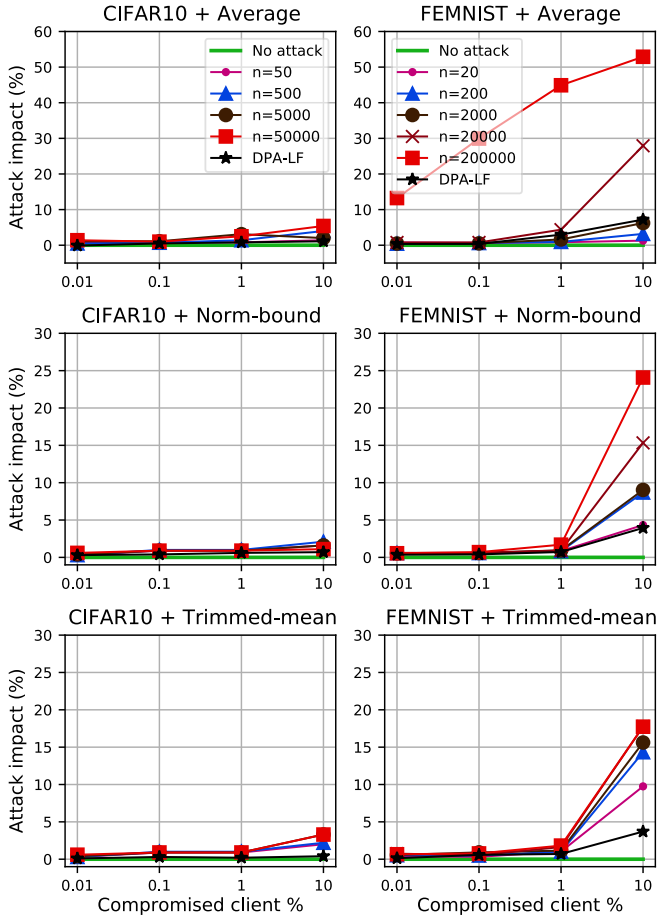


Figure 5: Effect of varying sizes of local poisoned dataset D_p on impacts I_θ of the best of DPAs. When $|D_p|$ and M are in practical ranges, I_θ's are negligible for robust AGRs and are dataset dependent for non-robust Average AGR.

that, for FEMNIST at M=10%, I_θ's on Trmean are *higher* than on Norm-bounding. For CIFAR10 at M=10%, I_θ's for Norm-bounding and Trmean are almost similar.

Sections V-B2 and V-B3 show that, some of the sophisticated, theoretically robust AGRs do not outperform simpler robust AGRs at *any* ranges of M. More importantly they demonstrate the shortcomings of the methodology used to assess the robustness of AGRs in previous works [10], [41], [68], [70] (because these works use very preliminary attacks) and highlight that a thorough empirical assessment is necessary to understand the robustness of AGRs in production FL systems.

(Takeaway V-B3) Understanding the robustness of AGRs in production FL requires a thorough empirical assessment of AGRs, on top of their theoretical robustness analysis.

C. Effect of FL Parameters on Poisoning (Cross-device)

1) Effect of the Size of Local Poisoning Datasets ($|D_p|$) on DPAs.: The success of our state-of-the-art data poisoning attacks depends on $|D_p|$ of compromised clients (Section IV-B2). In Sections V-A and V-B, we use large $|D_p|$ (e.g., 50,000 for CIFAR10) to find the highest impacts of DPAs. But, as argued in Section III-B3, in practice $|D_p| \leq 100 \times |D|_{\text{avg}}$; $|D|_{\text{avg}}$ is the average size of local datasets of benign clients and it is around 20 (50) for FEMNIST (CIFAR10). In Figure 5, we report I_θ of the best of DPA-SLF or DPA-DLF for $|D_p| \in \{1, 10, 10^2, 10^3, 10^4\} \cdot |D|_{\text{avg}}$; we use impractically high $|D_p|$'s of up to $10^4 \cdot |D|_{\text{avg}}$ only for experimental analyses.

Figure 5 shows that I_θ's of DPAs slightly increase with $|D_p|$. For FEMNIST and CIFAR10 with any AGR, including Average, I_θ's are negligible even for unrealistically high $|D_p|$ of $1000 \times |D|_{\text{avg}}$ for $M \leq 1\%$. We omit Mkrum here, as $|D_p|$ of the effective DPAs on Mkrum is always in practical ranges and close to $|D|_{\text{avg}}$ (Section IV-B2).

To summarize, *for all robust AGRs, DPAs have negligible impacts on FL when $|D_p|$ and M are in practical ranges, while for non-robust AGRs, the reductions in I_θ are non-trivial and dataset dependent*. This also means that using a reasonable upper bound on the dataset sizes of FL clients can make FL highly robust to DPAs.

(Takeaway V-C1) Enforcing a limit on the size of the local dataset of each client can act as a highly effective (yet simple) defense against untargeted DPAs in production FL.

2) Effect of the Average Dataset Size of Benign FL Clients ($|D|_{\text{avg}}$): Figure 9 in Appendix E shows I_θ when we vary $|D|_{\text{avg}}$. To emulate varying $|D|_{\text{avg}}$, we vary the total number of FL clients, N, for given dataset, e.g., for CIFAR10, $|D|_{\text{avg}}$ is 50 (10) for N=1,000 (N=5,000). As discussed in Section III-B3, we use $|D_p|$=100 $\times$ $|D|_{\text{avg}}$ for DPAs.

We observe *no clear effect of varying $|D|_{\text{avg}}$ on I_θ's*. For instance, at M=1%, I_θ's of our PGA and DPA-SLF on CIFAR10 + Normb reduce with increase in $|D|_{\text{avg}}$, while I_θ of any attacks on FEMNIST with robust AGRs do not change with varying $|D|_{\text{avg}}$. Due to space restrictions, we defer the explanations of each of these observations to Appendix D.

