RoFL: Robustness of Secure Federated Learning

Hidde Lycklama*, Lukas Burkhalter*, Alexander Viand, Nicolas Küchler, Anwar Hithnawi

ETH Zurich

Abstract—Even though recent years have seen many attacks exposing severe vulnerabilities in Federated Learning (FL), a holistic understanding of what enables these attacks and how they can be mitigated effectively is still lacking. In this work, we demystify the inner workings of existing (targeted) attacks. We provide new insights into why these attacks are possible and why a definitive solution to FL robustness is challenging. We show that the need for ML algorithms to memorize tail data has significant implications for FL integrity. This phenomenon has largely been studied in the context of privacy; our analysis sheds light on its implications for ML integrity. We show that certain classes of severe attacks can be mitigated effectively by enforcing constraints such as norm bounds on clients' updates. We investigate how to efficiently incorporate these constraints into secure FL protocols in the single-server setting. Based on this, we propose RoFL, a new secure FL system that extends secure aggregation with privacy-preserving input validation. Specifically, RoFL can enforce constraints such as L_2 and L_∞ bounds on high-dimensional encrypted model updates.

1. Introduction

Recent years have seen a surge of interest in collaborative, secure machine learning (ML) paradigms. These paradigms allow machine learning models to be trained without requiring direct access to training data, thus alleviating some of the risks associated with the large-scale collection of sensitive data. Federated Learning (FL) is a prominent form of collaborative ML that has recently emerged and is already being used in practice to train models for a variety of privacy-sensitive applications [16, 17, 68, 70, 74, 75]. In FL, the training process is distributed across a set of participants who collaborate to train a joint global model through an iterative process that aggregates locally trained updates. Although these updates contain less information than the underlying training data, they can still leak sensitive information. In fact, recent efforts have shown that by observing gradients submitted by the clients, one can reconstruct sensitive data from clients' local datasets [11, 42, 87]. Secure FL systems try to address this issue by employing secure aggregation [13, 34, 88] to protect clients' individual gradient updates. Here, clients send encrypted updates to the server, which can recover only the aggregated joint model update (c.f. Fig. 1). While FL provides many privacy

---

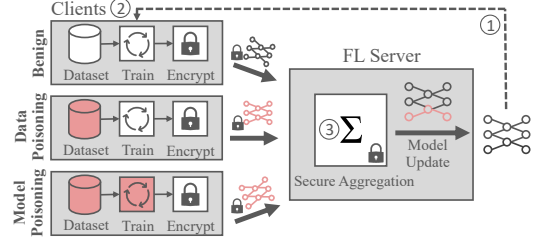* *These authors contributed equally to this work.*

Figure 1: Secure FL pipeline with malicious clients.

benefits, it introduces new ML robustness issues and exacerbates existing ones. Since FL systems are open to a multitude of participants, any of whom could be compromised, the training algorithm becomes susceptible to contributions from compromised clients. In addition to manipulating the joint model by poisoning their local training data, malicious clients can also directly submit manipulated model updates. The possibility of such *model poisoning* attacks poses a unique challenge for FL systems. A flurry of work, starting shortly after the introduction of FL, has exposed the severity of both kinds of attacks in FL [35, 85]. We have seen two forms of attack goals: (i) untargeted attacks, which can easily prevent or slow learning [9, 86]. While undesirable, these predominately impact availability and are eventually detected as the server can observe that the model accuracy is not improving. Recent work by [73] studied this type of attacks and their mitigation in depth. The second form of attacks is (ii) targeted attacks [5, 7, 82, 84]. In targeted attacks, the attacker aims to integrate a backdoor into the model through specific inputs and thus cause the global model to misbehave. These pose a severe threat to model integrity and thus to the robustness of FL. These have been studied to a lesser extent, and we address this gap in this paper.

A variety of defenses has been proposed against both targeted and untargeted attacks. These defenses rely primarily on anomaly detection [40, 76] or byzantine-robust estimators [9, 25, 61, 67, 86] introducing non-linear aggregation rules with audits to filter malicious updates. However, these existing solutions become prohibitively expensive when transferred to the *secure* setting, as they would require general-purpose secure computation techniques (e.g., generic MPC between all clients) to express them. Hence, there is a need for practical defenses that are compatible with existing secure aggregation protocols used in FL sys-

tems. Prior work has identified enforcing norm bounds on individual client updates as a computationally simple yet promising defense in practical FL setups [60, 73, 79]. This makes them a prime candidate for lifting to the secure setting because they can be expressed without requiring general-purpose MPC. Norm bounds, while not a panacea, have already been shown to prevent untargeted poisoning attacks in real-world adversarial scenarios [73]. However, to what extent norm bounds can be effective against targeted attacks is unclear. Recent work has come to apparently conflicting conclusions in this regard [5, 82]. Reconciling these observations and working towards a remedy for the underlying issues requires a deeper understanding of the inner workings of these attacks and the factors that enable them. Our work is the first to analyze robustness for single-server secure FL more holistically and, as a result, reconciles this apparent contradiction. We conclude that, while they have clear limits, norm bounds would indeed be an attractive robustness solution. However, this hinges on them being efficiently realizable in the secure setting. Therefore, we investigate how to efficiently incorporate such constraints into secure FL protocols in the single-server setting. Based on this, we propose a new secure FL system that extends secure aggregation with privacy-preserving input validation. Specifically, our system can enforce constraints such as L_2 and L_∞ bounds on high-dimensional encrypted model updates. As a result, this paper provides two contributions:

① **Understanding FL Robustness.** Even though recent years have seen many targeted poisoning attacks exposing severe vulnerabilities in FL, a holistic understanding of what enables these attacks and how they can be mitigated effectively is still lacking. To improve understanding of FL robustness, we demystify the inner workings of existing attacks. We empirically analyze why these attacks are possible and why a definitive solution to FL robustness is challenging. We show that although some attacks are still possible under a norm bound, enforcing norm bounds significantly reduces the attack surface of secure FL. In particular, norm bounds are effective in preventing a class of highly practical attacks in which an attacker can completely replace the model by controlling just a single client. We also show that the need for ML algorithms to memorize tail subpopulations has significant implications for ML integrity. This phenomenon has largely been studied in the context of privacy [36, 37]; in our analysis, we shed light on its implications for ML robustness. For attacks exploiting model memorization, we show that norm bounds are of limited effectiveness. Nevertheless, while norm bounds are not sufficient to prevent all attacks, they increase robustness immensely in practical settings. We make our framework for analyzing FL robustness available online [1] and hope our analysis can help practitioners better assess the implications and risks of open federated learning systems.

② **Secure Aggregation with Input Validation.** We present RoFL, a secure FL system that enables constraints to be expressed and enforced on high-dimensional encrypted

model updates to defend against attacks by malicious participants. RoFL augments existing secure FL systems [6] with zero-knowledge proofs that allow the server to enforce and verify constraints on client updates, e.g., norm bounds. In this work, we tackle the *single-server secure FL* setting proposed in Bell et al. [6], which poses unique compatibility and scalability challenges. Assuming a setting with multiple non-colluding servers (e.g., Prio [27]) enables more efficient approaches, but the required trust assumptions are hard to materialize in practice. Therefore, we focus on the (de-facto standard) single-server setting, which allows RoFL to be compatible with existing FL deployment scenarios. The key challenge of this setting is the *scale* of the problem domain, both in the number of clients that participate in each round and also the high dimensionality of the aggregation inputs, which correspond to the model size. Therefore, scalability is a key consideration in the design of a protocol for this setting. We construct our system from efficient additively homomorphic commitments which are compatible with the existing masking-based secure aggregation approaches used in secure FL systems. This approach allows us to construct the norm bounds with efficient range proofs that operate directly on the homomorphic commitments. We realize them using a discrete-log based zero-knowledge proof system (Bulletproofs [18]) that offers proof sizes that are logarithmic in the number of range proofs and also support batched verification, allowing our protocol to scale efficiently to large model sizes. Additionally, we introduce several optimizations at the ML layer that allow us to reduce the number of cryptographic checks needed. We implement and evaluate RoFL, showing that it scales to the model sizes used in real-world FL deployments. We make RoFL's prototype available online. [2]

2. Analysis of FL Robustness

FL Background. In FL, the server coordinates the training of a model $f_{\mathbf{w}_G}$ with weights $\mathbf{w}_G \in \mathbb{R}^\ell$, where ℓ is the number of parameters, based on the datasets D_i for $i \in \{1...n\}$ held locally by a set of clients N with size $n = |N|$. At each round t of the iterative learning process, the server selects a subset of m clients and broadcasts the current global model $\mathbf{w}_G^t$ to them. Each selected client fine-tunes the model on their local training data D_i using a training algorithm such as Stochastic Gradient Descent (SGD), producing a local model $\mathbf{w}_i^{t+1}$. Each client then sends its update $\triangle \mathbf{w}_i^{t+1} := \mathbf{w}_i^{t+1} - \mathbf{w}_G^t$ to the server. The server aggregates the updates using an aggregation function F, i.e., $\triangle \mathbf{w}_{\text{agg}}^{t+1} = F(\mathbf{w}_{i \in [m]}^{t+1})$. For FedAvg [13, 57], F_{avg} is defined as a weighted aggregation of updates. The server incorporates the aggregated updates into the next global model, i.e., $\mathbf{w}_G^{t+1} = \mathbf{w}_G^t + \eta \triangle \mathbf{w}_{\text{agg}}^{t+1}$. This process repeats until the server ends the training process.

What makes FL robustness challenging? FL is inherently open to a multitude of participants, often in the orders of hundreds of thousands. Consequently, training algorithms in

---

such settings can be susceptible to contributions from compromised clients. Besides being susceptible to the known security and privacy vulnerabilities encountered in typical ML settings [8, 26, 48, 54], FL exposes new attack surfaces. Moreover, several FL characteristics elevate the impact of attacks known in the centralized setting, such as backdoor attacks through data poisoning [5, 7, 82, 84]. These unique challenges are attributed to three characteristics: *(i) Open Nature:* In FL, the training process is open and often involves a large number of participants, among whom any participant can act maliciously. *(ii) Attackers' Capabilities:* In conventional centralized ML settings, an adversary can instigate targeted attacks only by data poisoning. In the FL setting, however, an attacker can do so by directly manipulating model updates to enhance their impact, allowing more effective attacks (i.e., model poisoning). *(iii) Active Attackers:* An attacker that controls a subset of clients can observe and influence the model behavior over multiple training rounds, which can allow for stronger forms of adaptive attacks.

Analysis Methodology. In this section, we extensively investigate FL robustness in the presence of malicious participants and answer to what extent norm bounding can effectively defend against malicious attacks. We primarily focus on *targeted* attacks where the adversary compromises the global model's integrity by causing it to misclassify a target set of samples. Recent work [73] has already addressed untargeted attacks in-depth and showed the high effectiveness of norm bounding against untargeted attacks in practice. Therefore, we refer to [73] for the analysis of untargeted attacks and only provide a few additional insights not covered in the existing work in Appendix A.

Initially, proposed targeted attacks [5, 7] for FL relied heavily on update scaling to succeed. Intuitively, attacks relying on scaling can be mitigated effectively with norm bounds [73, 79] that limit participants' contributions to the model to prevent a single participant from overpowering the aggregation process. More recently, however, attacks have been shown to be successful even in the presence of norm bounds. Consequently, this has called the ability of norm bounds to mitigate targeted attacks into question. Understanding to what extent and why norm bounds are (in)effective requires a deeper understanding of the factors that enable them.

In this work, we study the different attacks from the perspective of the targets they attack and relate this to insights into the nature of the input data distribution of common deep learning tasks. In particular, it has been widely observed that modern datasets of natural images and text data follow long-tailed distributions, which can be seen as mixtures of *subpopulations* [36, 37, 81, 91]. The long-tailed nature implies that a significant fraction of the samples in the dataset belong to rare subpopulations, such as the images of cars from unusual angles in Fig. 2. We use this to categorize the inputs that are targeted by attacks and then study attacks on two types of subpopulations representing opposing ends of the input distribution. The first type is *prototypical* targets, i.e., subpopulations of samples that frequently occur in the
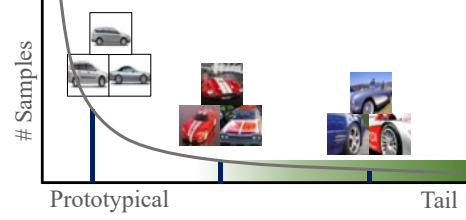

Figure 2: Modern datasets can be modeled as a long-tailed mixture of subpopulations, with the tail consisting of subpopulations of rare samples [81, 91].

dataset. The second type is *tail* targets, i.e., subpopulations that sit on the long tail of the distribution and rarely occur in benign clients' datasets. We show how attack effectiveness varies dramatically depending on the type of subpopulation targeted. We find that analyzing robustness issues through the lens of the data distribution is crucial, as some attacks exploit maliciously chosen input data (data poisoning) rather than deviations from the protocol.

2.1. Attack Strategies

We focus on targeted attacks that aims to violate the integrity of the global model [5, 7, 79, 82] by causing it to misclassify a subset of unmodified data samples while maintaining high accuracy on the main task. Specifically, a training set samples $\hat{D}$ that represent the target inputs are classified as a target class $\hat{t}$ rather than their ground-truth class.

Adversary Capabilities. The adversary can compromise a subset of client devices and adapt their training process to influence the global model $\mathbf{w}_G^{t+1}$ by submitting malicious updates to the server. We consider two adversary models, a model-poisoning adversary and a data-poisoning adversary (Fig. 1). The model-poisoning adversary can submit arbitrarily generated updates, while the data-poisoning adversary is limited to changing compromised clients' training data. We use the same realistic model poisoning threat model identified in [73] (Sec III-C2) where the adversary can only compromise a small fraction of clients ($< 5\%$ in a round). This is a reasonable assumption because a typical FL setup consists of hundreds of thousands of clients, and only a small fraction of them is selected in each training round. When a norm-bound defense is in place, the server can enforce an upper limit B on a p-norm of each client update such that $||\Delta\hat{\mathbf{w}}||_p \leq B$ [79, 82]. We assume that the attacker and clients are aware of this bound and that honest clients clip their updates to stay within the bound.

Data poisoning (DP). The adversary has access to the local dataset of compromised clients but cannot alter the training algorithm. We consider the widely used label flipping strategy [82] where the compromised clients' dataset contains a combination of benign samples D_i and target samples $\hat{D} = \{(x, \hat{t})\}$ with the labels flipped to the target class $\hat{t}$.

