SECURE BYZANTINE-ROBUST FEDERATED LEARNING WITH DIMENSION-FREE ERROR

Anonymous authors
Paper under double-blind review

ABSTRACT

In the present work, we propose a federated learning protocol with *bi-directional security guarantees*. First, our protocol is Byzantine-robust against malicious clients. Additionally, it is the first federated learning protocol with a per-round mean estimation error that is *independent of the update size* (*e.g.*, the size of the model being trained). Second, our protocol is secure against a semi-honest server, as it only reveals sums of the updates. The code for evaluation is provided in the supplementary material.

1 INTRODUCTION

Federated learning (FL) has drawn considerable attention as a novel distributed learning paradigm in recent years. In FL, users collaborate to train a model using a centralized server, yet all data is stored locally to protect users' privacy. The privacy benefit popularizes FL in a variety of sensitive applications, including Google GBoard, healthcare services, and self-driving cars. However, vanilla FL's privacy guarantee is specious and has been demonstrated to be vulnerable to a range of attacks (Bagdasaryan et al., 2020; Bhagoji et al., 2019; Fang et al., 2019; Nasr et al., 2019; Sun et al., 2020; Luo et al., 2021; Hu et al., 2021).

We consider two mainstream vulnerabilities in FL. First, the centralized server can infer information about the local data of the clients by inspecting their updates (Nasr et al., 2019; Luo et al., 2021; Hu et al., 2021). Second, recent research (Bagdasaryan et al., 2020; Bhagoji et al., 2019; Fang et al., 2019; Sun et al., 2020) has shown that a small number of clients can behave maliciously in a large-scale FL system and influence the jointly-trained FL model in a stealthy manner. Indeed, for the majority of today's SGD-based FL algorithms (McMahan & Ramage, 2017), the centralized server averages the local updates to obtain the global update, which is vulnerable to even a single malicious client. Particularly, a malicious client can craft its update in such a way that it prevents the global model from converging or leads it to a sub-optimal minimum.

Most prior efforts to improve FL security have focused on one of the two vulnerabilities. At one end of the spectrum, secure aggregation techniques (Bonawitz et al., 2017; Bell et al., 2020) are designed to conceal individual client's updates and reveal only the aggregated global update to a semi-honest server that attempts to infer the clients' privacy from their updates. At the other end of the spectrum, Byzantine-robust FL protocols (Blanchard et al., 2017; Yin et al., 2018; Fu et al., 2019; Pillutla et al., 2019) are proposed to suppress the influence of malicious clients' updates.

Orchestrating **robust FL estimators** (to mitigate malicious clients) with **secure aggregation schemes** (to mitigate semi-honest servers) is challenging, as robust estimators require access to local updates, whereas secure aggregation schemes normally hide them from the server. Consequently, most de facto FL protocols cannot protect both the server and the clients *simultaneously*, but must repose total trust in one of them. There are two parallel works (So et al., 2020; He et al., 2020) toward addressing the challenge through the use of secure multi-party computation (MPC). So et al. (2020) propose to run multi-party distance-based filtering among clients to remove potentially malicious updates. This, however, needs clients to be online consistently, which is impractical for cross-device FL and also incurs significant communication cost. He et al. (2020) propose a two-server protocol to provide bi-directional protection, but in real-world use scenarios, two non-colluding servers are uncommon. Due to the lack of universal and effective two-way protection, user trust in FL systems is significantly eroded, preventing them from being employed in a wide variety of security-related applications such as home monitoring and autonomous driving.

In the present work, we propose SHARDFL, a federated learning protocol that provides **bidirectional defense** against both a semi-honest server and Byzantine-malicious clients. The main challenge is to address the incompatibility between robust estimator and secure aggregation. Concretely, robust estimator requires checking individual updates from clients, whereas secure aggregation hides those updates from the semi-honest server. To this end, we propose a practical and highly effective solution to split the clients into *shards*, where SHARDFL securely aggregates each shard's update and launches robust aggregation on updates from different shards.

Furthermore, SHARDFL is the first Byzantine-robust federated mean estimation protocol with **dimension-free error**. To date, most Byzantine-robust aggregation protocols suffer from estimation error proportional to the square root of the dimension of the updates. As a concrete instance, a three-layer MLP on MNIST comprises more than 50,000 parameters and leads to $223\times$ constants in the estimation error, let alone large language models like BERT. Draco (Chen et al., 2018), BULYAN (Mhamdi et al., 2018), and ByzatineSGD (Alistarh et al., 2018) are the only three works that state to provide dimension-free estimation error. However, Draco is incompatible with FL as it requires redundant updates from each worker. Bulyan and ByzantineSGD are based on much stronger assumptions than other contemporary works. As will be discussed in Sec. 2, when the assumptions are relaxed to the common case, Bulyan and ByzantineSGD's estimation errors still scale up with the square root of the model size. SHARDFL overcomes limitations of existing Byzantine-robust FL protocols by employing and calibrating a well-established robust mean estimator FilterL2 (Steinhardt, 2018) in FL scenarios.

We evaluate SHARDFL under five frequently-launched attacks over two datasets and compare SHARDFL with five robust estimators. Evaluation results show that SHARDFL constantly achieves optimal or close-to-optimal performance under all the attacks. We also study how different settings of shards can influence security guarantees. In summary, we make the following contributions:

- We propose SHARDFL, the first FL protocol featuring principled and practical defense *simultaneously* against a semi-honest server and Byzantine malicious clients. We propose the sharding scheme to reconcile dimension-free robust estimators and secure aggregation. We also rigorously prove the robustness and security guarantee of SHARDFL.
- We point out the limitations in existing robust estimators with claimed dimension-free errors. We reuse and calibrate a well-studied robust mean estimation technique, FilterL2, to achieve dimension-free estimation error in FL.
- Our evaluation shows that SHARDFL can notably outperform existing robust estimators in the presence of five popular attacks by always achieving optimal or close-to-optimal performance.

2 RELATED WORK & LIMITATIONS IN EXISTING BYZANTINE-ROBUST PROTOCOLS

We now review the client-side privacy vulnerabilities and Byzantine-malicious attacks in FL. Existing defenses and their limitations are also discussed to motivate the present work. Other attacks and defenses in FL are covered in these comprehensive surveys (Kairouz et al., 2019; Lyu et al., 2020).