More importantly, we observe that *even with moderately high $|D|_{\text{avg}}$, cross-device FL completely mitigates state-of-the-art DPAs and MPAs despite M being impractically high*, with an exception of MPAs on Average AGR. For instance, for CIFAR10 with $|D|_{\text{avg}}$=50 and FEMNIST with $|D|_{\text{avg}}$=200, all robust AGRs almost completely mitigate all of DPAs and MPAs, while Average AGR mitigates all DPAs. However, as MPAs are very effective against Average, their I_θ remains high. As clients in FL continuously generate data locally [12], [40], it is common to have large $|D|_{\text{avg}}$ in practice. Interestingly, our evaluation also implies that simply lower bounding the dataset sizes of FL clients improves FL robustness.

(Takeaway V-C2) When local dataset sizes of benign clients are in practical regimes (Table III), cross-device FL with robust AGRs is highly robust to untargeted poisoning.

3) Number of Clients Selected Per Round.: Figure 10 (Appendix B2) shows the effect of varying the number of clients (n) selected by the server in each round (for M=1%). Similar to [23], we do not observe any noticeable effect of n on the impact of attacks, since the expected percentage of compromised clients (M) does not change with n. But, we

observe the opposite behavior for MPAs on Average AGR. This is because, as soon as the server selects even a single compromised client, MPA prevents any further learning of the global model. An increase in n increases the chances of selecting compromised clients, hence amplifying the attack.

(Takeaway V-C3) The number of clients selected in each round of production cross-device FL has no noticeable effect on the impacts of untargeted poisoning attacks, with the exception of MPAs on Average AGR.

4) Effect of Unknown Global Model Architecture on DPAs: DPA-DLF attack (Section IV-B2) uses the knowledge of global model's architecture to train a surrogate model. However, in practice, the nobox offline data poisoning adversary (Section III-C1) may not know the architecture. Hence, we evaluate impact of DPA-DLF under the unknown architecture setting.

We emulate the unknown architecture setting for FEMNIST dataset. We assume that the adversary uses a substitute convolutional neural network given in Table V (Appendix E) as they do not know the true architecture, which is LeNet in our experiments. Figure 7 (Appendix E) compares the impacts of DPA-DLF when the adversary uses the true and the substitute architectures. Note that, *impacts of DPA-DLF reduce when the adversary uses the substitute architecture*.

(Takeaway V-C4) The DPAs that rely on a surrogate model (e.g., our DLF) are less effective if the architectures of the surrogate and global models do not match.

D. Evaluating Robustness of Cross-silo FL

In cross-silo FL, each of N clients, i.e., silos (e.g., corporations like banks, hospitals, insurance providers, government organizations, etc.), collects data from many *users* (e.g., bank customers or hospital patients) and collaboratively train the FL model; we denote the total number of users by N'.

Recall from Section III-C2 that the model poisoning adversary completely breaks into the devices of compromised clients and, to be effective, persists in their systems for long duration because model poisoning attacks are online attacks (Section III-C2). For cross-silo FL, this means that the adversary should break into large corporations, e.g., a bank, who are bound by contract and have professionally maintained software stacks. Plausible cross-silo poisoning scenarios involve strong incentives (e.g., financial) and require multiple parties to be willing to risk the breach of contract by colluding or for one party to hack thereby risking criminal liability. This makes breaking into these silos practically unlikely, hence we argue that *model poisoning threats in cross-silo FL are impractical*.

Note that this is unlike the large scale data-breaches [46]–[48] which are short-lived and are only capable of stealing information, but not changing the infrastructure.

Hence, we only study the data poisoning threat for cross-silo FL. For worse-case analyses, we assume that the silos train their models on all the data contributed by their users. If the silos inspect the users' data and remove the mislabeled data, one should consider clean-label data poisoning attacks [27], [54]; we leave this study to future work. Note that, data inspection is not possible in cross-device FL as data of clients (who are also the users) is completely local, hence clean-label poisoning is not relevant in cross-device FL.

We assume that each silo collects data from equal number (i.e., N'/N) of users. For DPAs, we assume $M\%$ of the N' users are compromised and each of them shares poisoned data D_p (computed as described in Section IV-B2) with their parent silo; as discussed in Section III-B3, we assume $|D_p| = 100 \times |D|_{\text{avg}}$ for *each user*. We distribute the compromised users either *uniformly* across the silos or *concentrate* them in a few silos. For instance, consider 50 silos and 50 compromised users and that, each silo can have a maximum of 50 users. Then in the uniform case, a single compromised user shares her D_p with each silo, while in the concentrated case, all the 50 compromised users share their D_p with a single silo.

Figure 8 (Appendix E) shows the impacts of best of DPAs for the concentrated case. We see that *cross-silo FL is highly robust to state-of-the-art DPAs*. Because, in the concentrated case, very large numbers of *benign silos mitigate the poisoning impact of the very few ($M\%$) compromised silos*. We observe the same results for the uniform distribution case, because very large numbers of *benign users in each silo mitigate the poisoning impacts of the very few ($M\%$) compromised users*.

(Takeaway V-D) In production cross-silo FL, model poisoning attacks are not practical, and state-of-the-art data poisoning attacks have no impact even with Average AGR.

VI. Conclusions

In this work, we systematized the threat models of poisoning attacks on federated learning (FL), provided the practical ranges of various parameters relevant to FL robustness, and designed a suite of untargeted model and data poisoning attacks on FL (including existing and our improved attacks). Using these attacks, we thoroughly evaluated the state-of-the-art defenses under production FL settings. We showed that the conclusions of previous FL robustness literature cannot be directly extended to production FL. We presented concrete takeaways from our evaluations to correct some of the established beliefs and highlighted the need to consider production FL environments in research on FL robustness.