Model Poisoning (MP). In model poisoning, the adversary exploits its complete control over the training process to craft an arbitrary malicious update $\Delta\hat{\mathbf{w}}$. Specifically, in a model-replacement attack [5, 79, 82], the adversary injects

the desired attack target into their local model by fine-tuning their local model on both the benign and backdoor samples $D_i \cup \hat{D}$ using SGD, and then applies scaling to amplify the update to survive aggregation. This exploits the inherent vulnerability of linear aggregation functions to outliers, where the update of even a single client can influence the output of F_{avg} arbitrarily [9]. If a norm bound is present, the adversary adapts their strategy to the norm constraint. We consider three state-of-the-art adaptive attacks for the norm bound:

I) Projected Gradient Descent [79, 82](MP-PD). The adversary performs fine-tuning of the local update with SGD, but projects the update $\Delta\hat{\mathbf{w}}_i$ onto a constraint set after every iteration. This constraint set is defined such that the update satisfies the norm bound after scaling, i.e., $||\Delta\hat{\mathbf{w}}_i||_p \leq \frac{B}{\gamma}$, after being scaled by the scaling factor $\gamma = \frac{B}{||\Delta\hat{\mathbf{w}}_i||_p}$.

II) Neurotoxin [90] (MP-NT). This adaptive attack improves the durability of backdoors by poisoning specific weights that benign clients are unlikely to update. The adversary first determines $M = top_k(\mathbf{w}_G^t, \mathbf{w}_G^{t-1})$, the top-$k\%$ weights of $\mathbf{w}_G$ that are infrequently updated by comparing the current global model with the model from the previous round. The adversary then uses PGD to project their update on the coordinate-wise constraint $\Delta\hat{\mathbf{w}}_i \cup M = 0$.

III) Anticipate [83] (MP-AT). This adaptive attack is based on the idea that the optimal malicious update $\Delta\hat{\mathbf{w}}_i$ should take into account the effect of the aggregation and further training rounds. The attack tries to simulate the effect of the honest clients on the malicious update, including how future rounds will affect it. In order to compute the malicious update, the attack simulates multiple FL rounds locally in each optimization step and back-propagates through these simulated future updates.

2.2. Experimental Setup

We provide a brief overview of the experimental setup and refer the reader to Appendix §C for more details about the setup.

Analysis Tasks. Our first task (**FMN**) is a digit classification task on the Federated-MNIST dataset using the LeNet5 [52] architecture following [79, 82]. The second task (**C10**) is an image classification on the CIFAR-10 dataset using the ResNet-20 [46] CNN. We divide the dataset among 3383/100 (**FMN/C10**) clients in a non-IID fashion and select 30/40 clients in each round.

Attack Tasks. The attacks targeting prototypical inputs are: (**FMN-P**), which classifies images containing '7' as '1' instead [79], and (**C10-P**), which classifies images of green cars as birds [5]. The tail backdoor attacks target the two following inputs: (**FMN-T**), which classifies European-style '7' as '1' [82], and (**C10-T**), which misclassifies images of Southwest airline planes (which are not present in the original dataset) as trucks [82].

Client Selection. For both tasks, a constant number of clients m out of n total clients is selected in every round. We assume that the attacker controls a fraction $\alpha \in [0, 1]$ of the selected clients either for a single or multiple rounds.
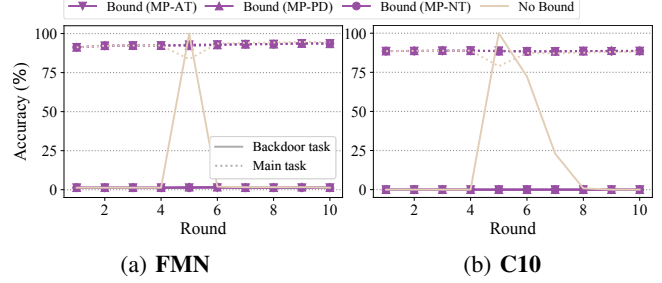

(a) **FMN**   (b) **C10**

Figure 3: Single-shot model-replacement attack [5] at round 5. The attacker can inject the backdoor in a single round by dominating the aggregation with scaling. (L_2-norm bound with three adaptive attacks, **FMN**: 4.0, **C10**: 5.0)

Metrics. We consider two metrics: main task accuracy and backdoor task accuracy. Main task accuracy denotes performance of the model on the benign objective, that is, the model's accuracy on the benign test set. The backdoor task accuracy reports the performance of the model on the malicious objective, which we define as the model's accuracy on a subset of the targeted backdoor images.

2.3. Attacks on Prototypical Targets

We first examine attacks on prototypical subpopulations. **Single-shot Attack.** Even an adversary controlling a single client (**FMN**: $\alpha = 3.3\%$, **C10**: $\alpha = 2.5\%$) that is selected in a single round can inject a backdoor by scaling its update by a factor of 30 (**FMN**) and 100 (**C10**) respectively, as Figure 3 shows. The backdoor is injected in the fifth round, after the model has achieved high accuracy on the main task, so that updates from benign clients only marginally change the model. The ability of an attacker to introduce backdoors targeting prototypical subpopulations is conditioned on the attacker's ability to scale its malicious contribution to overpower the benign clients' contributions. Specifically, a server that enforces an L_2-norm bound (**FMN**: 4.0, **C10**: 5.0 in Fig. 3) can prevent single-shot attacks for all adaptive attack strategies that we study, demonstrating the effectiveness of a constraint-based approach.

Continuous Attack. We now evaluate the effectiveness of the norm-bound defense against a continuous attacker, i.e., an attacker that controls a client participating in multiple rounds. An attacker controlling a single-client (**FMN**: $\alpha = 3.3\%$, **C10**: $\alpha = 2.5\%$) that is selected every round can maintain high accuracy on the malicious objective in the global model (**FMN**: $> 95\%$, **C10**: $> 80\%$) by continuously providing malicious updates (Fig. 4). We see that limiting the client contributions with an L_2−norm bound allows the server to reduce the attacker's success of injecting a prototypical backdoor (Figs. 4a and 4b). A norm bound of 1.0 (**FMN**) and 5.0 (**C10**) prevents the prototypical backdoor attack with hardly any classification accuracy loss on the main task. Although the attacker can send an update every round, the benign clients dominate the aggregation because the effect of scaling is limited by the bound. In Figs. 4a and 4b we only show the state-of-the-art MP-AT attack [83]. We additionally compare all three adaptive model poisoning

(a) **FMN-P**: Static (b) **C10-P**: Static (c) **FMN-P**: Adaptive (d) **C10-P**: Adaptive
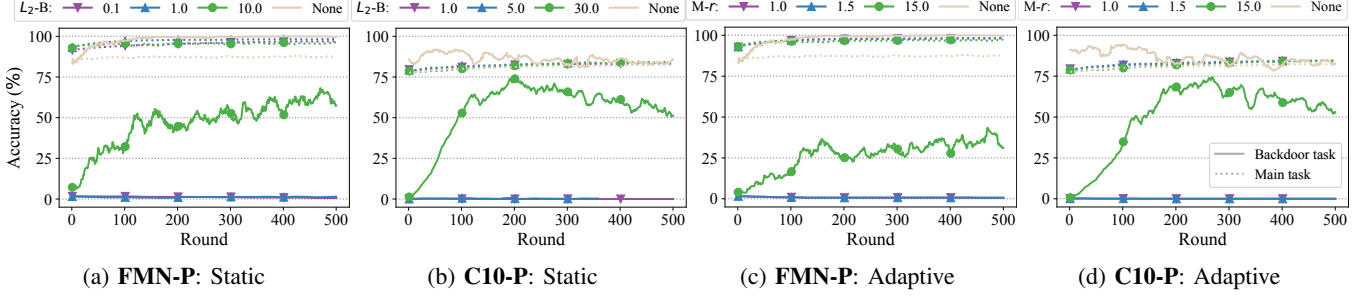
Figure 4: Anticipate attack under various static and adaptive norm bounds. The prototypical backdoor attack can be prevented by choosing an appropriate static or adaptive median-based norm bound.
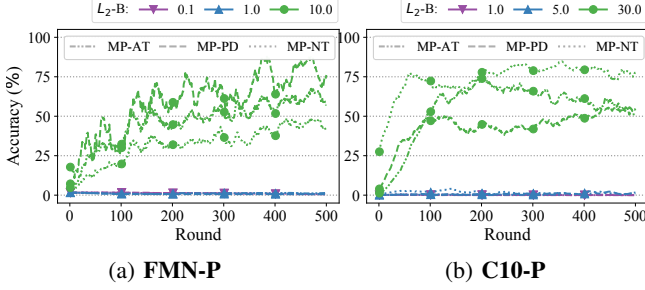

(a) **FMN-P** (b) **C10-P**

Figure 5: Comparison of adaptive attacks for various bounds.
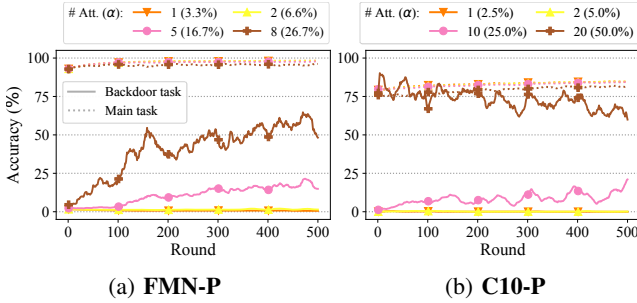

(a) **FMN-P** (b) **C10-P**

Figure 6: Continuous MP-AT attack for % of compromised clients per round. (M-r: 1.5)

strategies in Fig. 5. For an appropriate norm bound, attacks perform similarly with less than ten percent variation in backdoor accuracy and all three attacks fail to inject the backdoor. However, performance diverges when the bound is too loose (10 for **FMN** and 30 for **C10**). Since they perform similarly under appropriate norm bounds, we omit the other attacks in some figures to improve readability. We refer to Appendix §A for the full results.

Norm Bound Selection. The norm bound value must be chosen carefully if it is to be effective against prototypical backdoor attacks without interfering with the performance of the global model. If the bound is too loose, the attacker can exploit larger scaling factors to successfully inject a backdoor. Conversely, if the server selects the bound too tightly (i.e., 0.1 for **FMN** and 1.0 for **C10**), the convergence rate of the global model decreases unnecessarily because the honest clients have to clip their updates by a large margin (Figs. 4c and 4d). Intuitively, an effective norm bound should be close to the size of benign clients' updates.

However, what constitutes a suitable norm bound is not uniform across FL deployments and depends on factors such as the model architecture, training hyperparameters, and the data distribution of the clients. In addition, the size of benign clients' updates typically becomes smaller as the model converges. A bound set at the beginning would potentially give the adversary more influence in later stages of training. Hence, instead of picking a bound statically, we should select the norm bound dynamically, relative to the size of the benign clients' current updates.

However, estimating the average norm of benign updates is challenging due to the presence of malicious clients. Simple approaches, such as the mean or average of the client update sizes, can easily be manipulated by individual malicious outliers. Instead, we need to use a statistic that is robust to outliers, such as the median, which can tolerate up to 50% malicious inputs. Note that in the secure FL setting where the server does not have direct access to client updates, the robustness of the median also removes the need to cryptographically verify clients' reported[3] update sizes.

The median gives us a bound that is relative to the typical client update and that adjusts over time as the model converges. However, directly using the median m as the bound would discard half the updates, drastically slowing the convergence rate. Instead, the bound is set to rm for a small multiplier r. This multiplier is a hyperparameter that must be tuned in combination with the other training hyperparameters, but is less dependent on components such as model architecture than a static norm because of its relativity to the median client update norm. Figs. 4c and 4d show that a median-based bound with $r = 1.5$ successfully prevents attacks without impacting the convergence times of the main task. This translates to absolute L_2-norm bounds in the range of $[0.24, 0.86]$ for **FMN** and $[1.96, 2.81]$ for **C10**. For the rest of this analysis, we adopt the median-based selection strategy and use $r = 1.5$ across tasks.

Growing Number of Compromised Clients. So far, we have considered an adversary that controls a single client in each round. In Fig. 6, we see that when the attacker controls more clients per round, the effectiveness of injecting prototypical backdoor increases even if a norm bound is

---

3. Revealing the client update norm should be acceptable in practice because it affects privacy only marginally. However, privacy-preserving cryptographic protocols to approximate the median are available [12, 27].
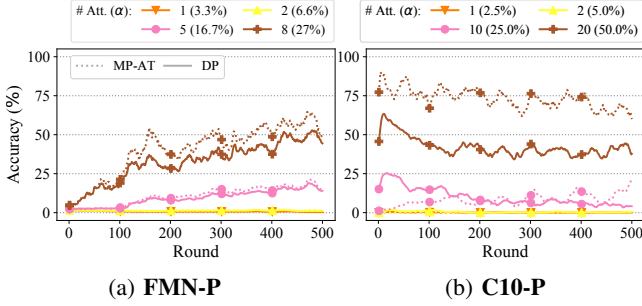
(a) **FMN-P**   (b) **C10-P**

Figure 7: Comparison of model poisoning and data poisoning attack strategies for % of compromised clients per round. The backdoor accuracy difference between model and data poisoning is small. (M-r: 1.5)
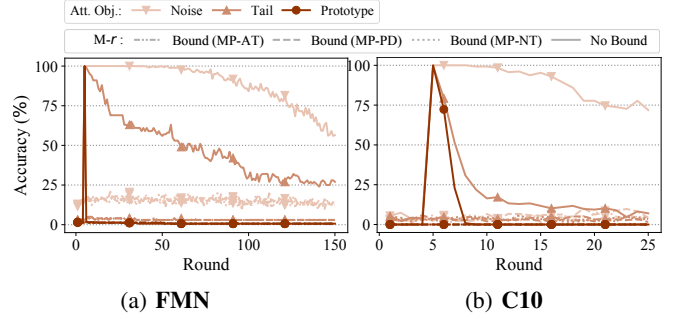

(a) **FMN**   (b) **C10**

Figure 8: Comparison of prototypical and tail attack targets for the single-shot attack. Prototypical targets are quickly reversed by the honest clients, but tail targets stay in the model longer. (M-r: 1.5)

enforced. In these experiments, the strategy of the attacker is to divide the scaled update among the compromised clients so that the individual scaling factor for each malicious client becomes smaller. For **FMN**, we see that the malicious objectives' success is close to zero for $\alpha <= 6.6\%$ but above 50% for $\alpha = 26.7\%$. Similarly, for **C10**, the accuracy of the malicious objective is low for a small number of attackers ($\alpha <= 5.0\%$), but increases to more than 50% when $\alpha = 50\%$. We observe that scaling is less relevant for attack success if the attacker controls enough clients in each round. This observation is supported in Fig. 7, which shows a comparison of model poisoning (MP-AT) with data poisoning. Data poisoning shows a similar effectiveness improvement as the number of compromised clients per round increases, staying within at least 60% of the backdoor accuracy of model poisoning for both tasks. This suggests that we can use norm bounding to eliminate or reduce the advantage an attacker gains from model poisoning to the point at which the effectiveness of their attack is little better than that of honestly following the protocol and merely training over poisoned data. Of course, if an adversary can continuously influence a significant fraction of the selected clients, the definition of FL implies that their malicious information will be incorporated into the global model. However, in practical FL deployments, it is unlikely that an attacker will be able to continuously control a large fraction of the clients each round, as they are selected randomly from a much larger pool of available clients.