Client Privacy Leakage and Mitigation. Inference attacks against centralized learning (Shokri et al., 2017; Fredrikson et al., 2015) aim to infer the private information of the model training data. Wang et al. (2019) explore the feasibility of recovering user privacy from a malicious server in FL. Nasr et al. (2019) show that a malicious server can perform highly accurate membership inference attacks against clients. To protect the privacy of clients, Bonawitz et al. (2017) propose secure aggregation, which provides security guarantee against a semi-honest server. Also, by utilizing secure multi-party computation (MPC), Mohassel & Zhang (2017) present a framework for training a global model on the clients' encrypted data between two non-colluding servers. However, these existing defense mechanisms are based on the assumption of benign clients, which is not often the case in practice.

Byzantine Malicious Clients. Byzantine-robust aggregation has drawn enormous attention over the past few years due to the emergence of various distributed attacks in FL. Fang et al. (2019) formalize the attack as an optimization problem and successfully migrate the data poisoning attack to FL. The proposed attacks even work under Byzantine-robust FL. Sun et al. (2020) manage to launch data poisoning attacks on the multi-task FL framework. Bhagoji et al. (2019) and Bagdasaryan

et al. (2020) manage to insert backdoors into the model via local model poisoning or local model replacement. Xie et al. (2019) propose to segment one backdoor into several parts and insert it into the global model. Chen et al. (2020) and Zizhan et al. (2020) migrate backdoor attacks to federated meta-learning and federated reinforcement learning, respectively. Meanwhile, Sun et al. (2019) show that norm clipping and "weak" differential privacy mitigate backdoor attacks in FL without impairing overall performance. Wang et al. (2020) refute this claim and illustrate that robustness to backdoors requires model robustness to adversarial examples, an open problem widely regarded to be difficult.

Byzantine-Robust Protocols. A variety of Byzantine-robust FL protocols are proposed to defend against malicious clients. Krum (Blanchard et al., 2017) selects and averages the subset of updates that have a sufficient number of close neighbors. Yin et al. (2018) use robust estimators like trimmed mean or median to achieve order-optimal statistical error rates under strongly convex assumptions. Fung et al. (2018) propose a similar robust estimator relying on a robust secure aggregation oracle based on the geometric median. Yin et al. (2019) propose to use robust mean estimators to defend against saddle point attacks. Fung et al. (2020) study Sybil attacks in FL and propose a defense based on the diversity of client updates. Ozdayi et al. (2020) design a defense against backdoor attacks in FL by adjusting the server-side learning rate. Mhamdi et al. (2018) point out that Krum, trimmed mean, and median all suffer from $\mathcal{O}(\sqrt{d})$ (d is the model size) estimation error and propose a general framework Bulyan for reducing the error to $\mathcal{O}(1)$.

Limitations of Existing Robust Estimators. We point out that the improvement of Bulyan (Mhamdi et al., 2018) actually comes from its stronger assumption. In particular, Bulyan assumes that the expectation of the distance between two benign updates is bounded by a constant σ_1, while Krum assumes that the distance is bounded by $\sigma_2\sqrt{d}$. We can easily see that if $\sigma_1 = \sigma_2\sqrt{d}$, Bulyan falls back to the same order of estimation error as Krum. The same loophole exists in the analysis of ByzantineSGD (Alistarh et al., 2018). As a result, no FL protocol is known to be equipped with dimension-free estimation error that can be used to mitigate Byzantine adversaries.

3 Problem Setup

In this section, we review the general pipeline of FL and introduce the threat model and defense goal. We use bold lower-case letters (*e.g.* $\mathbf{a}$,$\mathbf{b}$,$\mathbf{c}$) to denote vectors, and $[n]$ to denote $1 \cdots n$.

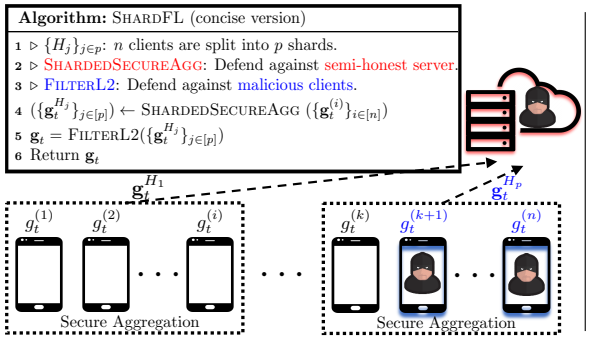
FL Pipeline. In an FL system, there is one server $\mathcal{S}$ and n clients $\mathcal{C}_i, i \in [n]$. Each client holds data samples drawn from some unknown distribution $\mathcal{D}$. Let $\ell(\mathbf{w}; \mathbf{z})$ be the loss function on the model parameter $\mathbf{w} \in \mathbb{R}^d$ and a data sample $\mathbf{z}$. Let $\mathcal{L}(\mathbf{w}) = \mathbb{E}_{\mathbf{z}\sim\mathcal{D}}[\ell(\mathbf{w}; \mathbf{z})]$ be the population loss function. Our goal is to learn the model $\mathbf{w}$ such that the population loss function is minimized: $\mathbf{w}^* = \arg\min_{\mathbf{w}\in\mathcal{W}} \mathcal{L}(\mathbf{w})$. To learn $\mathbf{w}^*$, the whole system runs a T-round FL protocol. Initially, the server stores a global model $\mathbf{w}_0$. In the t^{th} round, $\mathcal{S}$ broadcasts the global model $\mathbf{w}_{t-1}$ to the m clients. The clients then run the local optimizers (e.g., SGD, Adam, RMSprop), compute the difference $\mathbf{g}_t^{(i)}$ between the optimized model and the global model, and upload the difference to $\mathcal{S}$. In the t^{th} round, $\mathcal{S}$ takes the average of the differences and updates the global model $\mathbf{w}_t = \mathbf{w}_{t-1} + \frac{1}{n}\sum_{i=1}^{n} \mathbf{g}_t^{(i)}$.

Threat Model & Defense Goal. We assume that the centralized server $\mathcal{S}$ is semi-honest: the server can launch whatever attacks it wishes, including inference attacks, using only legitimate updates from the clients as inputs. The server, however, cannot deviate from the protocol for regulatory or reputational pressure. This makes a semi-honest server highly stealthy. The assumption on the semi-honest server is consistent with the convention in this line of work (Yang et al., 2019; Bonawitz et al., 2017; Mohassel & Zhang, 2017). Additionally, we assume that clients are ϵ-Byzantine malicious, meaning that at most ϵn clients are malicious: they can deviate arbitrarily from the protocol and tamper with their own updates for profitable or even mischief purposes. We also clarify that there is no collusion between the server and the clients. That is, the server cannot disguise as a client or hire clients to launch colluded attacks.