We hope that our systematization of practical poisoning threat models can steer the community towards practically significant research problems in FL robustness. For instance, one such open problem is to obtain concrete theoretical robustness guarantees of existing defenses in production FL settings where only a very small fraction of all clients is randomly selected in each FL round.

Acknowledgements

The work was supported by DARPA and NIWC under contract HR00112190125, and by the NSF grants 1953786, 1739462, and 1553301. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views,

opinions, and/or findings expressed are those of the author(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

REFERENCES

[1] "Federated learning: Collaborative machine learning without centralized training data," https://ai.googleblog.com/2017/04/federated-learning-collaborative.html, 2017.

[2] D. Alistarh, Z. Allen-Zhu, and J. Li, "Byzantine stochastic gradient descent," in *NeurIPS*, 2018.

[3] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *AISTATS*, 2020.

[4] M. Barreno, B. Nelson, and A. D. Joseph, "The security of machine learning," *Machine Learning*, 2010.

[5] M. Baruch, B. Gilad, and Y. Goldberg, "A Little Is Enough: Circumventing Defenses For Distributed Learning," in *NeurIPS*, 2019.

[6] J. Bernstein, J. Zhao, K. Azizzadenesheli, and A. Anandkumar, "signSGD with Majority Vote is Communication Efficient and Fault Tolerant," in *ICLR*, 2018.

[7] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *ICML*, 2019.

[8] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," in *ICML*, 2012.

[9] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *Pattern Recognition*, 2018.

[10] P. Blanchard, R. Guerraoui, J. Stainer *et al.*, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *NeurIPS*, 2017.

[11] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecný, S. Mazzocchi, B. McMahan, T. V. Overveldt, D. Petrou, D. Ramage, and J. Roselander, "Towards federated learning at scale: System design," in *MLSys*, 2019.

[12] M. H. Brendan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," in *ICLR*, 2018.

[13] S. Caldas, P. Wu, T. Li, J. Konečnỳ, H. B. McMahan, V. Smith, and A. Talwalkar, "LEAF: A benchmark for federated settings," *arXiv:1812.01097*, 2018.

[14] X. Cao, J. Jia, and N. Z. Gong, "Provably Secure Federated Learning against Malicious Clients," in *AAAI*, 2021.

[15] H. Chang, V. Shejwalkar, R. Shokri, and A. Houmansadr, "Cronus: Robust and Heterogeneous Collaborative Learning with Black-Box Knowledge Transfer," *arXiv:1912.11279*, 2019.

[16] L. Chen, H. Wang, Z. Charles, and D. Papailiopoulos, "Draco: Byzantine-resilient distributed training via redundant gradients," in *ICML*, 2018.

[17] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," *arXiv:1712.05526*, 2017.

[18] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik, "EMNIST: Extending MNIST to handwritten letters," in *IJCNN*, 2017.

[19] "CS231n: Convolutional Neural Networks for Visual Recognition," https://cs231n.github.io/optimization-2/#grad, 2021.

[20] D. Data and S. Diggavi, "Byzantine-resilient SGD in high dimensions on heterogeneous data," *arXiv:2005.07866*, 2020.

[21] E.-M. El-Mhamdi, R. Guerraoui, A. Guirguis, and S. Rouault, "Sgd: Decentralized byzantine resilience," *arXiv:1905.03853*, 2019.

[22] "Facebook has shut down 5.4 billion fake accounts this year," https://www.cnn.com/2019/11/13/tech/facebook-fake-accounts/index.html, 2019.

[23] M. Fang, X. Cao, J. Jia, and N. Z. Gong, "Local Model Poisoning Attacks to Byzantine-Robust Federated Learning," in *USENIX*, 2020.

[24] "Google Workshop on Federated Learning and Analytics," https://docs.google.com/document/d/1dWzVeFLrPinonQMauxIo0oI-Vbvqup5cZzgdPXvu97Y/edit#heading=h.7dsxad3c3nf7, 2020.

[25] S. Fu, C. Xie, B. Li, and Q. Chen, "Attack-resistant federated learning with residual-based reweighting," *arXiv:1912.11464*, 2019.

[26] C. Fung, C. J. Yoon, and I. Beschastnikh, "The limitations of federated learning in sybil settings," in *RAID*, 2020.

[27] M. Goldblum, D. Tsipras, C. Xie *et al.*, "Dataset Security for Machine Learning: Data Poisoning, Backdoor Attacks, and Defenses," *arXiv:2012.10544*, 2020.

[28] "Google Play Protect," https://developers.google.com/android/play-protect, 2021.

[29] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. D. Tygar, "Adversarial machine learning," in *AISec*, 2011.

[30] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li, "Manipulating machine learning: Poisoning attacks and countermeasures against regression learning," *39th IEEE Symposium on S&P*, 2018.

[31] M. S. Jere, T. Farnan, and F. Koushanfar, "A taxonomy of attacks on federated learning," *IEEE Security & Privacy*, 2020.

[32] P. Kairouz, H. B. McMahan, B. Avent *et al.*, "Advances and open problems in federated learning," *arXiv:1912.04977*, 2019.

[33] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *NIPS Workshop on Private Multi-Party ML*, 2016.

[34] A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009.

[35] Y. LeCun, L. Bottou, Y. Bengio *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, 1998.

[36] L. Li, W. Xu, T. Chen, G. B. Giannakis, and Q. Ling, "RSA: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets," in *AAAI*, 2019.

[37] T. Li, S. Hu, A. Beirami, and V. Smith, "Ditto: Fair and robust federated learning through personalization," in *ICML*, 2021.

[38] T. Lin, L. Kong, S. U. Stich, and M. Jaggi, "Ensemble distillation for robust model fusion in federated learning," in *NeurIPS*, 2020.

[39] H. Ludwig, N. Baracaldo, G. Thomas *et al.*, "IBM Federated Learning: An Enterprise Framework White Paper v0.1," *arXiv:2007.10987*, 2020.

[40] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*, 2017.