2.4. Attacks on Tail Targets

So far, we have discussed how attackers can embed backdoors in models by using scaling, which exploits the susceptibility inherent in the underlying linear aggregation rules used in FL. All the early targeted attacks that were proposed on FL [5, 7] relied on scaling for attack success. More recently, however, work has emerged that shows that introducing backdoors can be achieved without scaling if the attacker selects a particular subset of inputs as its attack target [82]. In the following, we provide insights into why this is possible and what aspect of the learning process these attacks exploit.

Single-shot Attack. Before we explore why these attacks are possible, we discuss some empirical observations to

compare the performance of prototypical and tail attack targets for model poisoning and data poisoning attacks. In Fig. 8, we can see a comparison between tail backdoors (**FMN-T**, **C10-T**) and prototypical backdoors (**FMN-P**, **C10-P**) for a single-shot attacker. We observe that a tail backdoor remains in the global model for a very long time (at least 200 rounds for **FMN**) after its initial introduction, whereas the prototypical backdoor is almost completely eliminated only a few rounds after its introduction. This is primarily because honest clients are unlikely to submit updates that strongly influence model behavior for the sub-population of the tail backdoor since it is less likely that the benign clients' training data contains data points from the tail of the distribution. To further highlight this phenomenon, we experiment with data samples of random (Gaussian) noise as the backdoor target. We use these to emulate the behavior of the model on extreme tail samples. The model memorizes the noise perfectly for many rounds before being overwritten by benign clients' updates, suggesting that the farther backdoor targets are away from the data distributions of benign clients, the longer the backdoor stays in the model. Nevertheless, similar to prototypical targets, a single-shot tail attack still fails to inject the behavior successfully when a norm bound is enforced (Fig. 8).

Continuous Attack. In contrast to all other settings, a continuous attacker can slowly inject a tail backdoor even in the presence of a norm bound. However, injection is not instantaneous (c.f. single-shot tail attacks, Fig. 8) and requires the attacker to participate continuously over many rounds. Backdoor accuracy reaches 50% only after 75 rounds for **FMN** and 500 for **C10** (Fig. 9). In contrast, the prototypical backdoor attack's success remains below 10% throughout the whole training process for both tasks. This difference is because, for tail targets, the attacker requires less influence over the aggregation process to succeed than for prototypical targets, as the malicious behavior is slowly learned by the global model. This is also apparent for the noise backdoor target, where the model learns to almost perfectly memorize samples of random noise in less than 100 rounds. Data poisoning is similarly effective as model poisoning at injecting tail targets in both tasks (within 50% of the accuracy of MP) and noise targets (similar accuracy as MP). This indicates
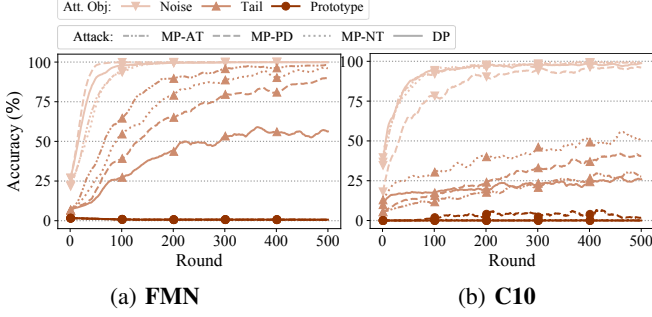
(a) **FMN**   (b) **C10**

Figure 9: Comparison of prototypical and tail attack targets for the continuous attack under a norm bound. (M-r: 1.5)

that scaling is less important for continuous tail attacks, allowing them to succeed even when the norm bound is tight. These results indicate that these tail attacks exploit the characteristics of the learning algorithm and not the aggregation rule itself. Based on our observations, we would expect previously unsuccessful poisoning attacks on prototypical targets to succeed once we artificially modify the samples with the intention making them less prototypical and more like the tail targets we studied above. To investigate this, we add an artificial trigger [45] to a set of prototypical attack targets. In Fig. 10, we show that it is indeed the case that the attack on the modified prototypical targets is successful, with an improvement in backdoor accuracy of more than 85% for **FMN** and 25% for **C10**. This is possible because artificial triggers can move samples into a new, rare, subpopulation, enabling the same attacks that work on naturally rare subpopulations.

Takeaway: Continuous attacks on tail targets *cannot* be prevented using a norm bound, because these attacks exploit the models learning capacity to embed their backdoors. Our results align with other empirical results [89] and a recent theoretical explanation [36, 37] that memorizing data at the tail of the distribution is an inherent and essential property of modern over-parameterized learning models. ML memorization's relation to data privacy has been explored in depth. Research shows that this is what enables membership inference attacks [78] or attacks that extract sensitive data from the model [22, 23]. However, with the emergence of learning paradigms such as FL that are vulnerable to active attacks, memorization can also pose serious risks to robustness. Because it is memorization that allows attackers to embed tail backdoors, attackers only need to participate in the learning process with malformed data and otherwise follow the protocol honestly. Even a single attacking client, if participating over multiple rounds, is sufficient for the backdoor to be embedded persistently. Our analysis shows that norm bounds effectively reduce the attack surface of FL while highlighting the limits of the approach.

Norm bounds provide useful robustness guarantees because of the ease of performing practical high-impact single-shot attacks in standard secure FL. While it is likely that an attacker will be selected at least once, thus enabling a single-shot attack, being selected consistently over many rounds to enable a continuous attack is unlikely in a practical
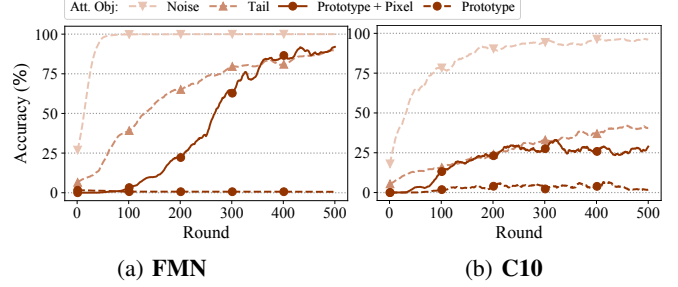

(a) **FMN**   (b) **C10**

Figure 10: Adding a pixel pattern to prototypical targets makes them behave similar to tail targets. (M-r: 1.5)

FL setup [73]. Achieving this persistence would require an attacker to compromise a significant portion ($> 1\%$) of clients, which corresponds to hundreds of thousands of devices in large-scale FL deployments. We thus conclude that norm bounds provide important practical robustness guarantees for FL. Although continuous attacks on tail targets remain effective even under norm bounds, they require a significantly stronger attacker, which is less of a concern in most practical deployments.

2.5. Discussion

As ML is deployed in a wider range of settings, it has become clear that it must move beyond simply pursuing accuracy and start to tackle important objectives that matters in real-world deployments such as privacy and robustness. FL, though presenting many privacy benefits, amplifies ML robustness issues by exposing the learning to an active attacker that can be present throughout the training process and fine-tune their attacks to the current state of the model. In consequence, we have seen new and powerful attacks emerge that effectively impact FL integrity. In this analysis, we show that norm bounding can provide meaningful practical robustness guarantees for FL. Furthermore, our analysis shows that many attacks that remain possible in the presence of norm bounds are likely inherent to the learning process. For example, norm bounds are not able to prevent an attack targeting the tail of the input distribution without sabotaging the model's ability to learn less representative data points. In general, defenses against all possible attacks are likely an unattainable goal, and we should not refrain from strengthening our systems with existing solutions while we continue to expand our understanding of the underlying robustness issues of decentralized learning. This mirrors trends in the ML community, moving from a binary view of robustness to a more differentiated way of addressing adversarial behavior.

The requirement for ML algorithms to memorize in order to learn tail subpopulations' specific behavior [36] has largely been studied in the context of privacy. However, as we have shown in this analysis, it also has significant implications for ML integrity. Consequently, mechanisms that have been employed to limit memorization, such as appropriate regularization and noise addition (i.e., differential privacy) are also likely to improve ML robustness for tail targets [36]. However, these countermeasures introduce
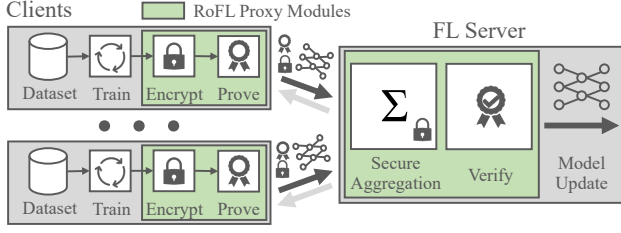
Figure 11: RoFL augments secure FL with proof and verification steps to verify constraints on private model updates.

various tradeoffs with respect to the accuracy, privacy, robustness, and fairness [4] and are beyond the scope of this paper.

The appeal of norm bounds is that they exist at an intersection of effectiveness and efficiency where they prevent an extensive range of simple-to-execute yet devastating attacks in practice while remaining efficiently implementable. In addition, they do so in a robust way that does not require obfuscation (i.e., the protection still holds if the attacker knows the bound). Norm bounds increase robustness immensely in practical settings, and more extensive protections currently require sacrificing privacy, efficiency, or both [9, 25, 40, 61, 67, 76, 86]. However, the privacy-preserving nature of secure-aggregation-based FL means that enforcing norm bounds is non-trivial. Relying on clients to self-regulate is not viable in the presence of actively malicious attackers, and using generic secure computation tools would be prohibitively expensive. Therefore, we need solutions that can enforce norm bounds over (private) updates from untrusted clients with an overhead small enough to enable practical deployment. The remaining of this paper aims to address this challenge.

3. RoFL Design

In this section, we present RoFL, a new secure aggregation protocol that allows the server to verify that clients' secret inputs satisfy a predefined constraint (i.e., input validation). We start the section with a brief overview of the cryptographic blocks used in RoFL, describe our setup and threat model, and then present the details of our protocol.

3.1. Preliminaries

Commitments. A hiding non-interactive commitment scheme is an algorithm that allows an entity to commit to a value v while keeping it hidden, with the option of later revealing the value. Given a function $Com : M \times R \to C$, from a message space M and a randomness space R to a commitment space C, we compute a commitment to a value $v \in M$ as $c = Com(v, r)$ for $r \leftarrow R$. Opening a commitment by revealing r and v allows the commitment to be verified by checking $c_v = Com(v, r)$. A commitment is *hiding* if it does not reveal any information about the committed value v. Furthermore, a commitment is *binding* if it is (computationally) infeasible to find v, r and v', r' such that $v \neq v'$ but $Com(v, r) = Com(v', r')$.

Zero-Knowledge Arguments of Knowledge. A ZKP is a statement that proves a fact to another entity without

revealing additional information. For example, the prover might want to show that they know the discrete log x of a group element g^x and that x lies in a specific range without revealing x. Traditionally, the prover would engage in an interactive challenge-response protocol with the verifier to prove a property about its private data. Using the Fiat–Shamir transform [38], these interactive ZKP protocols can be converted into a non-interactive zero-knowledge proof (NIZK) [10]. The prover generates a single binary string π that can be used by any entity to verify the claim without further interaction. For brevity, we refer to [18] for formal definitions of the security properties of NIZKs.

3.2. Setup and Threat Model

RoFL is designed for a standard FL deployment featuring a single server coordinating a model learning process with a large number of clients (Fig. 11). In each round of the training process, the server randomly selects m available clients and communicates the current global model to them. Each selected client i for $i \in \{1, 2, \ldots, m\}$ then performs local model training and computes an ℓ-dimensional update vector $\mathbf{w}_i \in \mathbb{Z}_q^\ell$ that they encode using a homomorphic masking scheme before sending it to the server. The server uses secure aggregation to combine the clients' update vectors $\sum_{i=1}^m \mathbf{w}_i$ into a single update vector for the global model. RoFL allows the server to verify that the client-updates $\mathbf{w}_i$ are valid, i.e., fulfill a predefined constraint while preserving the guarantees of secure aggregation and remaining compatible with the conventional deployment scenario. Specifically, it allows the L_p norm of client-updates $\mathbf{w}_i$ to be constrained, i.e., $\|\mathbf{w}_i\|_p < B$ for some $B \ll q$, and we provide concrete instantiations for both L_∞ and L_2 norm bounds. Upon receiving the client input, the server verifies that the proof is valid. If the verification fails, the server discards the client input, and flags the client as corrupt.

Threat Model. We distinguish between a malicious and a semi-honest server. Malicious parties can deviate from the protocol arbitrarily, whereas a semi-honest party does not deviate from the protocol but analyzes received messages to try and gain information about clients' private inputs. We inherit the confidentiality guarantees of secure aggregation [6, 13], which ensures input privacy for a malicious server that colludes with a fraction γm of malicious clients. Specifically, an adversary cannot learn anything beyond what they can derive from the aggregation output and the proven constraint of the inputs. Further RoFL additionally ensures correctness against malicious clients for a semi-honest server. We assume a PKI, which allows the server to facilitate end-to-end encrypted and authenticated communication between clients without the need for direct communication between the clients.

3.3. Secure Aggregation with Commitments

Secure Aggregation. We begin by describing the secure aggregation protocol first introduced by Bonawitz et al.[13] and extended by Bell et al. [6]. After this brief introduction, we show how we extend the protocol to support input

constraints. The secure aggregation protocol relies on a homomorphic masking scheme, where each client blinds their input vector $\mathbf{w}_i$ with a key $\mathbf{r}_i$, which are generated so that they cancel out when aggregated. The distribution of masking keys $\mathbf{r}_i$ between clients is achieved by the `ShareKeys` sub-protocol. In order to support dropouts from clients during the aggregation, `Unmask` is used to reconstruct mask contributions from clients that dropped out. RoFL extends the protocol by adapting the homomorphic masking scheme but does not need to modify `ShareKeys` and `Unmask`. We, therefore, refer to [6] for a description.

We abstract the method of how clients hide their vectors given a masking key as an encoding scheme in order to improve the presentation of our later extensions. This encoding scheme (`Enc`, `Dec`) should be homomorphic in both the messages $\mathbf{w}_i$ and the keys $\mathbf{r}_i$ created[4] by `ShareKeys`. Specifically, $\sum_i \mathrm{Enc}(\mathbf{w}_i, \mathbf{r}_i) = \mathrm{Enc}(\sum_i \mathbf{w}_i, \sum_i \mathbf{r}_i)$. The encoding scheme allows the server to aggregate messages encoded with clients' individual keys $\mathbf{r}_i$ and decrypt the aggregate $\mathrm{Dec}(\sum_i \mathrm{Enc}(\mathbf{w}_i, \mathbf{r}_i), \mathbf{r})$ iff it has access to the aggregate key $\mathbf{r} = \sum_i \mathbf{r}_i$ which can be computed securely using `Unmask`. The encoding and decoding functions of the original secure aggregation protocol are defined as:

$$\mathrm{Enc}(\mathbf{w}_i, \mathbf{r}_i) := \mathbf{w}_i + \mathbf{r}_i \mod q$$
$$\mathrm{Dec}(\mathbf{y}, \mathbf{r}) := \mathbf{y} - \mathbf{r} \mod q \tag{1}$$

for $\mathbf{w}_i, \mathbf{y}, \mathbf{r}_i, \mathbf{r} \in \mathbb{Z}_q^\ell$.