The defense objective is two-fold. First, we would like to achieve a dimension-free error for the mean estimation in each round. Let $\boldsymbol{\mu}$ be the true mean of the benign distribution, and the output of a protocol with contaminated inputs be $\hat{\boldsymbol{\mu}}$. The estimation error is defined as the ℓ_2 distance between the true mean and the estimation $\|\hat{\boldsymbol{\mu}} - \boldsymbol{\mu}\|_2$. We also would like to minimize the server's ability to infer sensitive information of the clients. Formally speaking, to ensure client-side privacy, we would like to conceal the client's individual update within the aggregate of multiple updates.

4 SHARDFL: ROBUST PRIVACY-PRESERVING FL

The full protocol of SHARDFL will be given Alg. 1, and we first summarize its high-level workflow in Fig. 1(a): SHARDFL provides bidirectional defense, defending against the semi-honest server through secure aggregation (marked in red) and malicious clients with the robust mean estimator FilterL2 (marked in blue). Fig. 1(b) formulates FilterL2, the details of which are presented in Sec. 4.1. We now introduce SHARDFL and formally establish the robustness and security guarantees in Sec. 4.2.



(a) **Illustration of ShardFL.**

Algorithm: FilterL2

Input: $\mathbf{g}_t^{H_1}, \cdots, \mathbf{g}_t^{H_p} \in \mathbb{R}^d$, $\eta > 1$

Let $c_1, \cdots, c_p = 1$

$\hat{\boldsymbol{\mu}}_c = (\sum_{i=1}^p c_i \mathbf{g}_t^{H_i})/(\sum_{i=1}^p c_i)$

$\hat{\boldsymbol{\Sigma}}_c = (\sum_{i=1}^p c_i (\mathbf{g}_t^{H_i} - \hat{\mu}_c)(\mathbf{g}_t^{H_i} - \hat{\mu}_c)^\top)/(\sum_{i=1}^p c_i)$

Let $\mathbf{v}$ be the maximum eigenvector of $\hat{\boldsymbol{\Sigma}}_c$

Let $\hat{\sigma}_c^2 = \mathbf{v}^\top \hat{\boldsymbol{\Sigma}}_c \mathbf{v}$

if $\hat{\sigma}_c^2 \le \eta\sigma^2$ **then**

  Return $\hat{\boldsymbol{\mu}}_c$

else

  Let $\tau_i = \langle \mathbf{g}_t^{H_i} - \hat{\boldsymbol{\mu}}_c \rangle^2$, $\tau_{\max} = \max_i \tau_i$

  Update $c_i \leftarrow c_i \cdot (1 - \tau_i/\tau_{\max})$

Go back to line 2

(b) **FilterL2: dimension-free robust mean estimation.**

Figure 1: High-level overview of SHARDFL and algorithm of FilterL2.

4.1 SHARDFL: BYZANTINE-ROBUST PRIVACY-PRESERVING FL

The complete SHARDFL protocol is presented in Algorithm 1. SHARDFL iteratively executes the following steps: (1) the server broadcasts the global model to the clients; (2) the clients train the global model using their local data; (3) clients within the same shard use a secure aggregation protocol to upload the mean of their updates to the server; (4) the server aggregates the received updates using robust mean estimation; and (5) the server updates the global model with the aggregated global update. We highlight steps (3) and (4) newly proposed in SHARDFL.

Algorithm 1: SHARDFL: Robust Privacy-Preserving Sharded FL.

1 **for** $t \leftarrow [T]$ **do**
2 **Server:**
3 Split n clients into p shards $\{H_j\}_{j\in[p]}$
4 Broadcast $\{H_j\}_{j\in[p]}$ and the global model $\mathbf{w}_{t-1}$ to all the clients
5 **Client:**
6 **foreach** *client* $i \in [n]$ **do**
7 Locate its own shard j and generate random masks $\mathbf{u}_{ik}^{(j)}, k \in H_j/i$
8 **foreach** $k \in H_j/i$ **do** Send $\mathbf{u}_{ik}$ to k
9 Train the local model $\mathbf{w}_t^{(i)}$ using $\mathbf{w}_{t-1}$ as initialization
10 $\mathbf{g}_t^{(i)} =$ QUANTIZE$(\mathbf{w}_t^{(i)} - \mathbf{w}_{t-1}) + \sum_{k\neq i, i\in H_j, k\in H_j} \mathbf{u}_{ik}^{(j)} - \sum_{k\neq i, i\in H_j, k\in H_j} \mathbf{u}_{ki}^{(j)}$
11 Send $\mathbf{g}_t^{(i)}$ to the server
12 **Server:**
13 **foreach** $H_j \in \{H_j\}_{j\in[p]}$ **do** $\mathbf{g}_t^{H_j} = \frac{1}{|H_j|} \sum_{k\in H_j} \mathbf{g}_t^{(k)}$
14 $\mathbf{g}_t =$ FilterL2$(\{\mathbf{g}_t^{H_j}\}_{j\in[p]})$
15 $\mathbf{w}_t = \mathbf{w}_{t-1} + \mathbf{g}_t$

Sharded Secure Aggregation (lines 7–8, 10, 13). Secure aggregation is developed by Bonawitz et al. (2017) to defend a semi-honest server in FL. Secure aggregation allows the server to obtain the sum of the clients' updates but cryptographically conceals individual updates. To begin, each client samples random values and transmits the values to other clients in the same shard (lines 7–8). Notably, secure aggregation and cryptographic protocols demand that client updates be integers while transmitting them via modular operation. Because each client's model update uses float numbers, we first quantize before secure aggregation (line 10). After receiving all values from other clients, each client adds the received values together and subtracts its own to produce a random mask (line 10). Each client blinds its local update with the random mask and sends the blinded update to the server

(line 11). The server then aggregates all blinded updates and obtains the plaintext of the summed update (line 13). All masks cancel out during aggregation, and the server receives the plaintext sum. Secure aggregation ensures client privacy by hiding each client in the aggregation of other clients.