[41] E. M. E. Mhamdi, R. Guerraoui, and S. Rouault, "The Hidden Vulnerability of Distributed Learning in Byzantium," in *ICML*, 2018.

[42] T. Minka, "Estimating a Dirichlet distribution," 2000.

[43] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli, "Towards poisoning of deep learning algorithms with back-gradient optimization," in *AISec*, 2017.

[44] L. Muñoz-González, B. Pfitzner, M. Russo, J. Carnerero-Cano, and E. C. Lupu, "Poisoning attacks with generative adversarial nets," *arXiv:1906.07773*, 2019.

[45] A. Newell, R. Potharaju, L. Xiang, and C. Nita-Rotaru, "On the practicality of integrity attacks on document-level sentiment analysis," in *AISec*, 2014.

[46] "Billion Passwords Stolen: Change All of Yours, Now!" https://www.nbcnews.com/tech/security/billion-passwords-stolen-change-all-yours-now-n174321, 2014.

[47] "Hackers Expose 8.4 Billion Passwords Post them Online in Possibly Largest Dump of Passwords Ever," https://www.thegatewaypundit.com/2021/06/hackers-expose-8-4-billion-passwords-post-online-possibly-largest-dump-passwords-ever/, 2014.

[48] "26 million stolen passwords found online — see if you're affected," https://www.tomsguide.com/news/mystery-malware-info-stealer, 2021.

[49] M. Paulik, M. Seigel, H. Mason *et al.*, "Federated Evaluation and Tuning for On-Device Personalization: System Design & Applications," *arXiv:2102.08503*, 2021.

[50] K. Pillutla, S. M. Kakade, and Z. Harchaoui, "Robust aggregation for federated learning," *arXiv:1912.13445*, 2019.

[51] "Acquire Valued Shoppers Challenge at Kaggle," https://www.kaggle.com/c/acquire-valued-shoppers-challenge/data, 2019.

[52] S. J. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečnỳ, S. Kumar, and H. B. McMahan, "Adaptive Federated Optimization," in *ICLR*, 2020.

[53] "SafetyNet Attestation API," https://developer.android.com/training/safetynet/attestation, 2021.

[54] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein, "Poison frogs! targeted clean-label poisoning attacks on neural networks," in *NeurIPS*, 2018.

[55] V. Shejwalkar and A. Houmansadr, "Manipulating the Byzantine: Optimizing Model Poisoning Attacks and Defenses for Federated Learning," in *NDSS*, 2021.

[56] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.

[57] G. Sun, Y. Cong, J. Dong, Q. Wang, L. Lyu, and J. Liu, "Data poisoning attacks on federated machine learning," *IEEE IoT Journal*, 2021.

[58] Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan, "Can you really backdoor federated learning?" *NeurIPS FL Workshop*, 2019.
[59] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," in *ESORICS*, 2020.
[60] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *40th IEEE Symposium on S&P*, 2019.
[61] H. Wang, K. Sreenivasan, S. Rajput, H. Vishwakarma, S. Agarwal, J.-y. Sohn, K. Lee, and D. Papailiopoulos, "Attack of the tails: Yes, you really can backdoor federated learning," in *NeurIPS*, 2020.
[62] "Utilization of FATE in Risk Management of Credit in Small and Micro Enterprises," https://www.fedai.org/cases/utilization-of-fate-in-risk-management-of-credit-in-small-and-micro-enterprises/, 2019.
[63] C. Wu, X. Yang, S. Zhu, and P. Mitra, "Mitigating backdoor attacks in federated learning," *arXiv:2011.01767*, 2020.
[64] H. Xiao, H. Xiao, and C. Eckert, "Adversarial label flips attack on support vector machines," in *ECAI*, 2012.
[65] H. Xiao, B. Biggio, B. Nelson, H. Xiao, C. Eckert, and F. Roli, "Support vector machines under adversarial label contamination," *Neurocomputing*, 2015.
[66] C. Xie, M. Chen, P.-Y. Chen, and B. Li, "CRFL: Certifiably Robust Federated Learning against Backdoor Attacks," in *ICML*, 2021.
[67] C. Xie, K. Huang, P.-Y. Chen, and B. Li, "DBA: Distributed backdoor attacks against federated learning," in *ICLR*, 2019.
[68] C. Xie, O. Koyejo, and I. Gupta, "Generalized byzantine-tolerant sgd," *arXiv:1802.10116*, 2018.
[69] C. Yang, Q. Wu, H. Li, and Y. Chen, "Generative poisoning attack method against neural networks," *arXiv:1703.01340*, 2017.
[70] D. Yin, Y. Chen, K. Ramchandran, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *ICML*, 2018.
[71] T. Yu, E. Bagdasaryan, and V. Shmatikov, "Salvaging federated learning by local adaptation," *arXiv:2002.04758*, 2020.

APPENDIX

A. Related Work

1) Targeted and Backdoor Attacks: Section IV-A discusses all state-of-the-art untargeted attacks in detail. Below, we discuss existing works on targeted and backdoor attacks.

Targeted attacks [7], [58], [59] aim to make the global model misclassify a specific set of samples at test time. Bhagoji et al. [7] aimed to misclassify a *single sample* and proposed a model poisoning attack based on alternate minimization to make poisoned update look similar to benign updates. [7] shows that their attack, with a single attacker, can misclassify a single sample with 100% success against the non-robust Average AGR. try Sun et al. [58] investigated constrain-and-scale attack [3] with the aim to misclassify all samples of a few victim FL clients. Tolpegin et al. [23], [59] investigated targeted data poisoning attacks when compromised clients compute their updates by mislabeling the target samples.