Correctness with malicious clients. In order to enforce robustness constraints in secure FL, the encoding scheme needs to support efficient client-side zero-knowledge constraints. More importantly, however, the encoding scheme must allow the server to verify *correctness* of the protocol in the face of actively adversarial clients. The original secure aggregation protocol only guarantees input privacy, but not correctness, which they note "is much harder to achieve" [13]. For instance, malicious clients can use inconsistent masking values $\mathbf{r}'_i$ in `ShareKeys` and `Unmask` that can result in arbitrary outputs at the server. We must first ensure the correctness of the secure aggregation protocol in this setting if we are to prove any form of robustness constraints meaningfully.

For the server to recover the correct result, encoding keys must add up to the aggregate key $\sum_i \mathbf{r}_i = \mathbf{r}$. In classical secure aggregation, this would require clients to prove that they executed $\mathrm{Enc}(\mathbf{w}_i, \mathbf{r}_i)$, including the generation of the $\mathbf{r}_i$, honestly as specified by the protocol. However, proving this is prohibitively expensive, as the formation of $\mathbf{r}_i$ entails key agreements and many evaluations of a PRG. One of our key insights is that it is sufficient for correctness to ensure that the sum of the keys used in the encodings $\sum_i \mathbf{r}_i$ is equal to the aggregate key output by `Unmask`. However, proofing this condition for the original encoding scheme still introduces considerable overhead. Instead, we propose an optimization that allows the server to verify this

condition efficiently by exploiting the homomorphic nature of the encoding scheme. We show this optimization for a generic construction before considering how to instantiate it efficiently.

We define a new generic encoding scheme from two homomorphic masking schemes E_w and E_r:

$$\mathrm{Enc}(\mathbf{w}_i, \mathbf{r}_i) = (\mathbf{c}_i^{(1)}, \mathbf{c}_i^{(2)}) = (\mathrm{E}_w(\mathbf{w}_i, \mathbf{r}_i), \mathrm{E}_r(0, \mathbf{r}_i))$$
$$\mathrm{Dec}((\mathbf{c}_i^{(1)}, \mathbf{c}_i^{(2)}), \mathbf{r}) = \mathrm{D}_w(\mathbf{c}_i^{(1)}, \mathbf{r}) \tag{2}$$

Here, E_w and E_r must be homomorphic in both message and key, just as the original encoding scheme is. We note that E_r must be secure even for a known message, which the original scheme does not fulfill. Depending on the instantiation, we might also require a proof that the randomness used in $\mathbf{c}_i^{(1)}$ and $\mathbf{c}_i^{(2)}$ is the same. The server can use the second element to verify if the output $\mathbf{r}$ from `Unmask` cancels out to zero by comparing $\mathrm{E}_r(0, \mathbf{r})$ to $\sum_{i=1}^m \mathbf{c}_i^{(2)}$. If they are equal, this implies $\mathbf{r} = \sum_{i=0}^m \mathbf{r}_i$. Otherwise, the server aborts that particular round and proceeds with another subset of clients. Hence, RoFL provides correctness for actively malicious clients by detecting malformed aggregations.

Commitment-based Encoding. We identify ElGamal commitments [32] as an ideal building block in our design because we can use them to instantiate our generic encoding scheme extremely efficiently. ElGamal commitments are both computationally hiding and information-theoretically binding under a standard discrete log hardness assumption. Furthermore, ElGamal commitments are additively homomorphic in both the messages and keys, thus supporting the required homomorphisms. In addition, efficient zero-knowledge range proofs [18] exist that are a natural fit to realize p-norm constraints and operate directly on (parts of) the ElGamal commitment.

We lift the masking approach from secure aggregation to the commitment-based setting by instantiating our generic encoding scheme with a *single* ElGamal commitments $Com_{EG}(\mathbf{w}_i, \mathbf{r}_i)$ (for each element in $\mathbf{w}_i$) by setting

$$(\mathrm{E}_w(\mathbf{w}_i, \mathbf{r}_i), \mathrm{E}_r(0, \mathbf{r}_i)) = Com_{EG}(\mathbf{w}_i, \mathbf{r}_i) = (g^{\mathbf{w}_i} h^{\mathbf{r}_i}, g^{\mathbf{r}_i})$$
$$\mathrm{D}_w(\mathbf{y}, \mathbf{r}) = dlog_g(\mathbf{y} \cdot h^{-\mathbf{r}})$$

where $\mathbf{w}_i, \mathbf{w}, \mathbf{r}_i, \mathbf{r} \in \mathbb{Z}_q^\ell$ and g, h are generators in a group $\mathbb{G}$ of prime order q. Note that $g^{\mathbf{w}_i} h^{\mathbf{r}_i}$ is also a valid Pedersen commitment [63] to $\mathbf{w}_i$ in its own, and we will occasionally make use of this fact in the rest of this section. The decoding function D_w computes the discrete logarithm with respect to g, which results in $\mathbf{w}$ if $\mathbf{y}$ is a correct encoding of $g^{\mathbf{w}} h^{\mathbf{r}}$. We assume that the discrete log problem is hard in $\mathbb{G}$. While this means that calculating the discrete logarithm of $y = g^x$ is hard for generic $x \in \mathbb{Z}_q$, the aggregation result space in our domain is small (e.g., a 32 bit integer) and hence the discrete logarithm can be computed efficiently [65, 72, 77]. Clients generate a standard non-interactive proof-of-knowledge of discrete logarithm [20] to ensure the commitments are well formed, i.e., clients use the same $\mathbf{r}_i$ in $g^{\mathbf{w}_i} h^{\mathbf{r}_i}$ and $g^{\mathbf{r}_i}$.

Alternative approaches. We compare the performance of our ElGamal-based encoding with an alternative candi-

---

4. Technically, in [13], `ShareKeys` is used to derive seeds which are then expanded into the keys via a PRG.

	RLWE + Groth16	EG + Groth16	RLWE + BP	EG + BP (§3)
Communication	1	1	$\log(\ell)$	$\log(\ell)$
Vector size Proof size	664KB 0.192KB	524KB 0.192KB	664KB 2.01KB	524KB 1050KB
Prover	$\ell\log(\ell)$ 208.37s	$\ell\log(\ell)$ 610.15s	ℓ 16896.05s	ℓ 5.50s
Verifier	1 0.002s	1 0.01s	ℓ 1140.97s	ℓ 0.57s
Trusted Setup	211.71s	618.56s	—	—

TABLE 1: We implement and evaluate several combinations of our encoding scheme and zero-knowledge proof system to present their concrete communication and computation overhead. The concrete rows are based on proving an 8-bit L_∞-norm bound for a vector of 2^{13} elements on a c5d.4xlarge AWS instance. For the asymptotic rows, Big-O notation is omitted for brevity.

	Plaintext	Bonawitz et al. [13]	Bell et al. [6]	RoFL (§3)
Client communication	ℓ	$\ell+m$	$\ell+\log(m)$	$\ell+\log(m)$
Client computation	ℓ	ℓm	$\ell\log(m)$	$\ell\log(m)$
Input privacy	○	●	●	●
Aggregation correctness	○	○	○	●
Input validation	●	○	○	●

TABLE 2: Overview of computation and communication asymptotics and security properties for different (secure) aggregation algorithms. Big-O notation is omitted for brevity.

date using RLWE-based encryption [15], instantiating our generic encoding as

$$\mathrm{E_w}(\mathbf{w}_i, \mathbf{r}_i) := (\mathrm{ct}_0, \mathrm{ct}_1) = (\mathbf{a} * \mathbf{r}_i + \mathbf{w}_i + t * \mathbf{e}, -\mathbf{a})$$
$$\mathrm{E_r}(\mathbf{r}_i) := (\mathrm{ct}_0', \mathrm{ct}_1') = (\mathbf{a}' * \mathbf{r}_i + 0 + t * \mathbf{e}', -\mathbf{a}')$$
$$\mathrm{D_w}(\mathrm{ct}_0, \mathrm{ct}_1) := (\mathrm{ct}_0 + \mathrm{ct}_1 * \mathrm{sk}) \quad \mathrm{mod}\ t$$

where the masks $\mathbf{a}, \mathbf{a}'$ are random ring elements, $\mathbf{e}$, $\mathbf{e}'$ are small noise terms added for security, and t is a public parameter. We refer to [15] for further details and a proof of security. While this encryption scheme is actually *fully homomorphic in the message*, it is also *additively* homomorphic in the keys. We implement this using the state-of-the-art Microsoft SEAL library. Table 1 shows that our ElGamal construction outperforms the RLWE-based construction independently of the proof system used to enforce the input constraints.

3.4. ZKP of Norm Bounds

In order to enforce norm bounds, RoFL requires each client to provide a NIZK proof that their update vector $\mathbf{c}_i = \mathrm{Enc}(\mathbf{w}_i, \mathbf{r}_i)$ is well formed and has a norm bounded by B, where B is set by the server. RoFL offers two methods to select the norm bound adaptively, either the server uses public training data to compute an update and estimate a bound, or the median method presented in §2.3 is followed. Let $\mathbf{w}_i$ be the parameter vector and $\mathbf{r}_i$ the vector of canceling nonces of client i. In Camenisch and Stadler notation [20], a client needs to provide a proof of the following relation to the server:

$$NIZK(\mathbf{w}_i, \mathbf{r}_i)\{\mathbf{c}_i = (g^{\mathbf{w}_i} h^{\mathbf{r}_i}, g^{\mathbf{r}_i}) \wedge \|\mathbf{w}_i\|_p < B\}$$

A wide range of ZKP systems exists to implement such proofs with various performance and security trade-offs. For FL workloads with a large number of parameters, we require a proof system that is as lightweight as possible on the client, has moderate bandwidth requirements, and has efficient realizations of range proofs compatible with the homomorphic commitments required for norm constraints. RoFL uses Bulletproofs [18] because they provide linear time complexity for prover and verifier, they only have a logarithmic proof size, they do not rely on a trusted setup, they have a specialized efficient realization of range proofs with batching, and, lastly, they operate directly on the first component of the ElGamal commitment. We compare the performance of RoFL with Bulletproofs with Groth16 [44], a highly efficient state-of-the-art ZKP systems that offers (extremely small) constant proof sizes and verification times. Note that this efficiency comes at the cost of requiring a trusted setup. We use Circom [41] to define a ZKP circuit for each encoding in its custom specification language. Circom then translates this into a system of rank-1 constraints (R1CS) which both Groth16 and Bulletproofs support. Groth16 tends to outperform Bulletproofs in terms of concrete prover time for generic R1CS proofs. However, since RoFL uses rangeproofs over Pedersen commitments, which can be instantiated much more efficiently in Bulletproofs than generic R1CS proofs, they significantly outperform Groth16 for this task. However, bridging the gap between the Pedersen commitment that Bulletproofs natively operate on and the ElGamal commitments we use requires a proof of well-formedness that introduces virtually no computational overhead (less than 0.15 s of the prover runtime are due to the well-formedness proof) but significant communication overhead, accounting for all but 1KB of the proof size of our system. However, the reduction in prover time in combination with very competitive verifier times more than makes up for this in practice as we show in §4.

We now show how to realize L_∞ and L_2 constraints; the two variants of norm constraints that RoFL supports. We start with the L_∞ norm bound where we can consider each parameter independently. We then show how to bound the L_2 norm, which depends on all parameters in the vector and which requires clients to additionally commit to, and prove range bounds over, the squares of each parameter. Although this introduces significant additional costs in communication and computation, L_2 bounds are frequently used in the ML robustness literature because the lack of strict bounds on each parameter allows for more uneven parameter weight distributions in the updates.

L_∞-Norm. To bound the L_∞ norm by B, it is sufficient for the prover to show to the verifier that each parameter $w_j \in \mathbf{w}_i$ is within a bounded range $w_j \in [0, B)$. For each committed parameter $c_j \in \mathbf{c}_i$, the client provides a ZKP that $w_j \in [0, B)$ and that c_j was generated correctly.

RoFL combines a proof-of-knowledge of the discrete logarithm [20] for the correctness of the commitment and a specialized efficient range-proof from Bulletproofs [18] for the range check. Bulletproofs can aggregate all ℓ required range proofs, one for each parameter update, into a single proof consisting of only $2(\log_2(b) + \log_2(\ell)) + 4$ group elements where b is the bit length of the range. In FL, this reduces the bandwidth cost from linear in the number of parameters to logarithmic (e.g., an improvement of 26000x for 100k parameters). Further, instead of mapping 32-bit floating-point parameters directly to $\mathbb{Z}_q$ with standard fixed-point encoding, RoFL compresses parameters into b-bit integers (e.g., $b = 8$ or $b = 16$) with probabilistic quantization [49]. Our evaluation in §4 shows that this quantization does not lead to a significant loss of accuracy in the overall model.

L_2-Norm. Bounding each parameter by B using the L_∞ bound implies a trivial bound of the L_2 norm to $\ell \cdot B^2$. However, since an L_2 norm bound is supposed to allow more flexibility for the individual parameters, this is not a useful way to instantiate L_2 bounds. On the other hand, it is not sufficient to merely proof $\sum_{i=1}^{\ell} w_i^2 < B_{L_2}^2$. Because we are working in $\mathbb{Z}_q$ and a malicious client could cause the sum to wrap around to small values by setting a w_j sufficiently large to cause an overflow. Therefore, we need to prove that each element is sufficiently small to prevent this attack. We note that each honest parameter must satisfy $w_j^2 < B_{L_2}^2$ because otherwise the sum would trivially violate the L_2 bound. Therefore, we can enforce an L_∞ bound of B_{L_2} without loss of generality. We therefore construct our L_2 norm bounds as an extension of our L_∞ norm bounds, appending an additional proof to verify that the sum of squared parameters is bounded. More specifically, for each commitment c_j to parameter w_j, the client provides an additional Pedersen commitment to the square of the parameter, $c_j' := Com_{PD}(w_j^2, r_j') = g^{w_j^2} h^{r_j'}$ and provide a proof that the sum of all c_j''s in the vector lies in the range $[0, B_{L_2}^2]$. In addition, the clients also generate a proof of consistency, to ensure that c_j' indeed commits to the square of the value committed to in c_j. This can be done with an additional non-interactive proof-of-knowledge of the discrete logarithm [20] by proving the knowledge of an opening to a Pedersen commitment by rewriting it to $c_j' = c_j^{w_j} h^{r_j' - w_j r_j}$. The verifier checks this by computing the same sum over all c_j' using the homomorphic property and checking the range-proof. Again, we use Bulletproofs to instantiate the range proof.