Note that in our threat model, vanilla secure aggregation is insufficient since it provides no protection for the server. Due to the total concealment of the individual updates, the server is unable to identify the malicious clients even after detecting the attack. To address the issue, we, therefore, propose that clients are partitioned into multiple shards (lines 7–8) and secure aggregation be performed within each shard. The size of the shards is a trade-off between server and client protection. The smaller the shard size, the more information the server reveals, making it easier to defend against Byzantine malicious clients and harder to combat the semi-honest server. The trade-off is elaborated in Sec. 4.2.

Robust Mean Estimation (line 14). The fundamental step of Byzantine-robust FL is to estimate the true mean of the benign updates as accurately as possible, even when some malicious clients are present. Averaging, the most widely used aggregator, has been shown to be vulnerable to even a single malicious client. However, existing works in this field (e.g., Krum (Blanchard et al., 2017) and Bulyan (Mhamdi et al., 2018)) suffer from a *dimension-dependent* estimation error. Note that such dimension-dependent error can impose an intolerably high cost even when training a three-layer MLP on MNIST, let alone more complex tasks and models such as VGG16 or ResNet50.

SHARDFL, in particular, incorporates a well-known robust mean estimator: FilterL2 (Steinhardt, 2018). We formulate FilterL2 in Alg. 2. FilterL2 assigns each client's update a weight and iteratively updates the weights until the weights for malicious clients' updates are sufficiently small. FilterL2 provides a dimension-free error rate that is formally presented as follows.

Theorem 1 (Steinhardt (2018))**.** *Let D be the honest dataset and D^* be the contaminated version of D by inserting malicious samples. Suppose that $|D^*| < |D|/(1-\epsilon)$, and further suppose that $\text{MEAN}[D] = \boldsymbol{\mu}$ and $\|\text{COV}[D]\|_{op} \leq \sigma^2$. Then, given $\mathcal{D}^*$, Algorithm 2 outputs $\hat{\boldsymbol{\mu}}$ s.t. $\|\hat{\boldsymbol{\mu}} - \boldsymbol{\mu}\|_2 = \mathcal{O}(\sigma\sqrt{\epsilon})$ using $\text{POLY}(n, d)$ time, where ϵ is the fraction of malicious data and $\epsilon < \frac{1}{2}$.*

Although Algorithm 2 runs in polynomial time, the per-round time complexity is $\mathcal{O}(nd^2)$ if implemented with power iteration. Given d is large, the running time is still quite expensive in practice. To address the problem, we divide the update vectors into k sections and apply the robust estimator to each section. The acceleration scheme reduces the per-round running time to $\mathcal{O}(nd^2/k)$, but increases the estimation error to $\mathcal{O}(\sigma\sqrt{\epsilon k})$. For instance, if we take $k = \sqrt{d}$, the per-round running time becomes $\mathcal{O}(nd\sqrt{d})$, while the estimation error increases to $\mathcal{O}(\sigma\sqrt[4]{\epsilon^2 d})$. Despite the tradeoff for acceleration, FilterL2 maintains the known optimal estimation error and outperforms other robust FL protocols by several orders of magnitude, as shown in Sec. 5.

4.2 ROBUSTNESS & SECURITY ANALYSIS

In this section, we rigorously prove the security and robustness guarantee of SHARDFL.

Security Guarantee. We first give the security guarantee of SHARDFL as follows. Intuitively, the centralized server learns nothing more about the clients than the averaged updates from the shards. Thus, each client's update is hidden by the rest clients in its shard.

Corollary 1 (Security against semi-honest server)**.** *Let Π be an instantiation of* SHARDFL*, there exists a PPT (probabilistic polynomial Turing machine) simulator* SIM *which can only see the averaged updates from the shards. For all clients $\mathcal{C}$, the output of* SIM *is computationally indistinguishable from the view of that real server $\Pi_{\mathcal{C}}$ in that execution, i.e., $\Pi_{\mathcal{C}} \approx \text{SIM}(\{\boldsymbol{g}_t^{H_j}\}_{j\in[p]})$.*

Proof for Corollary 1. The transcript of the server is the updates from the sharded clients $\{\mathbf{g}_t^{H_j}\}_{j\in[p]}$. Hence, Corollary 1 is equivalent to the following lemma since the SIM can split the aggregated updates into several random shards which is computationally indistinguishable from the true transcript.

Lemma 1 (Lemma 6.1 in (Bonawitz et al., 2017))**.** *Given any shard H_k formed by a set of clients C_k, the parameter size d, the group size q, and the updates $\boldsymbol{g}^{(i)}$ where $\forall i \in C_k$, $\boldsymbol{g}^{(i)} \in \mathbb{Z}_q^d$, we have*

$$\{\{\boldsymbol{u}_{ij} \xleftarrow{\$} \mathbb{Z}_q^d\}_{i<j}, \boldsymbol{u}_{ij} \coloneqq -\boldsymbol{u}_{ji}\ \forall\, i,j \in C_k,\ i>j \quad : \{\boldsymbol{g}^{(i)} + \sum_{j\in C_k/i} \boldsymbol{u}_{ij} \pmod q\}_{i\in C_k}\}$$

$$\equiv \{\{\boldsymbol{v}_i \xleftarrow{\$} \mathbb{Z}_q^d\}_{i\in C_k}\ s.t.\ \sum_{i\in C_k} \boldsymbol{v}_i = \sum_{i\in C_k} \boldsymbol{g}^{(i)} \pmod q \quad : \{\boldsymbol{v}_i\}_{i\in C_k}\}$$

, where $\boldsymbol{u}_{ij}$ is the random mask shared between client i and j, $\xleftarrow{\$}$ donates uniformly sampling from some field, and $\equiv$ denotes that the distributions are identical.

Lemma 1 illustrates that the distribution of updates with random masks added is identical to uniformly sampling from $\mathbb{Z}_q^d$. Thus, individual clients' updates are securely concealed behind the random masks added by secure aggregation, and a semi-honest server can infer zero information about individual clients from the aggregated updates alone. In the following, we prove Lemma 1 via induction on n, where n is the size of the clients set C_k, $n = |C_k|$.

Base Case: When $n = 2$, assume $C_k = \{i, j\}$, $i < j$, and $\sum_{i \in C_k} \mathbf{g}^{(i)} \pmod q = c$, c is a constant. The first elements of two distributions are $\mathbf{g}^{(i)} + \mathbf{u}_{ij} \pmod q$ and $\mathbf{v}_i$, respectively, both of which are uniformly random sampled from $\mathbb{Z}_q^d$. The second elements are $\mathbf{g}^{(j)} + \mathbf{u}_{ji} \pmod q = c - (\mathbf{g}^{(i)} + \mathbf{u}_{ij}) \pmod q$ and $\mathbf{v}_j = c - \mathbf{v}_i \pmod q$, respectively, which are the sum c minus the corresponding first elements. As a result, the distributions are identical.