Backdoor attacks [3], [61], [67] aim to make the global model misclassify the samples with adversary-chosen backdoor trigger. Backdoor attacks are *semantic*, if the trigger is naturally present in samples [3], [61] and *artificial* if the trigger needs to manually added at test time [67]. Bagdasaryan at al. [3] demonstrate a constrain-and-scale attack against simple Average AGR to inject semantic backdoor in the global model. They show that their attacks achieve accuracy of >90% on backdoor task in a next word prediction model. Wang et al. [61] propose data and model poisoning attacks to inject backdoor to misclassify out-of-distribution samples. Xie et al. [67] show how multiple colluding clients can distribute backdoor trigger to improve the stealth of poisoned updates. Backdoor (as well as targeted) attacks can be further divided in *specific-label* and *arbitrary-label* attacks. For a backdoored test sample, specific-label attack aims to misclassify it to a specific target class, while arbitrary-label attack aims to misclassify it to any class.

Note that, trivial extensions of the targeted and backdoor attack algorithms to mount untargeted attacks cannot succeed, because untargeted attacks aim at affecting almost *all* FL clients and test inputs. For instance, a simple label flipping based data poisoning [61] can insert a backdoor in FL with state-of-the-art defenses. However, such label flipping based untargeted poisoning attacks have no effect even on unprotected FL (Section V-A).

2) Existing Defenses Against Targeted and Backdoor Attacks: In Section II-B, we discuss the defenses against untargeted poisoning in detail. Here, we review existing defenses against targeted and backdoor attacks. FoolsGold [26] identifies clients with similar updates as attackers, but incur very high losses in performances as noted in [25]. Sun et al. [58] investigate efficacy of norm-bounding to counter targeted poisoning and, as we will show, is also effective against untargeted poisoning. CRFL [66] counters backdoor attacks by providing certified accuracy for a given test input, but incurs large losses in FL performance (Table I). Defenses based on pruning techniques [60], [63] remove parts of model that are affected by targeted/backdoor attacks, and hence cannot be used against untargeted attacks which affect the entire model.

Algorithm 1 Our PGA model poisoning attack algorithm

Input: $\nabla_{\{i\in[n']\}}$, θ^g, f_{agr}, D_p
$\tau = \frac{1}{n'}\sum_{i\in[n']}\|\nabla_i\|$ ▷*Compute norm threshold*
▷*τ is given for norm-bounding AGR*
$\theta' \leftarrow A_{\mathsf{SGA}}(\theta^g, D_p)$ ▷*Update using stochastic gradient ascent*
$\nabla' = \theta' - \theta^g$ ▷*Compute poisoned update*
$\nabla' = f_{\mathsf{project}}(f_{\mathsf{agr}}, \nabla', \tau, \nabla_{\{i\in[n']\}})$ ▷*Scale ∇' appropriately*
Output ∇'

Algorithm 2 The projection function (f_{project}) of our PGA from Section IV-B3.

Input: f_{agr}, ∇', τ, $\nabla_{\{i\in[n']\}}$
$d^* = 0$ ▷*Initialize maximum deviation*
$\gamma^* = 1$ ▷*Optimal scaling factor that maximizes deviation in (1)*
$\nabla' = \frac{\nabla'\times\tau}{\|\nabla'\|}$ ▷*Scale ∇' to have norm τ*
$\nabla^b = f_{\mathsf{avg}}(\nabla_{\{i\in[n']\}})$ ▷*Compute reference benign update*
for $\gamma \in [1, \Gamma]$ **do**
  $\nabla'' = \gamma \cdot \nabla'$
  $d = \|f_{\mathsf{agr}}(\nabla''_{\{i\in[m]\}}, \nabla_{\{i\in[n']\}}) - \nabla^b\|$
  $\gamma^* = \gamma$ **if** $d > d^*$ ▷*Update optimal γ*
  $\gamma = \gamma + \delta$ ▷*Update γ*
end for
Output $\gamma^* \cdot \nabla'$

B. Missing details of our data and model poisoning attacks from Sections IV-B2 and IV-B3

1) Missing data poisoning attack methods: ***Multi-krum.*** Following [55], our attack aims to maximize the number of poisoned updates in the selection set (S) of Multi-krum AGR

(Section II-C3). As the size of S is fixed, maximizing the number of poisoned updates in S implicitly means minimizing the number of benign updates. This objective is formalized as:

$$\underset{D_p \subset D'_p}{\text{argmax}} \quad m' = |\{\nabla \in \nabla'_{\{i\in[m]\}} | \nabla \in S\}| \tag{3}$$

where D'_p is all the available labels flipped data and m' is the final number of poisoned updates in S of Multi-krum.

We solve (3) based on an observation: In Figure 2-(d) we vary $|D_p|$ and plot the fraction of corresponding poisoned updates that Multi-krum selects. Let $|D|_{\text{avg}}$ be the average dataset size of benign clients, e.g., $|D|_{\text{avg}}$ is 23.7 for FEMNIST. Note from Figure 2-(d) that, even for $|D_p|$ slightly higher than $|D|_{\text{avg}}$, Multi-krum easily discards most of the poisoned updates. Only when $|D_p|$ is small ($\sim$10), Multi-krum selects most of the poisoned updates. Hence, we sample $D_p \subset D'_p$, where we vary $|D_p| \in [0.5 \cdot |D|_{\text{avg}}, 3 \cdot |D|_{\text{avg}}]$, and check the poisoning impact of D_p on Multi-krum; to reduce variance, we repeat this 10 times for each $|D_p|$. We report the results for D_p with the maximum poisoning impact.

Trimmed-mean. For Trimmed-mean AGR (Section II-C4), we use the objective in (2), but it is cumbersome to solve it directly. Hence, similar to our attacks on Average and Norm-bounding AGRs, we use large $|D_p|$ for poisoned data on each of the compromised clients. Our approach is based on the observation in Figure 2-(c): The higher the $|D_p|$ (obtained using DLF/SLF strategies), the higher the Trimmed-mean objective value, i.e., $\|\nabla^p - \nabla^b\|$.