General Constraints. In addition to the L_∞ and L_2 norm bounds, RoFL can support arbitrary client-side constraints expressed as circuits by using standard general-purpose zero-knowledge proofs supported by Bulletproofs. However, the optimizations proposed in this paper to make the ZKPs feasible for FL workloads might not be directly compatible with arbitrary constraints, where other tailored optimizations might need to be considered.

3.5. Optimizations

So far, we have described the cryptographic building blocks of RoFL and how our system can achieve its performance through a careful co-design of cryptography and FL protocol. Despite these improvements, scaling the protocol to realistic neural network models remains challenging. Even as we optimize the cryptographic protocol to the specifics of FL, the sheer number of proofs required due to the number of model parameters will result in significant overhead in both computation and bandwidth. Thus, we explore how we can reduce the number of proofs required by considering optimizations that take advantage of the combination of our protocol with the underlying FL model. First, we consider probabilistic range checks to optimize the L_∞ variant of our protocol. Second, we present the application of a compression technique based on random subspace learning, compatible with our L_2 constraint. Finally, we discuss how we can use optimistic continuation to interleave the server-side verification of the previous round with client-side training for the next round. This significantly reduces the effective wall-time overhead at the cost of requiring a one-round rollback in case verification fails due to malicious activity.

Probabilistic Range-Checking (L_∞). For the L_∞ bound, the range proofs are by far the most significant component of the computation overhead (§4). In the base protocol, the client proves that each element in $\mathbf{w}_i$ is smaller than the bound B. However, it is not necessary to check all ℓ elements to detect a malicious client with high probability. RoFL can employ probabilistic checking of a random subset of the update to reduce the number of range proofs required. The key insight is that the server can choose the random subset that it wants to verify after the clients have uploaded their commitments. Because the commitments are binding, the clients are unable to change the already committed parameters. A single failed verification in the update is sufficient to detect a malicious client, which means that even comparatively small subsets provide strong guarantees. The probability that the server selects at least one element in which the malicious update exceeds the bound follows a hypergeometric distribution. We show that, due to the properties of the hypergeometric distribution, we can make the probability of a violation being missed vanishingly small. For example, assume that a fraction $p_v \in (0, 1]$ of the ℓ parameters in an update exceeds the bound and the server checks a fraction $p_c \in (0, 1]$ of the parameters. The probability that the server does not detect a malicious update with $p_v \cdot \ell$ elements above the bound, while checking $p_c \cdot \ell$ elements, can be modeled as $Hyp(p_c \cdot \ell | \ell, \ell \cdot (1 - p_v), p_c \cdot \ell)$. The security guarantees of this optimization depend on the ratio of checked parameters to above-bound malicious parameters. Although one cannot exclude the possibility of a successful attack modifying a very small number of parameters, down to only 1, such a hypothetical attack would have to be incredibly sophisticated, because the attacker is significantly more constrained than with existing attacks. As we move to apply this optimization to larger models, the fraction of checks required remains small even as the total number of parameters increases. We provide an empirical analysis of the security of probabilistic checking under generic and adaptive attacks in Appendix §B.

Unfortunately, this optimization cannot be applied in the L_2 norm variant, because even a single out-of-bound parameter in the update vector could lead to an overflow of the sum-of-squares proof. Essentially, this would allow an adversary to submit an arbitrary update vector while remaining undetected with high probability.

Compression Techniques (L_2). Because both computation and communication overheads are linear in the size of the update vectors (§4), we can benefit from ML [49] compression techniques that reduce the size of the updates. However, the need to be compatible with secure aggregation and norm bounding limits the number of applicable techniques. In RoFL, we consider random subspace learning [53] as an optimization, which was initially introduced to help understand the hardness of ML tasks. Random subspace learning applies an L_2-norm-preserving transformation that reduces the number of parameters required for a model update. The transformation preserves the L_2-norm, and thus the technique is compatible with our L_2 constraint. In random subspace learning, the model is trained in a random subspace W^d that has a lower dimension d than the original model's dimension l. The random subspace W^d is projected onto the original model space W^l using an orthonormal projection matrix $P \in \mathbb{R}^{l \times d}$: $W^l = W_0^l + PW^d$. During training, W_0^l and P are treated as constants, and optimizations are done on W^d. Therefore, only W^d has to be exchanged between the client and server in each round, resulting in an improvement in the number of parameters from l to d. The parameter d, referred to as the *intrinsic dimension* [53], defines the compression ratio and has to be set by the server depending on the training task.

Optimistic Continuation. The computation time at the server is dominated by proof verification (§4). However, proof verification does not necessarily need to block the system from running the next training round because the server already has access to the aggregation result, i.e., the global model for the next round. The clients' training usually takes significant time, depending on the complexity of the model and the size of the client dataset. In RoFL, we leverage this insight to execute the client training and the server proof verification in parallel. On receiving all client update vectors, the server optimistically aggregates the vectors, attempts to decode the sum with the aggregate key from Unmask, and proceeds to the next round before verifying the proofs. The server then verifies the proofs of the previous round while the model training of the next round is already underway. If the verification of previous round succeeds, the server takes part in Unmask to reconstruct the aggregate key for the current round. Should there be any inconsistencies, e.g., a bound violation, the server aborts the new round and resets the model. However, because inconsistencies should be infrequent in practice, this optimistic approach can reduce overall wall-time significantly. Note that the server waits before initiating the second phase of secure aggregation because otherwise, a malicious client could compromise the privacy of other clients' training data. In this scenario, clients could train on a model of which the integrity is not yet verified by

	MNIST	CIFAR-10 S	CIFAR-10 L	Shakespeare
Network	CNN	LeNet5 [52]	ResNet-20 [46]	LSTM [47]
Weights	19k	62k	273k	818k
Dataset	F-MNIST [19]	CIFAR-10 [50]	CIFAR-10 [50]	Shakespeare [19]
Task	image class.	image class.	image class.	text gen.

TABLE 3: Training tasks used in our evaluation.

the server, allowing a malicious client to replace the global model in aggregation with a malicious one. This setting could be exploited by applying a range of recent privacy attacks [11, 39, 62] that are able to extract training data from maliciously altered models. Delaying Unmask means that the server cannot decrypt the global update and thus does not gain any additional information compared to the protocol without optimistic continuation.

4. Evaluation

In this section, we quantify the overhead of RoFL and show that it can be used to train practical ML models.

Implementation. We developed an end-to-end prototype of RoFL that we make available online. We implement the framework in Rust and interface it with Python to train neural networks with TensorFlow [33]. For client-server communication, we rely on the Tonic RPC framework [1] with protobuf. We use the elliptic curve Curve25519 (i.e., 126-bit security) implementation from the *dalek curve25519* library [30] for cryptographic operations. The library supports *avx2* and *avx512* hardware instructions for increased performance on supported platforms. The range proofs are implemented with the Bulletproof library [29], which builds on the same elliptic curve library.

Microbenchmarks. We use a single AWS EC2 instance (c5d.4xlarge, 16 vCPU, 32GiB, Ubuntu18.04) with support for *avx512* instructions. To evaluate the performance of the microbenchmarks on constrained client devices, we also perform the experiments on a less powerful AWS EC2 instance (t2.medium, 2 vCPU, 2GiB, Ubuntu18.04).

End-to-End benchmarks. We evaluate end-to-end system performance for four tasks on the image- and text-based datasets. Table 3 provides an overview of the tasks and models we use in our evaluation. To simulate a practical deployment, we use five AWS EC2 instances (c5d.9xlarge, 36 vCPU, 72 GiB, Ubuntu18.04) with support for *avx512* instructions and with a latency of 0.5ms between them. The server uses one instance, and 48 clients are evenly distributed over the other four instances.

4.1. Input Validation with ZKP

We begin our evaluation by investigating the overhead of integrating ZKPs for the L_∞- or L_2-norm bound checks into secure aggregation on clients and the server. In the following, we first analyze the computation and bandwidth costs of the baseline protocol and then show the benefits of using our probabilistic checking optimization. We defer discussion of the subspace learning optimization for L_2 to the end-to-end experiments.

Computation. The client-side computational costs, beyond the FL protocol itself, only include computing commitments to each parameter and creating the corresponding zero-knowledge norm proofs. This additional cost is linear in
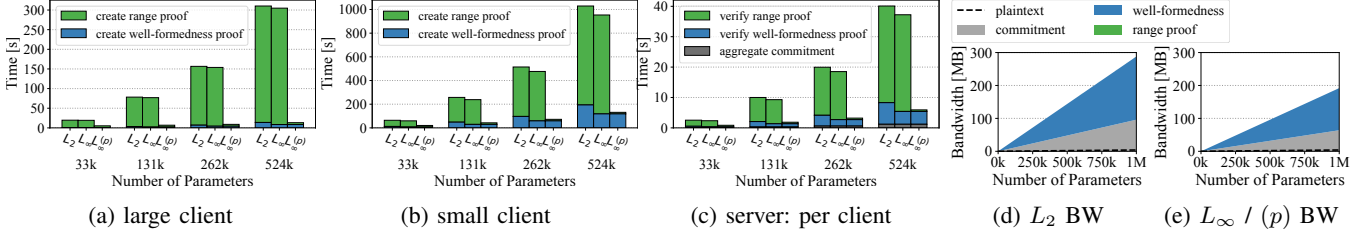
Figure 12: Microbenchmark results. Client computation cost for creating the L_2/L_∞-norm bound proof (a, b). Server computation for ZKP verification, commitment aggregation (c). Additional bandwidth cost per client to upload the update to the server (d, e).
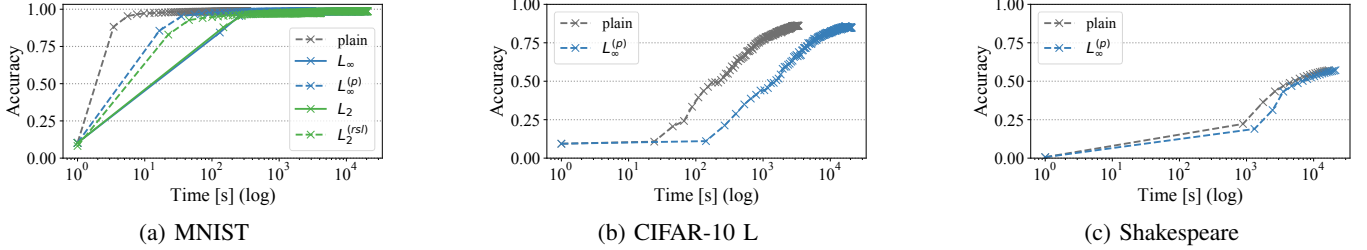

Figure 13: Model convergence over time.

the number of parameters, as seen in Figs. 12a and 12b. On a strong client (Fig. 12a), the creation of L_2 and L_∞ proofs for 262k parameters takes 157 and 154 seconds, respectively, whereas on a less powerful machine (Fig. 12b), this cost increases by 3x to 514 and 477 seconds. These costs are predominately determined by the range-proof creation, to which the additional well-formedness proofs to ensure correct encoding do not contribute significantly. Although these costs are significant, they are within an order of magnitude of the model training cost, so clients that can perform the underlying FL computation efficiently should be able to handle the overhead too.

When using L_2-norm proofs, clients need to compute additional proof-of-squares, which adds a minor overhead compared to L_∞-norm proofs. Meanwhile, the server side verifies the proofs of each client, aggregates them into the global vector, and reconstructs the aggregated vector once all updates are included. Similar to the client-side costs, the server-side verification and aggregation costs increase linearly with the number of parameters, as shown in Fig. 12c. With an update size of 262k, the server requires 19–20s per client to verify the proofs and aggregate the commitments, where the verification of the range proofs dominates the processing time. Once the clients' commitments are aggregated, the server has to solve the discrete-log for each parameter. The discrete-log solving time increases linearly with the number of parameters (i.e., 390ms for 262k and 464ms for 524k params), but the overhead is marginal compared to proof verification costs. The tasks at the server are highly parallelizable, so we can reduce the overall computation time by horizontal scaling by adding more processors.

Communication. In Figs. 12d and 12e, we show that the bandwidth requirements grow linearly in the number of parameters. The bandwidth costs are higher for the L_2-norm (Fig. 12d) than the L_∞-norm (Fig. 12e), due to the additional commitments to the square and the corresponding

proofs. The range proofs can be aggregated into a single proof, whose size increases logarithmically in the number of parameters. Thus, the bandwidth costs are dominated by the commitment size and the additional group elements required for the well-formedness proof. For 262k parameters, the data transmitted to the server per client is 75MB (72x compared to plaintext) for the L_2-norm setting and 51MB (48x) for the L_∞-norm setting. Clearly, this is a significant overhead that limits the size of the model that this technique can scale to. However, it is worth noting that while the relative size increase is dramatic, the absolute size of the updates is about the same as a few minutes of compressed HD video and, therefore, still well within the capabilities of most modern connections, including wireless and mobile connections.

Probabilistic Checking. We now show the performance improvements of probabilistic checking of L_∞ constraints. We set a maximum failure probability of 10^{-8} and assume that at least $p_v = 0.005$ of the parameters in a malicious update must exceed the bound for an attack to be successful. We provide a more thorough justification, including empirical evidence, for these choices in Appendix §B. Bandwidth is negligibly affected by probabilistic checking because the aggregated range-proofs are already logarithmically sized (Fig. 12e). However, probabilistic checking can reduce the computation cost by 6x–17x for the clients and 6x for the server (Fig. 12). The optimization shows better improvements when using larger numbers of parameters: for instance, from 154s to 9s on the large client and 19s to 3s per client server-side for 262k parameters. We conclude that probabilistic checking dramatically reduces the overhead of applying L_∞-norm constraints while offering almost the same level of protection.

Additional Setup Costs. These experiments exclude the setup cost for establishing the canceling nonces with Diffie-Hellman key exchanges between the clients. Compared to the overheads of commitments and ZKPs, these costs are

small (i.e., 9KB bandwidth and 24ms computation time per client and round for a 100-client setup).

4.2. RoFL End-to-End Performance

We demonstrate the effectiveness of RoFL in a practical deployment by showing that RoFL provides a significant speedup in computation time compared to the unoptimized L_∞- and L_2-norm constraints. For the optimized version of the L_∞-norm, we apply probabilistic checking (denoted by $L_\infty^{(p)}$), and for L_2, we compress the updates with random subspace learning (rsl, denoted by $L_2^{(rsl)}$) [53]. We also show the feasibility of deploying RoFL for practical models with a large number of parameters.