Inductive Hypothesis: When $n = k$, the lemma holds.

Inductive Step: According to the inductive hypothesis, the left and right distributions of the first k clients are indistinguishable. We follow the protocol to generate the left transcript when the $(k+1)^{th}$ client is added to the shard. To deal with the right-hand-side transcript, we first add the same randomness as for the left-hand-side to the first k updates and then subtract them from the total sum to obtain the $(k+1)^{th}$ update. It's easy to prove that the first k updates on both sides follow the same uniformly random distribution, and that the $(k+1)^{th}$ update is the difference between the total sum and the sum of the first k updates. Hence, the left and right transcripts are indistinguishable.

□

In case the readers are not familiar with the simulation proof technique, please refer to Lindell (2017) for more information about simulation-based security proof.

Robustness Guarantee. We now give the formal robustness guarantee of SHARDFL. The proof involves a trivial application of Theorem 1 so we omit it here.

Corollary 2 (Robustness against Byzantine adversaries)**.** *Given the number of clients n, the number of shards p, and the fraction of corrupted clients η,* SHARDFL *provides a mean estimation with dimension-free error as long as $\eta n < \epsilon p$, where $\epsilon < \frac{1}{2}$ is the tolerance of FilterL2 given in Theorem 1.*

Proof for Corollary 2. In the following analysis, we assume that the updates from the shards follow an i.i.d. distribution.

The fraction of malicious shards is bounded by the worst case where each malicious client is exclusively assigned to different shards: $\epsilon' \leq \frac{\eta n}{p} \leq \epsilon$, $\epsilon < \frac{1}{2}$. Given the assumption above, we have satisfied all the requirements in Theorem 1. Hence, SHARDFL provides a mean estimation with dimension-free error as long as $\eta n < \epsilon p$. □

Remark. Given the formal security and robustness guarantees, it is clear that SHARDFL enables easy calibration of the server's or clients' protection. SHARDFL, in particular, can tolerate up to $\lfloor \epsilon p \rfloor$ malicious clients and conceal each honest client's update in the mean of $\lfloor \frac{n}{p} \rfloor$ updates. As illustrated in Fig. 1, SHARDFL employs a novel sharding scheme to overcome the incompatibility between a robust estimator and secure aggregation. The value of p, denoting the number of shards, primarily affects the robustness and security provided by SHARDFL. If the number of benign shards is less than the number of malicious shards (i.e., $p \leq 2\eta n$), FilterL2's requirement on the fraction of malicious shards $\epsilon < \frac{1}{2}$ may not hold. That is, SHARDFL may not exclude malicious shards if p is insufficiently large. However, if p is too large, implying that the number of clients in each shard is small, the semi-honest server is more likely to infer the clients' private updates from the aggregations of the shards. In practice, p should be determined by the total number of clients, the desired level of security and robustness, and the nature of the FL task. See evaluation in Fig. 4 for various numbers of shards.

Attentive readers might notice that the robustness tolerance is lower than when there is no sharding. We argue that it is an almost unavoidable cost because if we can have the same robustness tolerance as the no-sharding case, then we can design a robust estimator for partially aggregated samples with the same tolerance. As the latter problem is unsolved, we deem it extremely challenging to further increase the tolerance in our scenario.

5 EVALUATION

This section empirically explores the following questions: (Q_1) Does FilterL2 outperform other aggregators when used alone? (Q_2) Does SHARDFL outperform other robust FL protocols augmented with sharded secure aggregation? (Q_3) How does the shard size affect the performance of SHARDFL?

Attacks. The first and second attacks are the model poisoning attacks (Fang et al., 2019). Model poisoning attacks aim to raise the error rate of the converged model even facing Byzantine-robust protocols. In these attacks, malicious clients search for poisoning updates by solving optimization problems. We employ two of their proposed attacks targeting Krum and Trimmed Mean. These two attacks are referred to as Krum Attack (KA) and Trimmed Mean Attack (TMA).

The third attack we consider is a backdoor attack from (Bhagoji et al., 2019). The attack aims to inject a backdoor functionality while preserving high accuracy on the validation set. Similarly, the search for the attack gradient is formalized as an optimization problem, and the authors modify the objective function using stealth metrics to make the attack gradient hard to detect. We refer to this attack as Model Poisoning Attack (MPA).

The fourth attack is also a backdoor attack proposed by (Bagdasaryan et al., 2020). In this attack, the adversary locally trains a model using a backdoor and then attempts to replace the global model with the local model by uploading the difference between the target and global models. We refer to the attack as Model Replacement Attack (MRA).

The fifth attack is a distributed backdoor attack (DBA) (Xie et al., 2019), in which the attacker manipulates updates of several clients to insert a backdoor into the global model collaboratively.

Clarification on Targeted Attacks for FilterL2. Attentive readers might notice that the first and second attacks are designed specifically for Krum and TrimmedMean, and wonder whether it is possible to construct an attack particularly for FilterL2. We argue that designing such an attack using the same idea from Fang et al. (2019) is not possible, as the optimization problem becomes intractable when FilterL2 is plugged in. Due to the theoretically stronger robustness, we assume that it is very challenging to design targeted attacks for FilterL2 like Krum or Trimmed Mean. We leave it as an important future direction to design such an attack or rigorously prove its impossibility.

Experimental Setup. We selected two datasets, MNIST (LeCun et al., 2010) and FashionMNIST (Xiao et al., 2017), to evaluate SHARDFL. We also chose three other Byzantine-robust FL protocols as baselines: (1) Krum (Blanchard et al., 2017); (2) Trimmed Mean (Yin et al., 2018); and (3) Bulyan (Mhamdi et al., 2018). Note that Bulyan acts as a wrapper for other robust estimators. Therefore, in the evaluation, we have two versions of Bulyan: Bulyan Krum and Bulyan Trimmed Mean. We run all the protocols on the two datasets and present the protocols' performance under different attacks. Performance is measured differently according to different attack targets. For KA and TMA, we use model accuracy to quantify attack performance. Increased model accuracy indicates increased robustness. For MPA, MRA, and DBA, we assess the percentage of the remembered backdoors to demonstrate the attack performance. The fewer backdoors remembered, the more robust the estimator is. Please refer to Appendix A for details on model architecture, quantization, and hyper-parameters.