2) Missing model poisoning attack methods: ***Multi-krum.*** Similar to our DPA (Section IV-B2), the objective of our MPA on Multi-krum is to maximize the number of poisoned updates in the selection set S. We aim to find a scaling factor γ for ∇' such that maximum number of $\nabla'' = \gamma\nabla'$ are selected in S. This is formalized below:

$$\underset{\gamma^* \in \mathbb{R}}{\text{argmax}} \quad m = |\{\nabla \in \nabla''_{\{i\in[m]\}} | \nabla \in S\}| \tag{4}$$

To solve the optimization in (4), our f_{project} searches for the maximum γ in a pre-specified range $[1, \Gamma]$ such that Multi-krum selects all the scaled poisoned updates. Specifically, in Algorithm 2, instead of computing the deviation (line-8), we compute the number of ∇'' selected in S and update γ^* if S has all of ∇''s.

Trimmed-mean. Here, we directly plug Trimmed-mean algorithm in Algorithm 2 (line-8). Our attack is similar to that of [55], but instead of using one of several perturbation vectors, ω's, we use stochastic gradient ascent to tailor ω to the entire FL setting (e.g., θ^g, data, optimizer, etc.) to improve the attack impact.

C. Experimental setup

Real-world FL datasets [1], [49] are proprietary and cannot be publicly accessed. Hence, we follow the literature on untargeted poisoning in FL [5], [23], [55], [59] and focus on image and categorical datasets. But, we ensure that our setup embodies the production FL [32], e.g., by using large number of clients with extremely non-iid datasets.

1) Datasets and Model Architectures: **FEMNIST [13], [18]** is a character recognition classification task with 3,400 clients, 62 classes (52 for upper and lower case letters and 10 for digits), and 671,585 grayscale images. Each client has data of her own handwritten digits or letters. Considering the huge number of clients in real-world cross-device FL (up to 10^{10}), we further divide each of the clients' data in $p \in \{2, 5, 10\}$ non-iid parts using Dirichlet distribution [42] with $\alpha = 1$. Increasing the Dirichlet distribution parameter, α, generates more iid datasets. Unless specified otherwise, we set $p = 10$, i.e., the total number of clients is 34,000. We use LeNet [35] architecture.

CIFAR10 [34] is a 10-class classification task with 60,000 RGB images (50,000 for training and 10,000 for testing), each of size 32×32. Unless specified otherwise, we consider 1,000 total FL clients and divide the 50,000 training data using Dirichlet distribution [42] with $\alpha = 1$. We use VGG9 architecture with batch normalization [56].

Purchase [51] is a classification task with 100 classes and 197,324 binary feature vectors each of length 600. We use 187,324 of total data for training and divide it among 5,000 clients using Dirichlet distribution with $\alpha = 1$. We use validation and test data of sizes 5,000 each. We use a fully connected network with layer sizes {600, 1024, 100}.

2) Details of Federated learning and attack parameters: For FEMNIST, we use 500 rounds, batch size, $\beta = 10$, $E = 5$ local training epochs, and in the e^{th} round use SGD optimizer with a learning rate $\eta = 0.1 \times 0.995^e$ for local training; we select $n = 50$ clients per round and achieve baseline accuracy A_θ=82.4% with N=34,000 clients. For CIFAR10, we use 1,000 rounds, $\beta = 8$, $E = 2$, and in the e^{th} round use SGD with momentum of 0.9 and $\eta = 0.01 \times 0.9995^e$; we use $n = 25$ and achieve A_θ=86.6% with N=1,000. For Purchase, we use 500 rounds, $\beta = 10$, $E = 5$, and in the e^{th} round use SGD with $\eta = 0.1 \times 0.999^e$; we use $n = 25$ and achieve A_θ=81.2% with N=5,000.

We generate large poisoned data D_p required for our DPAs (Section IV-B2) by combining the dataset of compromised clients and adding Gaussian noise to their features. We round the resulting feature for categorical Purchase dataset.

D. Explanations of effects of $|D|_{\text{avg}}$ from Section V-C2

At M=1%, I_θ's of STAT-OPT on CIFAR10 + Normb reduce with increase in $|D|_{\text{avg}}$. This is because, increasing $|D|_{\text{avg}}$ improves the quality of updates of benign clients, but does not improve the attacks. Hence, when the benign impact of benign updates overpowers the poisoning impact of poisoned updates, I_θ's reduce.

On the other hand, I_θ's of any attacks on FEMNIST with robust AGRs do not change with varying $|D|_{\text{avg}}$. This is because, FEMNIST is an easy task, and therefore, the presence of compromised clients does not affect the global models.

Interestingly, I_θ of MPAs on CIFAR10 with Average AGR *increases* with $|D|_{\text{avg}}$. This is because, due to the difficulty of CIFAR10 task, MPAs on CIFAR10 with Average AGR

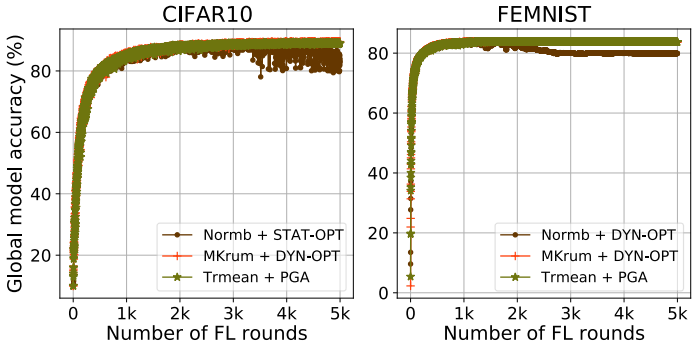


Figure 6: Even with a very large number of FL rounds (5,000), the state-of-the-art model poisoning attacks with M=0.1% cannot break the robust AGRs (Section V-B).

Table V: The architecture of the surrogate model that we use to emulate the unknown architecture setting (Section V-C4).