RoFL L_2 vs Secure Aggregation. We set the effective number of parameters in L_2-norm with rsl, i.e., the intrinsic dimension, for MNIST to 5000 (3.8x parameter reduction), for CIFAR-10 S to 12000 (5.2x parameter reduction), and for CIFAR-10 L to 40000 (6.8x parameter reduction) as explored by Li et al. [53]. Table 4 compares the results of the different runs. In the MNIST task, our optimization reduces the training time per round from 63x to 13x compared to secure aggregation. In the CIFAR-10 S and CIFAR-10 L experiments, we find that RoFL reduces the computation per round from 153x to 29x and from 106x to 10x. Quantization slightly reduces the accuracy for CIFAR-10 S and CIFAR-10 L, from 0.61 for plaintext to 0.60 for CIFAR-10 S, and from 0.86 to 0.85 for CIFAR-10 L. In addition, we observe a small impact on accuracy in all tasks due to the subspace ML compression, which is consistent with the work by Li et al. [53]. Moreover, the compression reduces bandwidth costs from 37x to 7x for CIFAR-10 S and from 36x to 5x for CIFAR-10 L compared to the secure aggregation baseline.

RoFL L_∞ vs Secure Aggregation. We show the use of probabilistic checking to optimize the L_∞-norm constraint for various model sizes. Similar to the microbenchmarks, we set a maximum failure probability of 10^{-8} in the optimized experiments while assuming $p_v = 0.005$ The optimization reduces the computation overhead in RoFL per round from 58x to 9x for MNIST, and from 137x to 21x for CIFAR-10 S (Table 4). The probabilistic checking optimization shows benefits even for smaller models, but it shows the most significant speedups when applied to larger models (e.g., CIFAR-10 L and Shakespeare). Although bandwidth use remains similar to unoptimized RoFL, probabilistic checking reduces the computation overhead for CIFAR-10 L by more than one order of magnitude: from 105x to 6x compared to secure aggregation. Meanwhile, for Shakespeare, it reduces the computation overhead from 6x to 1.3x. Because this model has a higher ratio of training computation to the number of parameters, the cryptographic overhead of RoFL is smaller. The larger number of parameters allows probabilistic checking to achieve an even greater relative speedup here. Model convergence is similar for the secure aggregation and, as shown in Fig. 13, and accuracy reductions due to quantization are less than 1%. As a result, although RoFL introduces a noticeable overhead, especially in terms of bandwidth,

Type	Acc.	Computation Time			Bandwidth	
		Round [s]	Total [m]	Factor	Total [GB]	Factor
MNIST (19k params, rsl 5k params, 160 rounds)						
SA	0.99	2	6	1x	1.4	1x
L_2	0.99	131	349	63x	43.0	36x
$L_2^{(rsl)}$	0.97	26	70	13x	11.2	10x
L_∞	0.99	122	325	58x	28.9	24x
$L_\infty^{(p)}$	0.99	20	53	9x	28.9	24x
CIFAR-10 S (62k params, rsl 12k params, 100 rounds)						
SA	0.61	2	3	1x	2.8	1x
L_2	0.60	280	467	153x	86.9	37x
$L_2^{(rsl)}$	0.58	53	89	29x	16.8	7x
L_∞	0.60	251	419	137x	58.3	25x
$L_\infty^{(p)}$	0.60	38	63	21x	58.3	25x
CIFAR-10 L (273k params, 160 rounds)						
SA	0.86	21	57	1x	19.7	1x
L_2 *	0.85	2250	5999	106x	612.4	36x
$L_2^{(rsl)}$	0.82	208	554	10x	89.7	5x
L_∞ *	0.85	2216	5911	104x	411.1	24x
$L_\infty^{(p)}$	0.85	131	348	6x	411.1	24x
Shakespeare (818k params, 20 rounds)						
SA	0.57	881	294	1x	3.7	1x
L_2 *	0.57	5623	1874	6x	114.6	37x
L_∞ *	0.57	5386	1795	6x	77.0	25x
$L_\infty^{(p)}$	0.57	1114	371	1.3x	77.0	25x

TABLE 4: E2E model training with RoFL (* extrapolated). Bandwidth is reported per client per round.
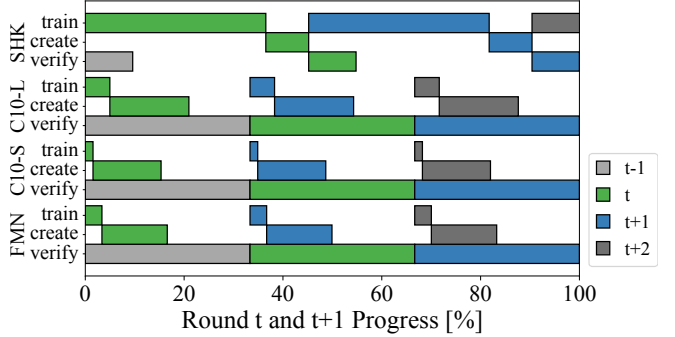

Figure 14: Optimistic continuation allows the parallelization of client-side model training and proof creation with server-side proof verification.

computation and communication in RoFL remain efficient enough to be practical for real-world deployments.

Optimistic Continuation. Fig. 14 shows the effect of optimistic continuation. The server utilizes the idle time by verifying proofs lazily after starting the next round. The optimization results in a speedup of 39% for CIFAR-10 L and 18% for Shakespeare by parallelizing proof verification with model training and proof creation. For the Shakespeare, proof verification requires no additional wall-clock time, because model training and proof creation in $t + 1$ take longer than the proof verification of t round.

5. Related Work

In the interest of space, we review related work solely on cryptographic systems similar to RoFL; we have discussed the wider field of related work on FL robustness in §2.

Secure Data Collection. Secure aggregation systems [2, 21, 24, 27, 51, 55, 56, 59, 66, 69, 71, 77] enable users

to compute sums over their private data without disclosing the inputs. A variety of secure aggregation protocols have emerged to meet the needs of a wide range of applications that introduce new security, scalability, and performance requirements. Bonawitz et al. [13] and Bell et al. [6] introduce a secure aggregation protocol designed for FL based on canceling random masks; their protocol builds on [2, 59]. However, none of these systems address integrity. The protocol used by RoFL without ZKP is similar to that proposed by Shi et al. [77]; however, ours is tailored to FL and does not assume a trusted setup. Kursawe et al. [51] extend a variation of the protocol by Shi et al. [77] with ZKP to provide correct execution guarantees but does not constrain client inputs. Canny et al. [21] propose a private peer-to-peer machine learning platform that allows clients to verify the L_2-norm of inputs. However, it exhibits high overheads that make it unsuitable for the FL setting. A different body of work in this space [3, 14, 27, 28, 31] considers a non-colluding trust model in which the clients split the input values into shares and submit them to non-colluding servers. This enables efficient input checks, as expensive cryptographic operations can be avoided on the assumption that the servers do not collude. However, we know from our discussions with FL industry players that organizations have struggled to deploy this trust model in practice, even when, e.g., using an external non-profit organization. Thus, the realistic deployment of the non-colluding trust model remains an issue hindering the deployment of these protocols in practice.

FL with Differential Privacy. RoFL opts for a cryptographic solution to hide client updates from the server. An alternative approach to client update privacy is to employ differentially-private mechanisms that perturb the local client updates [43, 58, 64, 80]. However, a pure perturbation-based approach for privacy in the FL setting that guarantees local differential privacy (i.e., hiding updates) requires clients to add a substantial amount of noise [43, 64, 80], significantly reducing utility, making it impractical in many situations. RoFL composes well with differential-privacy techniques for output privacy by enforcing norm bounds on clients' updates but one must navigate the trade-off between utility, privacy, fairness, and robustness (§2.5). Still, we believe that perturbation-based differential privacy can be an orthogonal feature that RoFL can be combined with to increase robustness [5, 60, 79, 82] and privacy further.

Acknowledgments

References

[1] "Tonic RPC framework," Online: https://github.com/hyperium/tonic, May 2021.

[2] G. Ács and C. Castelluccia, "I have a DREAM! (DiffeRentially private smart metering)," in *Information Hiding*. Springer Berlin Heidelberg, 2011, vol. 6958, pp. 118–132.

[3] S. Addanki, K. Garbe, E. Jaffe, R. Ostrovsky, and A. Polychroniadou, "Prio+: Privacy Preserving Aggregate Statistics via Boolean Shares," *Cryptology ePrint Archive cryptoeprint:2021/576*, 2021.

[4] E. Bagdasaryan, O. Poursaeed, and V. Shmatikov, "Differential privacy has disparate impact on model accuracy," in *NeurIPS*, 2019.

[5] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How To Backdoor Federated Learning," in *AISTATS*, 2020, pp. 2938–2948.

[6] J. Bell, K. A. Bonawitz, A. Gascón, T. Lepoint, and M. Raykova, "Secure Single-Server Aggregation with (Poly)Logarithmic Overhead," in *ACM CCS*, 2020.

[7] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *ICML*, 2019, pp. 634–643.

[8] B. Biggio, B. Nelson, and P. Laskov, "Poisoning Attacks against Support Vector Machines," in *ICML*, 2012, p. 1467–1474.

[9] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent," in *NeurIPS*, 2017, pp. 119–129.

[10] M. Blum, P. Feldman, and S. Micali, "Non-interactive zero-knowledge and its applications," in *ACM STOC*. ACM, 1 Jan. 1988, pp. 103–112. [Online]. Available: http://dx.doi.org/10.1145/62212.62222

[11] F. Boenisch, A. Dziedzic, R. Schuster, A. S. Shamsabadi, I. Shumailov, and N. Papernot, "When the curious abandon honesty: Federated learning is not private," *arXiv preprint arXiv:2112.02918*, 2021.

[12] J. Böhler and F. Kerschbaum, "Secure Multi-party Computation of Differentially Private Median," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 2147–2164.

[13] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for Privacy-Preserving machine learning," in *ACM CCS*, 2017, pp. 1175–1191.

[14] D. Boneh, E. Boyle, H. Corrigan-Gibbs, N. Gilboa, and Y. Ishai, "Lightweight Techniques for Private Heavy Hitters," in *IEEE Symposium on Security and Privacy (SP)*, 2021.

[15] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(leveled) fully homomorphic encryption without bootstrapping," *ACM Trans. Comput. Theory*, vol. 6, no. 3, pp. 13:1–13:36, Jul. 2014. [Online]. Available: http://doi.acm.org/10.1145/2633600

[16] D. R. Brendan McMahan, "Federated Learning: Collaborative Machine Learning without Centralized Training Data," Online: https://ai.googleblog.com/2017/04/federated-learning-collaborative.html, 2017.

[17] T. S. Brisimi, R. Chen, T. Mela, A. Olshevsky, I. C. Paschalidis, and W. Shi, "Federated learning of predictive models from federated electronic health records," *Int. J. Med. Inform.*, vol. 112, pp. 59–67, Apr. 2018.

[18] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short proofs for confidential transactions and more," in *IEEE Symposium on Security and Privacy (SP)*, May 2018, pp. 315–334. [Online]. Available: http://dx.doi.org/10.1109/SP.2018.00020

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

[20] J. Camenisch and M. Stadler, "Proof systems for general statements about discrete logarithms," *Technical report/Dept. of Computer Science, ETH Zürich*, vol. 260, 1997.

[21] J. Canny, "Collaborative Filtering with Privacy," in *IEEE Symposium on Security and Privacy (SP)*, 2002, pp. 45–57.

[22] N. Carlini, C. Liu, Ú. Erlingsson, J. Kos, and D. Song, "The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks," in *USENIX Security*, 2019.

[23] N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson, A. Oprea, and C. Raffel, "Extracting training data from large language models," in *USENIX Security*, 2021.

[24] C. Castelluccia, A. C.-F. Chan, E. Mykletun, and G. Tsudik, "Efficient and provably secure aggregation of encrypted data in wireless sensor networks," *ACM Trans. Sen. Netw.*, vol. 5, no. 3, pp. 20:1–20:36, Jun. 2009. [Online]. Available: http://doi.acm.org/10.1145/1525856.1525858

[25] L. Chen, H. Wang, Z. B. Charles, and D. S. Papailiopoulos, "DRACO: byzantine-resilient distributed training via redundant gradients," in *ICML*, 2018, pp. 902–911. [Online]. Available: http://proceedings.mlr.press/v80/chen18l.html

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

[27] H. Corrigan-Gibbs and D. Boneh, "Prio: private, robust, and scalable computation of aggregate statistics," in *USENIX NSDI*, 2017, pp. 259–282.

[28] H. N. Corrigan-Gibbs, "Protecting Privacy by Splitting Trust," Ph.D. dissertation, Stanford University, 2019.

[29] D. Cryptography, "Rust Bulletproofs Library," Online: https://github.com/dalek-cryptography/bulletproofs, August 2020.

[30] ——, "Rust Curve25519 Library," Online: https://github.com/dalek-cryptography/curve25519-dalek, August 2020.

[31] Y. Duan, N. Youdao, J. F. Canny, and J. Z. Zhan, "P4P: Practical Large-Scale Privacy-Preserving distributed computation robust against malicious users," in *USENIX Security*, 2010, pp. 207–222. [Online]. Available: http://www.usenix.org/events/sec10/tech/full_papers/Duan.pdf

[32] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," in *Advances in Cryptology*, G. R. Blakley and D. Chaum, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1985, pp. 10–18.

[33] M. A. et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

[34] P. K. et al., "Advances and Open Problems in Federated Learning," *arXiv preprint arXiv:1912.04977*, 2019.

[35] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to Byzantine-robust federated learning," in *USENIX Security*, 2020, pp. 1605–1622.

[36] V. Feldman, "Does learning require memorization? a short tale about a long tail," in *ACM STOC*, 2020, p. 954–959.

[37] V. Feldman and C. Zhang, "What neural networks memorize and why: Discovering the long tail via influence estimation," in *NeurIPS*, 2020, pp. 2881–2891.

[38] A. Fiat and A. Shamir, "How to Prove Yourself: Practical Solutions to Identification and Signature Problems," in *CRYPTO*, 1987, pp. 186–194.

[39] L. Fowl, J. Geiping, W. Czaja, M. Goldblum, and T. Goldstein, "Robbing the fed: Directly obtaining private data in federated learning with modified models," October 2021. [Online]. Available: http://arxiv.org/abs/2110.13057

[40] C. Fung, C. J. M. Yoon, and I. Beschastnikh, "Mitigating Sybils in Federated Learning Poisoning," *arXiv preprint arXiv:1808.04866*, 2018.

[41] H. García Navarro, "Design and implementation of the circom 1.0 compiler," Master's thesis, Universidad Complutense de Madrid, 2020. [Online]. Available:

https://eprints.ucm.es/id/eprint/62475/

[42] J. Geiping, H. Bauermeister, H. Droge, and M. Moeller, "Inverting Gradients - How Easy is It to Break Privacy in Federated Learning?" in *NeurIPS*, 2020.

[43] R. C. Geyer, T. Klein, and M. Nabi, "Differentially Private Federated Learning: A Client Level Perspective," *arXiv preprint arXiv:1712.07557*, 2017.