All the experiments were conducted on a Ubuntu16.04 LTS server with eight Geforce GTX 1080 Ti. Please refer to the codebase in the supplementary material for further implementation details.

5.1 EVALUATION RESULTS

In this section, we present the evaluation results. We first show that FilterL2 outperforms other robust aggregators when used alone. Then, we run complete SHARDFL with sharding, and the results show that SHARDFL consistently achieves optimal or close-to-optimal performance under different attacks. Last, we discuss the effect of the shard size on performance and privacy.

FilterL2 Performance without Sharding. To answer question Q_1, we evaluated six aggregators using MNIST and FashionMNIST as shown in Fig. 2. We ran the protocols with 20 clients, and 5 of them were malicious under attacks.

Fig. 2 reports promising results that FilterL2 achieves optimal performance among all 6 aggregators. For MNIST, under KA and TMA, FilterL2 separately achieves 96.85% and 96.15% accuracy within

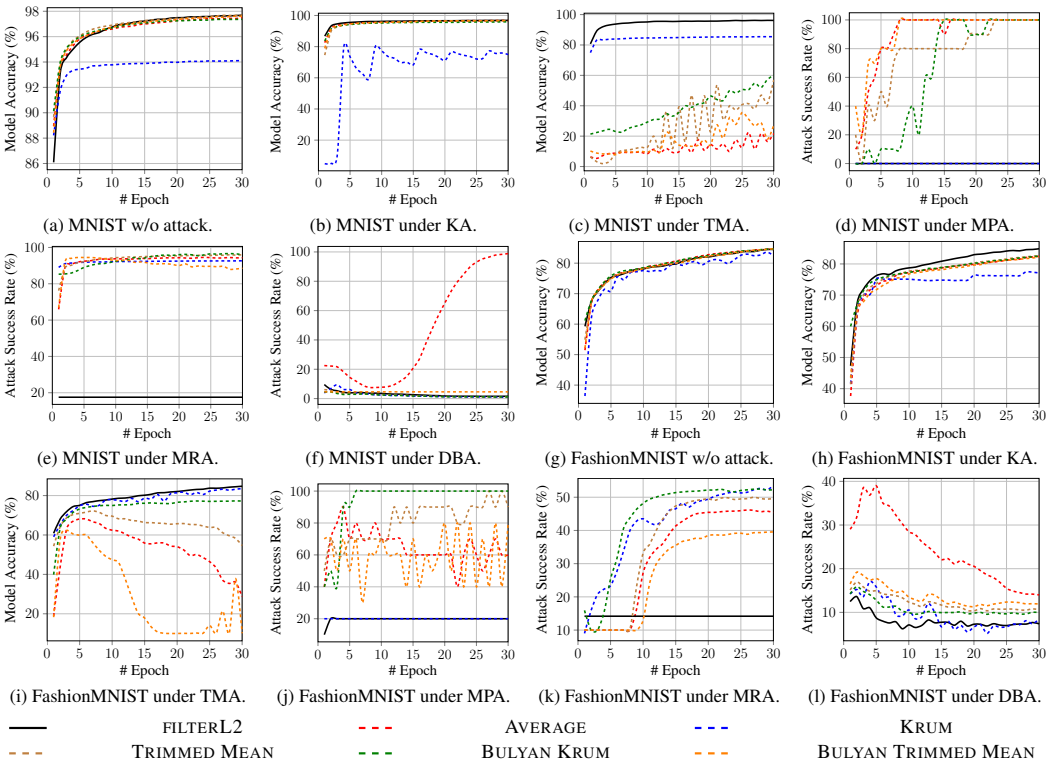


Figure 2: Attack performance under various Byzantine-robust estimators.

30 epochs, respectively, comparable to the non-malicious setting's 97.48% accuracy. Under MPA, MRA, and DBA, FilterL2 separately reduces the attack success rate to 0.00%, 17.53%, and 1.62%. For Fashion-MNIST, under KA and TMA, FilterL2 separately achieves 84.87% and 84.75% accuracy, slightly better than the non-malicious setting with an accuracy of 84.66%. Under MPA, MRA, and DBA, FilterL2 successfully suppresses the attack success rates to 20.00%, 14.18%, and 7.38%. Specifically, FilterL2 is the only aggregator that consistently performs well under all five attacks.

SHARDFL Performance. To answer question Q_2, we evaluated six aggregators with sharding on MNIST and FashionMNIST as shown in Fig. 3. We ran the protocols with 100 clients, and 10 of them were malicious under attacks. The clients were randomly split into 25 shards. And the clients inside the same shard performed secure aggregation.

For the experiments without attack, with TMA, or with MPA (Fig. 3a,3c,3d,3g.3i,3j), SHARDFL still achieves optimal or near-optimal performance. Specifically, SHARDFL achieves 97.19% and 95.79% accuracy under KA and TMA on MNIST, whereas averaging with sharding achieves 97.28% without attack. It also suppresses the attack success rates to 10.00%, 10.15%, and 23.32%, respectively, under MPA, MRA, and DBA. On FashionMNIST, SHARDFL achieves 85.66% and 85.82% under KA and TMA, respectively, whereas the baseline is 86.59% without attack. Under MPA, MRA, and DBA, the attack success rates are kept below 10.00%, 11.59%, and 2.91% at the end of training.

Influence of Shard Size. To answer question Q_3, we evaluate the influence of shards number p in Fig. 4, where we launch the TMA attack against SHARDFL with different ps. When the number of shards equals the number of clients, the system is trivially equivalent to FilterL2 without sharding and is capable of achieving optimal model accuracy. This setting, however, compromises security, as the semi-honest server has access to each client's individual update, and SHARDFL offers no further security guarantee compared with vanilla FL. Contrarily, when the number of shards reaches one, the system degrades to averaging each client's update. This extreme setup delivers the highest security but the lowest robustness. When the amount

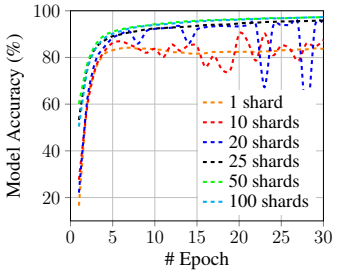


Figure 4: MNIST under TMA with various number of shards.