Layer name	Layer size
Convolution + Relu	$5 \times 5 \times 32$
Max pool	2×2
Convolution + Relu	$5 \times 5 \times 64$
Max pool	2×2
Fully connected + Relu	1024
Softmax	62

are very effective and when the server selects even a single compromised client, it completely corrupts the global model.

E. Miscellaneous figures

Below, we provide all the missing figures and the corresponding sections in main paper.

- Figure 6 for Section V-B shows the impacts of strongest of model poisoning attacks on robust AGRs over a very large number of FL rounds.
- Figure 7 for Section V-C4 shows impact of unknown architecture on our state-of-the-art data poisoning attacks from Section IV-B2. Table V shows the convolutional neural network architecture that the adversary uses as a substitute to the true LeNet architecture.
- Figure 8 for Section V-D shows impacts of data poisoning attacks on cross-silo FL.
- Figure 10 for Section V-C3 shows impacts of poisoning attacks for increasing the number of clients selected in each FL round.
- Figures 9 and 11 for Section V-C2 show the attack impacts and accuracy of the global model, respectively, when the average size of benign clients' local data increases.
- Figure 12 for Section V-C2 shows attack impacts (on the left y-axes) and global model accuracy (on the right y-axes) for Multi-krum and Trimmed-mean robust AGRs for CIFAR10 and FEMNIST datasets.

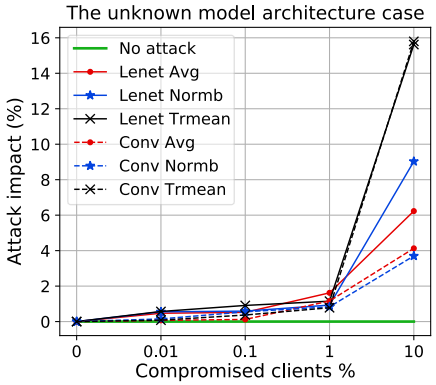


Figure 7: As discussed in Section V-C4, impacts of the DPA-DLF attack from Section IV-B2 reduce if the architectures of the surrogate and the global model are different.

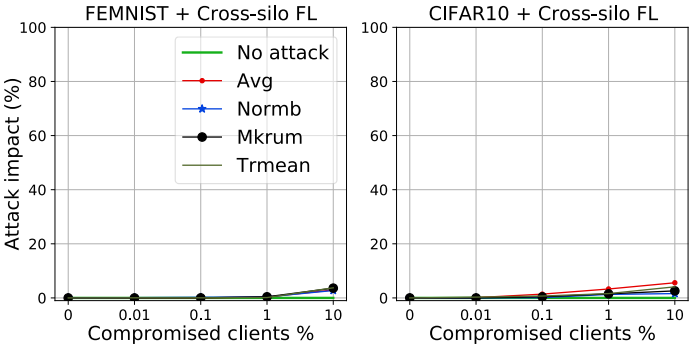


Figure 8: All data poisoning attacks have negligible impacts on cross-silo FL, when compromised clients are concentrated in a few silos or distributed uniformly across silos (Section V-D).

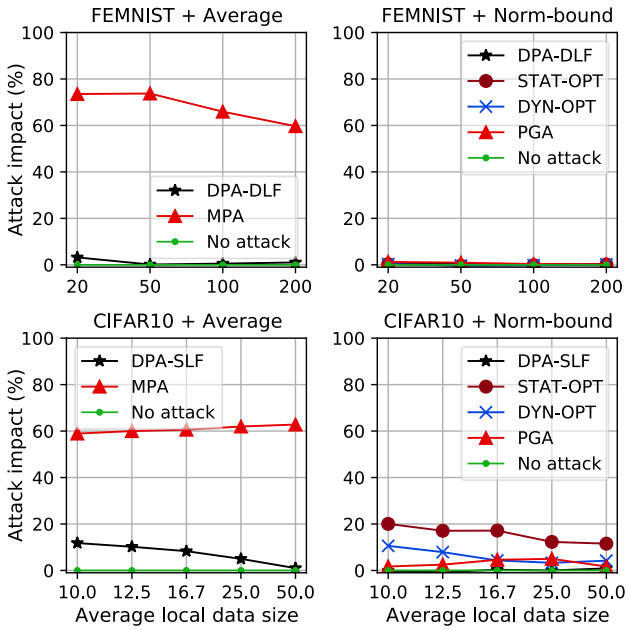


Figure 9: With 1% compromised clients, *increasing* $|D|_{\text{avg}}$ *has no clear pattern of effects of on attack impacts, but it increases the global model accuracy* as shown in Figure 11. Figure 12 shows the plots of attack impacts and the global model accuracy for Multi-krum and Trimmed-mean AGRs.