[44] J. Groth, "On the size of Pairing-Based non-interactive arguments," in *Advances in Cryptology – EURO-CRYPT 2016*. Springer Berlin Heidelberg, 2016, pp. 305–326.

[45] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," *arXiv preprint arXiv:1708.06733*, 2017.

[46] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[47] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, p. 1735–1780, nov 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[48] Y. Ji, X. Zhang, S. Ji, X. Luo, and T. Wang, "Model-reuse attacks on deep learning systems," in *ACM CCS*, 2018, pp. 349–363.

[49] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtarik, A. T. Suresh, and D. Bacon, "Federated Learning: Strategies for Improving Communication Efficiency," in *NeurIPS Workshop on Private Multi-Party Machine Learning*, 2016.

[50] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[51] K. Kursawe, G. Danezis, and M. Kohlweiss, "Privacy-Friendly aggregation for the Smart-Grid," in *Privacy Enhancing Technologies*. Springer Berlin Heidelberg, 2011, pp. 175–191. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-22263-4_10

[52] Y. LeCun *et al.*, "LeNet-5, convolutional neural networks," http://yann.lecun.com/exdb/lenet/, 2015.

[53] C. Li, H. Farkhoor, R. Liu, and J. Yosinski, "Measuring the Intrinsic Dimension of Objective Landscapes," in *ICLR*, 2018.

[54] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang, "Trojaning Attack on Neural Networks," in *NDSS*, 2018.

[55] Lukas Burkhalter and Anwar Hithnawi and Alexander Viand and Hossein Shafagh and Sylvia Ratnasamy, "TimeCrypt: Encrypted Data Stream Processing at Scale with Cryptographic Access Control," in *USENIX NSDI*, 2020.

[56] Lukas Burkhalter, Nicolas Kuchler, Alexander Viand,

Hossein Shafagh, Anwar Hithnawi, "Zeph: Cryptographic Enforcement of End-to-End Data Privacy," in *USENIX OSDI*, 2021.

[57] 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.

[58] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," in *6th International Conference on Learning Representations, ICLR 2018*, 2018.

[59] L. Melis, G. Danezis, and E. De Cristofaro, "Efficient private statistics with succinct sketches," in *NDSS*, 2016.

[60] M. Naseri, J. Hayes, and E. De Cristofaro, "Local and central differential privacy for robustness and privacy in federated learning," in *NDSS*, 2022.

[61] X. Pan, M. Zhang, D. Wu, Q. Xiao, S. Ji, and Z. Yang, "Justinian's GAAvernor: Robust Distributed Learning with Gradient Aggregation Agent," in *USENIX Security*, 2020, pp. 1641–1658.

[62] D. Pasquini, D. Francati, and G. Ateniese, "Eluding secure aggregation in federated learning via model inconsistency," November 2021. [Online]. Available: http://arxiv.org/abs/2111.07380

[63] T. P. Pedersen, "Non-Interactive and Information-Theoretic secure verifiable secret sharing," in *CRYPTO*. Springer Berlin Heidelberg, 1992, pp. 129–140.

[64] V. Pihur, A. Korolova, F. Liu, S. Sankuratripati, M. Yung, D. Huang, and R. Zeng, "Differentially-Private "Draw and Discard" Machine Learning," *arXiv preprint arXiv:1807.04369*, 2018.

[65] J. M. Pollard, "Monte carlo methods for index computation (mod p)," *Mathematics of Computation*, vol. 32, no. 143, p. 918, Jul. 1978. [Online]. Available: http://dx.doi.org/10.2307/2006496

[66] R. A. Popa, A. J. Blumberg, H. Balakrishnan, and F. H. Li, "Privacy and accountability for location-based aggregate statistics," in *ACM CCS*, 2011, pp. 653–666.

[67] S. Rajput, H. Wang, Z. Charles, and D. Papailiopoulos, "Detox: A redundancy-based framework for faster and more robust gradient aggregation," in *NeurIPS*, 2019.

[68] S. Ramaswamy, R. Mathews, K. Rao, and F. Beaufays, "Federated Learning for Emoji Prediction in a Mobile Keyboard," *arXiv preprint arXiv:1906.04329*, 2019.

[69] L. Reyzin, A. Smith, and S. Yakoubov, "Turning HATE Into LOVE: Compact Homomorphic Ad Hoc Threshold Encryption for Scalable MPC," Cryptology ePrint Archive, Report 2018/997, 2018, https://ia.cr/2018/997.

[70] N. Rieke, J. Hancox, W. Li, F. Milletarì, H. R. Roth, S. Albarqouni, S. Bakas, M. N. Galtier, B. A. Landman, K. Maier-Hein, S. Ourselin, M. Sheller, R. M. Summers, A. Trask, D. Xu, M. Baust, and M. J. Cardoso, "The future of digital health with federated learning," *NPJ Digit Med*, vol. 3, p. 119, Sep. 2020.

[71] E. Roth, D. Noble, B. H. Falk, and A. Haeberlen, "Honeycrisp: large-scale differentially private aggregation without a trusted core," in *ACM SOSP*, 2019, pp. 196–210.

[72] H. Shafagh, A. Hithnawi, L. Burkhalter, P. Fischli, and S. Duquennoy, "Secure sharing of partially homomorphic encrypted IoT data," in *ACM SenSys*, 2017, pp. 29:1–29:14.

[73] V. Shejwalkar, A. Houmansadr, P. Kairouz, and D. Ramage, "Back to the Drawing Board: A Critical Evaluation of Poisoning Attacks on Production Federated Learning," in *IEEE Symposium on Security and Privacy (SP)*, 2022, pp. 1117–1134.

[74] M. J. Sheller, B. Edwards, G. A. Reina, J. Martin, S. Pati, A. Kotrotsou, M. Milchenko, W. Xu, D. Marcus, R. R. Colen, and S. Bakas, "Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data," *Sci. Rep.*, vol. 10, no. 1, p. 12598, Jul. 2020.

[75] M. J. Sheller, G. A. Reina, B. Edwards, J. Martin, and S. Bakas, "Multi-Institutional deep learning modeling without sharing patient data: A feasibility study on brain tumor segmentation," *Brainlesion*, vol. 11383, pp. 92–104, Jan. 2019.

[76] S. Shen, S. Tople, and P. Saxena, "Auror: Defending against poisoning attacks in collaborative deep learning systems," in *ACM ACSAC*, 2016, pp. 508–519.

[77] E. Shi, H. T. H. Chan, E. Rieffel, R. Chow, and D. Song, "Privacy-preserving aggregation of time-series data," in *NDSS*, 2011.

[78] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 3–18.

[79] A. T. Suresh, B. McMahan, P. Kairouz, and Z. Sun, "Can you really backdoor federated learning?" in *Federated learning workshop at NeurIPS*, 2019.

[80] S. Truex, L. Liu, K.-H. Chow, M. E. Gursoy, and W. Wei, "LDP-Fed: Federated learning with local differential privacy," in *Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking*, 2020, pp. 61–66.

[81] G. Van Horn and P. Perona, "The Devil is in the Tails: Fine-grained Classification in the Wild," *arXiv preprint arXiv:1709.01450*, 2017.

[82] 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, pp. 16 070–16 084.

[83] Y. Wen, J. Geiping, L. Fowl, H. Souri, R. Chellappa, M. Goldblum, and T. Goldstein, "Thinking two moves ahead: Anticipating other users improves backdoor attacks in federated learning," 2022. [Online]. Available: https://arxiv.org/abs/2210.09305

[84] C. Xie, K. Huang, P.-Y. Chen, and B. Li, "DBA: Distributed Backdoor Attacks against Federated Learning," in *ICLR*, 2020.

[85] C. Xie, O. Koyejo, and I. Gupta, "Fall of empires: Breaking byzantine-tolerant SGD by inner product manipulation," in *UAI*, vol. 115, 2020, pp. 261–270.

[86] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates," in *ICML*, 2019.

[87] H. Yin, A. Mallya, A. Vahdat, J. M. Alvarez, J. Kautz, and P. Molchanov, "See through gradients: Image batch recovery via gradinversion," in *CVPR*, 2021.

[88] C. Zhang, S. Li, X. Junzhe, , W. Wang, F. Yan, and L. Yang, "BatchCrypt: Efficient Homomorphic Encryption for Cross-Silo Federated Learning," in *USENIX ATC*, 2020.

[89] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," in *ICLR*, 2017.

[90] Z. Zhang, A. Panda, L. Song, Y. Yang, M. Mahoney, P. Mittal, R. Kannan, and J. Gonzalez, "Neurotoxin: Durable backdoors in federated learning," in *ICML*, ser. Proceedings of Machine Learning Research, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, Eds., vol. 162. PMLR, 2022, pp. 26 429–26 446. [Online]. Available: https://proceedings.mlr.press/v162/zhang22w.html

[91] X. Zhu, D. Anguelov, and D. Ramanan, "Capturing long-tail distributions of object subcategories," in *IEEE CVPR*, 2014, pp. 915–922.

Appendix A.

We present additional experiments for the robustness analysis in this appendix.

Untargeted attacks. In addition to targeted attacks, we consider the performance untargeted attacks for various norm bounds. We use the state-of-the-art Projected Gradient Ascent (PGA) attack proposed by Shejwalkar et al. [73] in our setup. The authors show the performance of this attack for different fractions of compromised clients. We take an orthogonal approach and study the impact of the choice of norm bound on the attack. We use the same setup as for targeted attacks which allows the attacker to perform an attack for 500 rounds. We define *Attack Impact* as the
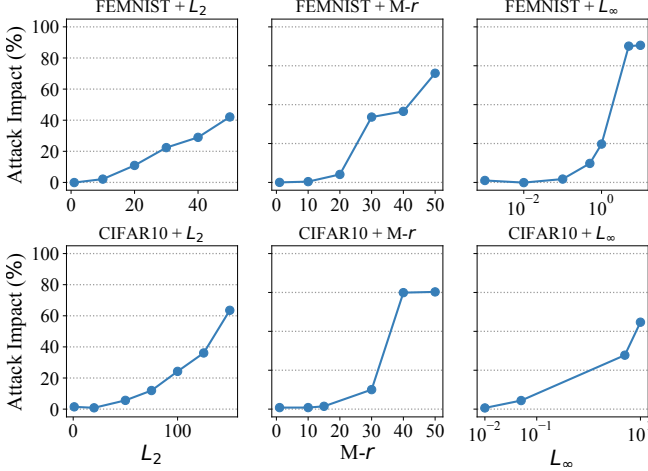
Figure 15: Attack impact of the state-of-the-art untargeted PGA attack.

(a) **FMN-P**   (b) **C10-P**

Figure 16: Comparison of model poisoning attacks under various median bounds.

(a) **FMN-P**   (b) **C10-P**

Figure 17: Comparison of model poisoning attacks for various % of compromised clients per round. (M-r: 1.5)

difference between the accuracy of the unattacked global model and the attacked model after 500 rounds. To account for per-round variance of model performance, we compute the accuracy as the average over the last 20 rounds. For tight norm bounds (**FMN**: L_2-B < 10, r-M < 15, L_∞-B < 0.1, **C10**: L_2-B < 20, r-M < 15, L_∞-B < 0.05), untargeted attack impact is negligible (i.e., less than 6%, Fig. 15). As the bound loosens, attack impact becomes noticeable. However, when comparing to targeted attacks, untargeted attacks are only effective under a much larger bound than targeted attacks. For instance, untargeted attacks start to become effective for a median-based bound r-M $>= 30$, whereas targeted attacks already have significant success for r-M $>= 15$ (Figs. 4c and 4d). This suggests that strong untargeted attacks are easier to prevent with a norm bound than targeted attacks.

Comparison of model poisoning attacks. We evaluate the three model poisoning attack strategies (MP-AT, MP-PD, MP-NT) under adaptive norm bounds in Fig. 16 to complement the comparison under static bounds (Fig. 5 in the analysis). None of the attacks are successful under an appropriate multiplier of the median norm M-r = 1.5. For a looser bound (M-r: 15), the attack strategies are successful and their relative performance diverges, with MP-PD achieving highest malicious accuracy for **FMN** and MP-NT for **C10**.

Growing number of attackers. We compare the performance of data poisoning with all three model poisoning in Fig. 17. All attacks show similar performance relative to the % of compromised clients per round, with MP-AT achieving the highest malicious accuracy improvement over data poisoning (at most 40%) as shown in Section §2.

L_∞**-norm bound.** we show that an L_∞-norm bound has the ability to defend against prototypical backdoor attacks, similar to the static and adaptive L_2-norm bounds shown in Section §2. An L_∞-norm bound of 0.01 for **FMN** and 0.05 for **C10** is sufficient to successfully defend against a backdoor attack on prototypical samples, without impacting model convergence or accuracy (Fig. 18). Similar to the
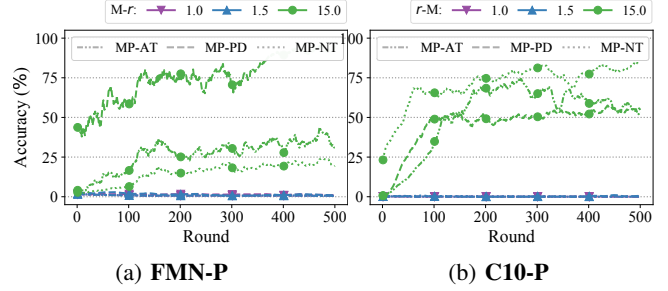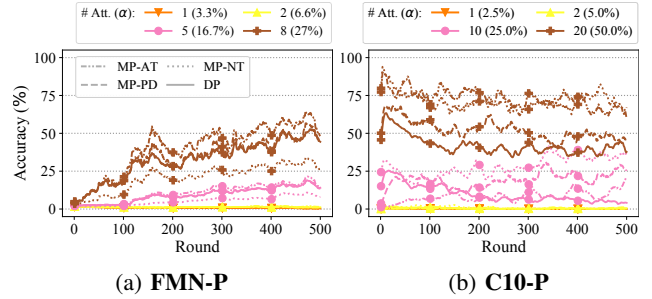
L_2-norm bound, when the norm bound is too tight, model convergence is impacted. Conversely, when the bound is too loose, the attacker is able to successfully inject the prototypical backdoor into the model.

Appendix B.

In this appendix, we discuss the security of the probabilistic checking optimization for L_∞-norm constraints in practical model poisoning attacks and justify our choice of security parameters.