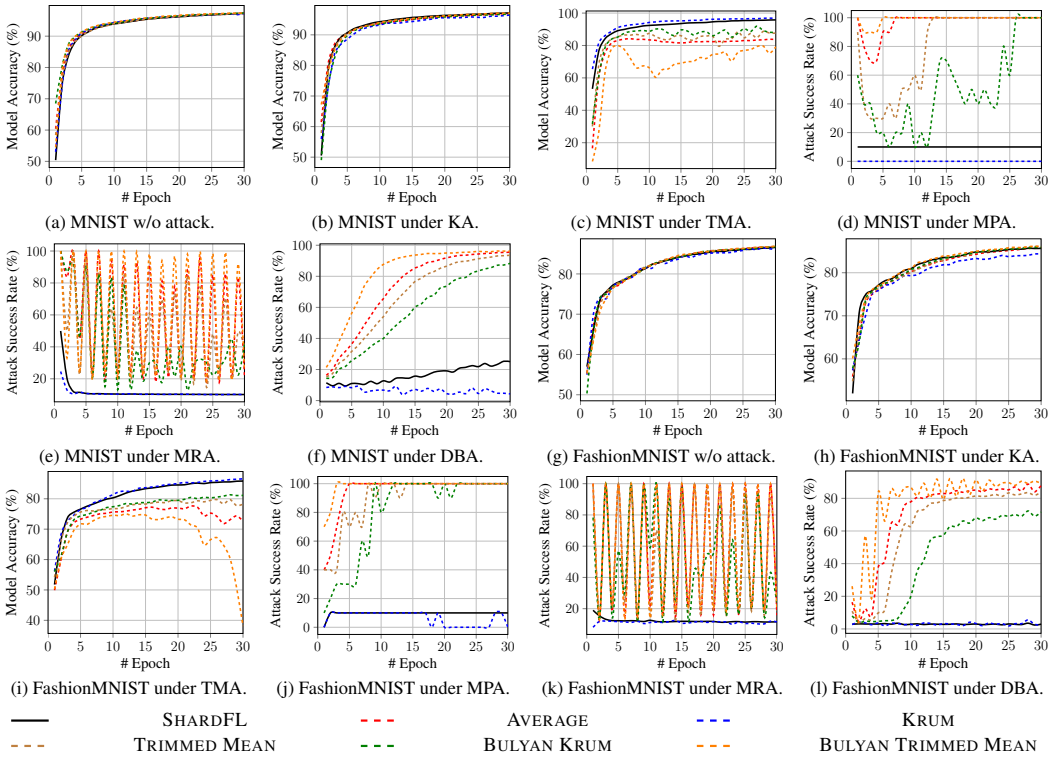


Figure 3: Attack performance under various estimators with sharded secure aggregation.

of shards falls between these two extremes, the model accuracy gradually changes under the TMA attack, as in Fig. 4. Our empirical finding indicates that $p = 25$ would be desirable for datasets like MNIST and FashionMNIST when 10 out of 100 clients are malicious. Holistically, the optimal choice of p should depend on the total number of clients, the required level of security and robustness, and also the specific FL task.

6 LIMITATION & CONCLUSION

In this paper, we design and develop SHARDFL, the first robust FL protocol with a dimension-free per-round mean estimation error. SHARDFL provides bi-directional security guarantees against both a semi-honest server and Byzantine malicious clients. We propose to use FilterL2 to robustly aggregate the possibly contaminated updates and to use secure aggregation to protect the client's privacy. We reconcile the contradictory components with sharding. The evaluation results show that SHARDFL consistently achieves the optimal or close-to-optimal performance among five robust FL protocols. As far as we can see, SHARDFL tackles the two primary privacy concerns in FL systems simultaneously and shows the potential to further popularize FL in sensitive applications.

We also identify several unsolved challenges in SHARDFL that may motivate future works in bidirectionally protected FL. For instance, vanilla FilterL2 incurs extra overhead due to its nearly-quadratic complexity. Although the accelerated FilterL2 partially tackles the issue, it does so at the expense of the asymptotic estimation error. A promising future direction is to integrate low-complexity robust mean estimators such as Cheng et al. (2019). However, because Cheng et al. (2019)'s approach is rather complicated, developing a low-complexity robust mean estimator with simple intuition is also an intriguing direction. Besides, the number of shards p should be selected carefully, as discussed in Sec. 4.2. Large p weakens the security guarantee of SHARDFL while small p degrades its robustness. As shown in our evaluation, $p = 25$ would be a good choice for the datasets like MNIST and FashionMNIST with 10 out of 100 malicious clients. However, in practice, it should depend on the total number of clients, the required level of security and robustness, and the specific FL task. Another promising future direction is to design a secure FL protocol with higher robustness threshold. The idea in Diakonikolas et al. (2018) can potentially shed some light on this direction.

REFERENCES

Dan Alistarh, Zeyuan Allen-Zhu, and Jerry Li. Byzantine stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pp. 4613–4623, 2018.

Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 2938–2948, 2020.

James Henry Bell, Kallista A Bonawitz, Adrià Gascón, Tancrède Lepoint, and Mariana Raykova. Secure single-server aggregation with (poly) logarithmic overhead. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1253–1269, 2020.

Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens. In *International Conference on Machine Learning*, pp. 634–643, 2019.

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

Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1175–1191, 2017.

Chien-Lun Chen, Leana Golubchik, and Marco Paolieri. Backdoor attacks on federated meta-learning. *arXiv preprint arXiv:2006.07026*, 2020.

Lingjiao Chen, Hongyi Wang, Zachary Charles, and Dimitris Papailiopoulos. Draco: Byzantine-resilient distributed training via redundant gradients. *arXiv preprint arXiv:1803.09877*, 2018.

Yu Cheng, Ilias Diakonikolas, and Rong Ge. High-dimensional robust mean estimation in nearly-linear time. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 2755–2771. SIAM, 2019.

Ilias Diakonikolas, Daniel M Kane, and Alistair Stewart. List-decodable robust mean estimation and learning mixtures of spherical gaussians. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pp. 1047–1060, 2018.

Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Local model poisoning attacks to byzantine-robust federated learning. *arXiv preprint arXiv:1911.11815*, 2019.

Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 1322–1333, 2015.

Shuhao Fu, Chulin Xie, Bo Li, and Qifeng Chen. Attack-resistant federated learning with residual-based reweighting. *arXiv preprint arXiv:1912.11464*, 2019.

Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. Mitigating sybils in federated learning poisoning. *arXiv preprint arXiv:1808.04866*, 2018.

Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. The limitations of federated learning in sybil settings. In *23rd International Symposium on Research in Attacks, Intrusions and Defenses ({RAID} 2020)*, pp. 301–316, 2020.

Lie He, Sai Praneeth Karimireddy, and Martin Jaggi. Secure byzantine-robust machine learning. *arXiv preprint arXiv:2006.04747*, 2020.

Hongsheng Hu, Zoran Salcic, Lichao Sun, Gillian Dobbie, and Xuyun Zhang. Source inference attacks in federated learning. *arXiv preprint arXiv:2109.05659*, 2021.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

Yehuda Lindell. *How to Simulate It – A Tutorial on the Simulation Proof Technique*, pp. 277–346. Springer International Publishing, Cham, 2017. ISBN 978-3-319-57048-8. doi: 10.1007/978-3-319-57048-8_6. URL https://doi.org/10.1007/978-3-319-57048-8_6.

Xinjian Luo, Yuncheng Wu, Xiaokui Xiao, and Beng Chin Ooi. Feature inference attack on model predictions in vertical federated learning. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pp. 181–192. IEEE, 2021.

Lingjuan Lyu, Han Yu, and Qiang Yang. Threats to federated learning: A survey. *arXiv preprint arXiv:2003.02133*, 2020.

Brendan McMahan and Daniel Ramage. Federated learning: Collaborative machine learning without centralized training data. *Google Research Blog*, 3, 2017.

El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Rouault. The hidden vulnerability of distributed learning in byzantium. *arXiv preprint arXiv:1802.07927*, 2018.

Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 19–38. IEEE, 2017.

Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE symposium on security and privacy (SP)*, pp. 739–753. IEEE, 2019.

Mustafa Safa Ozdayi, Murat Kantarcioglu, and Yulia R Gel. Preventing backdoors in federated learning by adjusting server-side learning rate. 2020.

Krishna Pillutla, Sham M Kakade, and Zaid Harchaoui. Robust aggregation for federated learning. *arXiv preprint arXiv:1912.13445*, 2019.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 3–18. IEEE, 2017.

Jinhyun So, Başak Güler, and A Salman Avestimehr. Byzantine-resilient secure federated learning. *IEEE Journal on Selected Areas in Communications*, 2020.

Jacob Steinhardt. *Robust learning: Information theory and algorithms*. PhD thesis, Stanford University, 2018.

Gan Sun, Yang Cong, Jiahua Dong, Qiang Wang, and Ji Liu. Data poisoning attacks on federated machine learning. *arXiv preprint arXiv:2004.10020*, 2020.

Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H Brendan McMahan. Can you really backdoor federated learning? *arXiv preprint arXiv:1911.07963*, 2019.

Hongyi Wang, Kartik Sreenivasan, Shashank Rajput, Harit Vishwakarma, Saurabh Agarwal, Jy-yong Sohn, Kangwook Lee, and Dimitris Papailiopoulos. Attack of the tails: Yes, you really can backdoor federated learning. *arXiv preprint arXiv:2007.05084*, 2020.

Zhibo Wang, Mengkai Song, Zhifei Zhang, Yang Song, Qian Wang, and Hairong Qi. Beyond inferring class representatives: User-level privacy leakage from federated learning. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pp. 2512–2520. IEEE, 2019.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. Dba: Distributed backdoor attacks against federated learning. In *International Conference on Learning Representations*, 2019.

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

Dong Yin, Yudong Chen, Kannan Ramchandran, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. *arXiv preprint arXiv:1803.01498*, 2018.

Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. Defending against saddle point attack in byzantine-robust distributed learning. In *International Conference on Machine Learning*, pp. 7074–7084. PMLR, 2019.

Zheng Zizhan, Shen Wen, and Henger Li. Learning to attack distributionally robust federated learning. 2020.

APPENDIX

A DETAILS OF EVALUATION SETUP

This section reports detailed information regarding model architecture, hyper-parameters, quantization, and datasets processing procedures.

The model without attack and under KA, TMA, MPA, and MRA attacks is constructed by two convolutional layers following two fully connected layers with ReLU as the activation function. However, the DBA attack cannot be launched successfully in simple models, as simple models are prone to overfit the injected backdoors. As a result, we use the default model architecture which is ResNet18 evaluated in the DBA paper Xie et al. (2019). For DBA, we use the default parameters used in the original paper Xie et al. (2019) and use the multiple-shot attack approach proposed in that paper to evaluate the attack success rate for all estimators. As for the other experiments, we use a learning rate $1e-3$ and a batch size 10. We configure the same initial model state. We run 30 epochs for these experiments. For estimator Trimmed Mean, we set the threshold as 30%, which means that it can rule out 30% of out-of-distribution updates. For Bulyan that can tolerate f Byzantine workers, we set $f = 4$ to satisfy the assumption $n \geq 4f + 3$ required by Bulyan, where n is the number of clients. For FilterL2, we set $\sigma = 1e - 6$ and $\eta = 20$. All other settings like random seeds are identical for all attack methods, estimators, and training process to ensure a comparison fair.

The datasets we evaluate are MNIST and FashionMNIST, and we use both training and testing splits provided by PyTorch. The data is randomly distributed to clients and each client holds the same number of data samples. We emphasize that for each experiment, we use the same random seed to split or distribute the dataset for fair comparison. In the experiment evaluating the performance of FilterL2, we setup 20 clients, and 5 of them are malicious under attacks. In the experiment evaluating the performance of SHARDFL, we setup 100 clients, and 10 of them are malicious and these clients are randomly splitted into 25 shards.

As for the quantization, we first set fixed minimum and maximum updates values for clients (e.g., r_1 and r_2), then the clients updates are clipped by the minimum and maximum values r_1 and r_2. Then the values in the range of r_1 and r_2 are discretized into L points. The updates of the clients are quantized by mapping them to the L points. For attack DBA and MNIST, we set $r_1 = -1.0$, $r_2 = 1.0$, and $L = 1,000,000$. For attack DBA and FashionMNIST, we set $r_1 = -1.5$, $r_2 = 1.5$, and $L = 1,000,000$. For other attacks and MNIST, we set $r_1 = -0.05$, $r_2 = 0.05$, and $L = 100,000$. For other attacks and FashionMNIST, we set $r_1 = -0.5$, $r_2 = 0.5$, and $L = 100,000$.