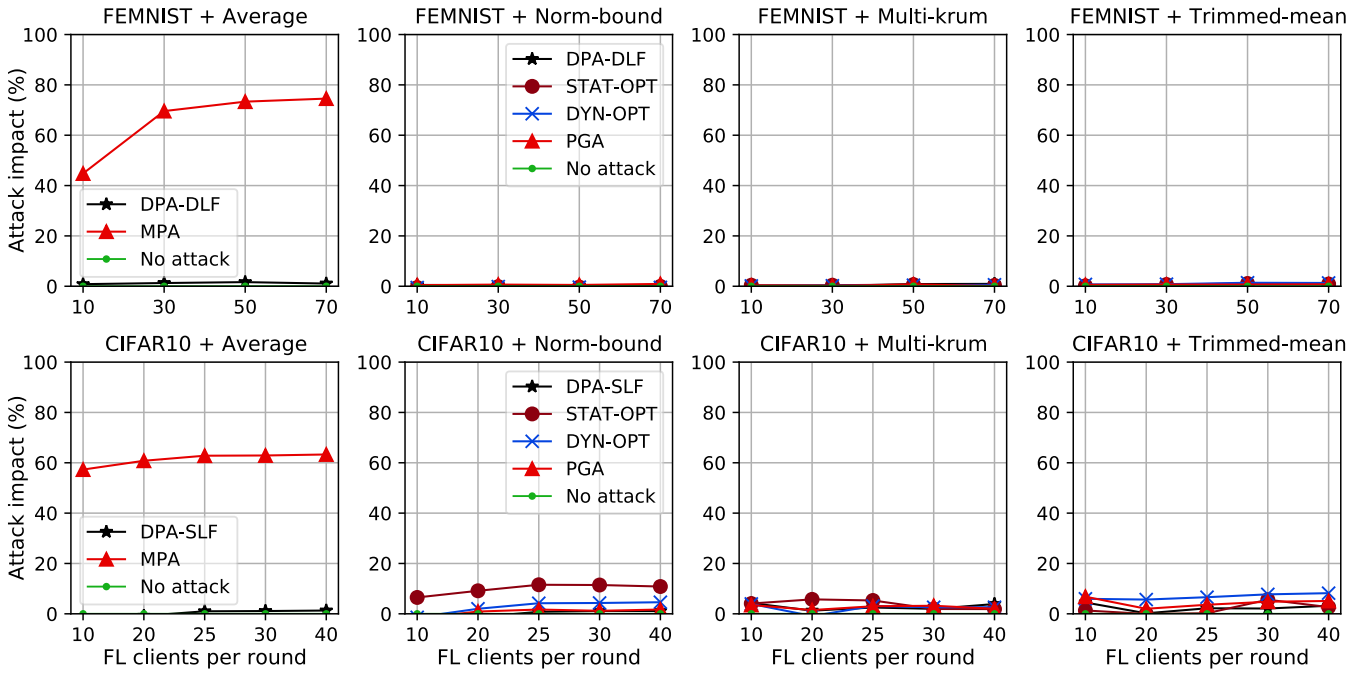


Figure 10: As discussed in Section V-C3, the number of clients, n, chosen in each FL round has no noticeable effect on the attack impacts, with the exception of model poisoning on Average AGR. We use $M = 1\%$ of compromised clients.

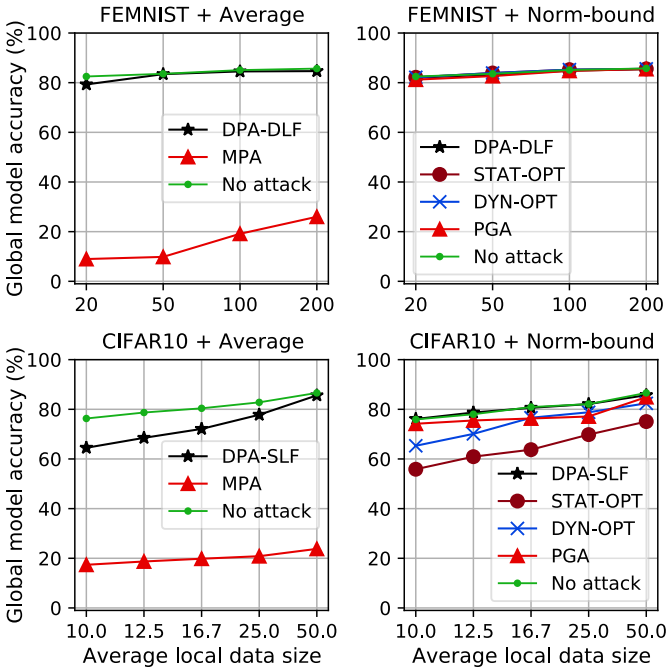


Figure 11: Effect on **the accuracy of global models** of the average of local dataset sizes, $|D|_{\text{avg}}$, of the benign clients, with 1% compromised clients. As discussed in Section V-C2, *increasing* $|D|_{\text{avg}}$ *increases the accuracy of the global models*.

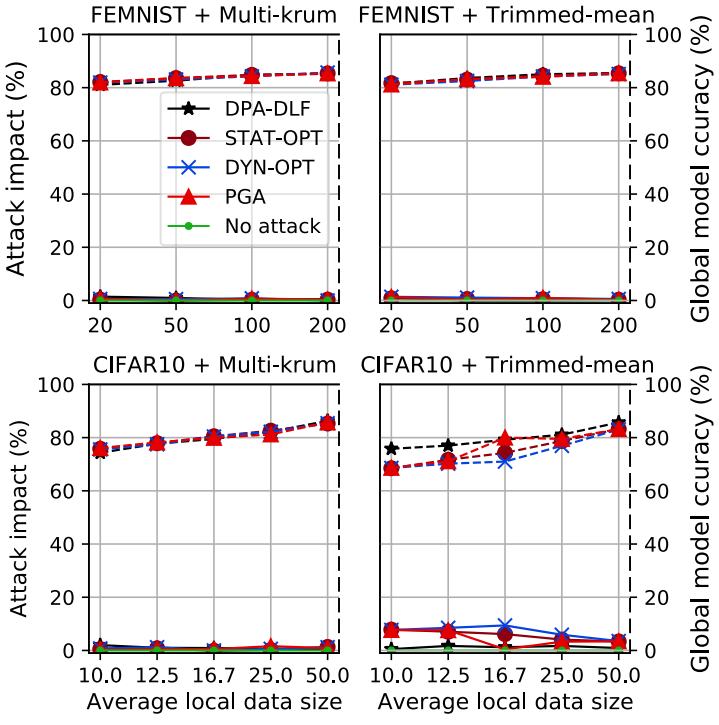


Figure 12: We make observations similar to Average and Norm-bound AGRs (Figures 9, 11 in Section V-C2) for Multi-krum and Trimmed-mean about the effect of $|D|_{\text{avg}}$ on the attack impacts (left y-axes, solid lines) and on the global model accuracy (right y-axes, dotted lines), with M=1%. All y-axes are from 0 to 100.