The security of probabilistic checking is determined by the probability that a client is not detected (i.e., the failure probability) by the server while submitting a malicious update that violates the L_∞-norm constraint. Let $\mathbf{w}$ be a malicious update vector of length ℓ that violates the L_∞-norm constraint, i.e., $||\mathbf{w}||_\infty > B$, because $p_v \cdot \ell$ for $p_v \in (0, 1]$ parameters in $\mathbf{w}$ are above B. The server selects a fraction $p_c \in (0, 1]$ of the parameters $\mathbf{w}'$ in $\mathbf{w}$ uniformly at random and checks whether $||\mathbf{w}'||_\infty > B$. We can derive the probability of not detecting this malicious update vector by realizing this random selection by the server is equivalent to drawing $p_c \cdot \ell$ parameters without replacement. The probability of not detecting is then defined as the probability that all $p_c \cdot \ell$ draws are below-bound parameters for a pool in which $1-(p_v \cdot \ell)$ out of ℓ parameters are below the bound. This probability $\Pr(X = k)$ follows the hypergeometric distribution for k draws. Specifically, we are interested in the case when $k = p_c \cdot \ell$, which defines the
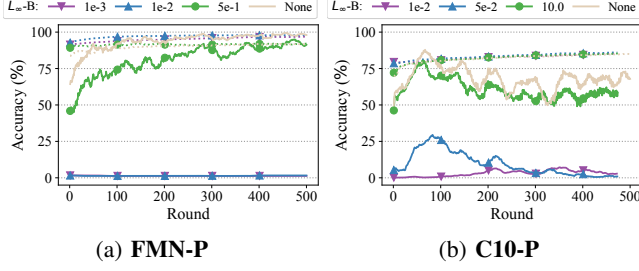
(a) **FMN-P**

(b) **C10-P**

Figure 18: Comparison of PGD attack under various static L_∞-norm bounds.

failure probability as

$$\Pr(X = p_c\ell) = Hyp(p_c\ell \mid \ell, \ell \cdot (1 - p_v), p_c\ell)$$

$$= \frac{\binom{\ell(1-p_v)}{p_c \cdot \ell}\binom{\ell-\ell\cdot(1-p_v)}{p_c\ell-p_c\ell}}{\binom{\ell}{p_c\ell}}$$

$$= \frac{\binom{\ell(1-p_v)}{p_c\ell}}{\binom{\ell}{p_c\ell}}.$$

The security of the probabilistic checking approach depends on the assumption of the fractions p_v and p_c, where p_c can be set by the server. To configure p_c, RoFL assumes a fixed minimal p_v and a bounded failure probability of $1e-8$. In Fig. 19b, we present the number of parameters the server has to check for a fixed detection failure probability δ and fraction p_v of parameters assumed to be above the bound. We argue that in practical model poising attacks that depend on scaling, a significant fraction (i.e., $\geq p_v$) of parameters have to be above the bound for an attack to be successful. Here, we analyze the fractions of parameters above the bound for generic attacks from §2, and then show results for an adaptive attack in which the attacker optimizes for the number of scaled parameters.

Generic Attacks. Fig. 20a shows the update weight distribution of a benign and a malicious update for the **FMN-P** single-shot attack. For the malicious update, 81.8% of parameters lie outside the L_∞-bound of 0.01, versus 0.87% for the benign update. In the continuous attack scenario, **FMN-P**, a similarly large fraction of parameters greater than 47% are above the bound in each round (Fig. 20b).

Adaptive attacks. In the adaptive case, we consider an adversary that optimizes for a minimal number of parameters above the bound in the poisoned update. The adaptive attacker optimizes their model poisoning update by selecting a fraction of K weights to scale based on their magnitude. We instantiate the following *top-K* adaptive attacker: Upon receiving the global model, the attacker crafts a malicious update as in the generic case, but scales only the $K \cdot \ell$ largest weights, clipping the remaining weights to the bound before sending the update to the server. Fig. 21 shows the performance of the top-K **C10-P** single-shot attack for varying percentages of K. Attacks are always successful for $K \geq 10\%$, which means the attacker does not have to
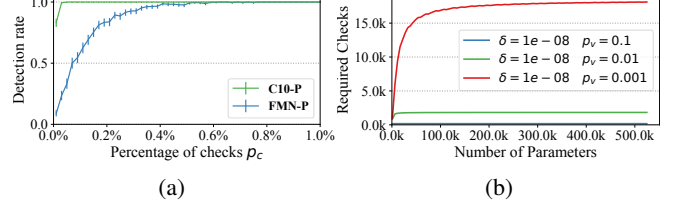

(a)   (b)

Figure 19: (a) The server only has to check a small fraction of parameters p_c to ensure a high detection rate on two malicious updates created using the *top-K* adaptive attack. (b) Number of update parameter ranges the server has to check with a failure probability of δ and the assumption that a fraction p_v of the parameters are above the bound.
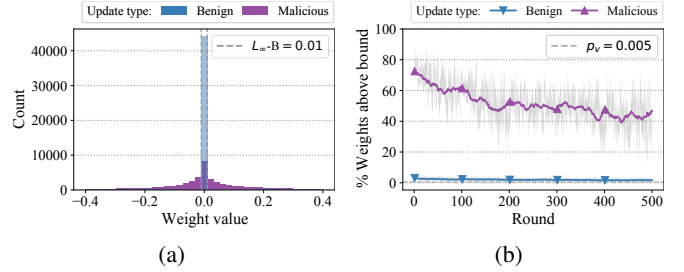

(a)   (b)

Figure 20: Generic attacks: (a) Weight distribution of a malicious update (**FMN**, **FMN-P**). Parameters outside of the interval $[-0.4, 0.4]$ are hidden. (b) Percentage of weights outside the L_∞-norm bound of 0.01 for malicious and benign clients per round (**FMN**, **FMN-P**). The total number of weights is 44426. We plot the assumption $p_v = 0.005$ as used in the evaluation.

scale the smallest 90% of weights to attack successfully. However, attack success starts to decrease for $K < 10\%$, with success below 50% for $K = 5\%$. Finally, the attack becomes completely ineffective (0% success) for $K \leq 2\%$. In **FMN-P**, the attacker has to scale less update parameters for success, but still a significant fraction of over 2 percent must be above the bound. In Fig. 19a, we show the effect of varying p_c on the detection rate for specific top-K attack updates in both **FMN-P** and **C10-P** tasks. The detection rate quickly converges to 1.0 as p_c increases. In our analysis, we opt for 0.005 as a conservative estimate for p_v, which is well below the observations in the generic and adaptive attacks.

Appendix C.

C.1. Analysis Setup

We first describe the details of the tasks in the analysis, and then continue with the configurations of the attacks for these tasks.

Analysis Tasks. We consider two image classification tasks that have been used before in the context of FL backdoor attacks to provide comparison to previous work. The first task (**FMN**) is a digit classification task on the Federated-MNIST dataset, with $n = 3383$ available clients and $m =$
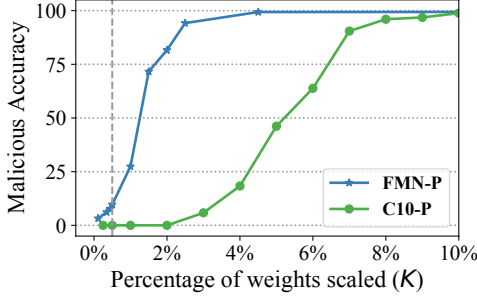
Figure 21: Top-K success attack on probabilistic checking.

Client	FMN	C10
# Training samples	341873	50000
Model	LeNet5	ResNet-20
# Classes	10	10
# Total number of clients	3383	100
# Selected clients per round	30	40
# Local epochs per round	5	2
Batch size	32	64
Optimizer	SGD	
Learning rate	0.01	0.02

TABLE 5: Hyperparameters for honest clients.

30 clients selected per learning round, following [79, 82]. This dataset consists of samples from 3383 hand writers and is inherently non-IID because of the per-writer grouping, which simulates a natural distribution of training data. The second task (**C10**) is an image classification on the CIFAR-10 dataset. We divide the dataset among $n = 100$ clients and select $m = 40$ clients in each round. We distribute the images to clients in a non-IID fashion using a Dirichlet distribution with parameter 0.9 in all dimensions. The full hyperparameters for the setups of **FMN** and **C10** are shown in Table 5. For **C10**, we use standard data augmentations (random horizontal flips and shifts) to reduce overfitting on the training dataset.

The global learning rate η controls the fraction of the model that is updated each round. For **FMN**, the full model is updated with the average of $m = 30$ client updates ($\eta = \frac{3383}{30}$). For **C10**, we follow [5] and use a global learning rate of 1, indicating that 40% of the model is replaced every round. We pre-train the models for both tasks once and use the same weights as a starting point for each experiment, similar to [82]. The pre-trained models have not yet fully converged, to show the effect of different configurations on model convergence. For **FMN**, we use a pre-trained LeNet5 with 88% test accuracy and for **C10** we use a ResNet-20 with 80% test accuracy. The full configuration of the models are given in §C.2.1. All experiments in our analysis, except those in Figs. 3 and 8, show the accuracy as a moving average with a window size of 20, to improve readability.

Attack Tasks. For the **FMN** task, we consider a backdoor attack targeting prototypical inputs that classifies images containing the number 7 from 30 randomly selected hand-writers as the number one instead (**FMN-P**, 310 images in total), following [79]. For a tail backdoor attack, we follow [82] and use images of sevens from the ARDIS dataset that are mostly in the European style (with a horizontal bar in the middle), unlike the sevens in F-MNIST (**FMN-T**, 660 images). On the **C10** task, we use a prototypical backdoor attack from [5]: classifying images of green cars as birds (**C10-P**, 30 images). For the tail backdoor attack, we use images from airplanes of Southwest airline that are collected by [82] to be introduced as tail backdoor images for CIFAR-10 (**C10-T**, 784 images).

Attack Success. For the **FMN-P** attack, attacker success is defined by the malicious classification accuracy on the backdoor images. For the **C10-P** attack, we measure ac-

curacy using three backdoor samples that were not made available to the attacker during training. Test images are augmented into 200 versions per test image and used to evaluate the attacker success in the global model [5]. For the tail attacks (**FMN-T**, **C10-T**), the images are already grouped into train and test sets, the latter of which are used to evaluate accuracy.

Attack Strategies. We discuss the configuration of the backdoor injection attacks used by the adversary to create a malicious model update. We explore two attacker models: data poisoning and model poisoning. In data poisoning strategy, the attacker follows the training process (i.e., hyperparameters) as the benign clients, but is allowed to insert a number of malicious samples into each batch according to the poison ratio. For model poisoning, we explore three different attacks that are adaptive with regards to the norm bound that is enforced by the server.

The model poisoning clients train the model for a fixed amount of SGD steps, determined by the number of epochs and batches shown in Table 6. Each batch contains benign samples, and a number of target samples mislabeled with the target class. The amount of malicious samples in each batch is controlled by the poison ratio. The optimal backdoor injection attack also includes benign samples because it allows the attacker to move the model into a direction where both objectives are satisfied simultaneously, making the attacker's update harder to reverse by honest client updates [5, 82]. We apply learning rate decay to malicious training to ensure the model update has a good accuracy on both the main and backdoor tasks. The learning rate starts at 0.1 and decays step-wise with $\frac{1}{10}$ every $\frac{1}{3}$ of the total number of steps. We now discuss details specific to the attacks.

PGD When an L_p-norm bound B is present, the adversary uses Projected Gradient Descent (PGD) to adaptively craft malicious updates. Concretely, after every SGD step, the adverary projects model update $\Delta\hat{\mathbf{w}}$ back to a point in the constraint set. For the norm bound, this set is defined as $||\Delta\hat{\mathbf{w}}||_p \leq \frac{B}{\gamma}$, where γ is the scaling factor. More precisely, similar to [82], during an epoch we project the update back onto a slightly larger space ($1.2\frac{B}{\gamma}$). Only after the last SGD step of the epoch do we use γ as a scaling factor to satisfy the norm constraint (Table 6). In practice, this approach leads to a higher attacker success, similar as to what was reported in [82].

Setup	FMN	C10
# Local epochs per round	10	6
# Batches	25	10
Batch size	32	64
# Poison samples per batch	12	20
Optimizer	SGD	
Learning rate	0.1	0.1
Scale factor	30	100

TABLE 6: Hyperparameters for model poisoning attacks

Neurotoxin We choose a mask ratio $k = 5$ which means that the malicious update is projected onto the bottom 95% of frequently updated weights using PGD. This value for the mask ratio achieved good overall performance in both the original attack paper and in our results.

Anticipate We follow the hyperparameters used in the original attack as closely as possible and set the number of anticipate steps $k = 9$ for **C10**. We choose $k = 5$ anticipate steps for **FMN** because Federated-MNIST dataset was not used to evaluate the attack.

C.2. RoFL Evaluation Setup

We now describe additional details of the experiments performed as part of the evaluation.

Microbenchmarks. For each reported result, we take the average of 4 measurements recorded after a single warm-up mock execution.

End-to-end. We evaluate the end-to-end performance on four different models. We use the two models that are also used in analysis and we evaluate on two additional models. ResNet-20 [46] (CIFAR-10 L, 273k params) and LeNet5 [52] (CIFAR-10 S, 62k params). In addition, we evaluate the performance for a smaller CNN network (MNIST, 19k params) and a text-generative LSTM model (Shakespeare, 818k params). An overview of the tasks

LeNet5. We make use of the standard LeNet5 [52] convolutional neural network for both **FMN** in the analysis

and models is shown in Table 3. The configuration of the end-to-end setup experiments are similar to those of the analysis, except for some minor changes. We increase the number of clients in the evaluation to 48 to benchmark the performance of a larger deployment. In each training round, all 48 clients are selected. For MNIST, we use a learning rate of 0.05, for CIFAR-10 S a learning rate of 0.01, for CIFAR-10 L a learning rate of 0.05 and for Shakespeare 0.3. In RoFL, we apply 8-bit probabilistic quantization with 7 fractional bits for the floating point numbers in the client updates for encryption.

C.2.1. Model Architectures. We use the following model architectures for analysis and evaluation (end-to-end) experiments.

CNN. For the MNIST task, we use a standard convolutional neural network (CNN) architecture, consisting of two convolution layers of 8 and 4 filters, respectively, with kernel size 3-by-3. These are followed by a 2-by-2 max-pooling layer, a fully connected layer of 32 units and then the output layer. The model has 19k trainable parameters.

and CIFAR-10 S. Note that the input layers for both applications of the model are different: 28-by-28, 1 channel for **FMN** and 32-by-32, 3 channels for CIFAR-10 S.

ResNet-20. We use the ResNet [46] model for experiments in the analysis (**FMN**) as well as in evaluation experiments (CIFAR-10 L). We use the ResNet implementation provided by the Keras ResNet CIFAR-10 example[5], corresponding to the ResNet20v1 model.

LSTM. We use the LSTM architecture as used in the LEAF federated learning benchmark [19] and the original federated learning paper [57]. The architecture consists of an embedding layer of size 8 followed by a two-layer stacked LSTM with 256 hidden units. The LSTMs are connected to dense output layer with width 80, which corresponds to the vocabulary size.

---

5. https://github.com/rstudio/keras/blob/master/vignettes/examples/cifar10_resnet.py