

FLShield: A Validation Based Federated Learning Framework to Defend Against Poisoning Attacks

Ehsanul Kabir, Zeyu Song, Md Rafi Ur Rashid and Shagufta Mehnaz
Penn State University

Email: {ekabir, zzs5287, mur5028, smehnaz}@psu.edu

Abstract—Federated learning (FL) is revolutionizing how we learn from data. With its growing popularity, it is now being used in many safety-critical domains such as autonomous vehicles and healthcare. Since thousands of participants can contribute in this collaborative setting, it is, however, challenging to ensure security and reliability of such systems. This highlights the need to design FL systems that are secure and robust against malicious participants’ actions while also ensuring high utility, privacy of local data, and efficiency. In this paper, we propose a novel FL framework dubbed as **FLShield** that utilizes benign data from FL participants to validate the local models before taking them into account for generating the global model. This is in stark contrast with existing defenses relying on server’s access to clean datasets—an assumption often impractical in real-life scenarios and conflicting with the fundamentals of FL. We conduct extensive experiments to evaluate our **FLShield** framework in different settings and demonstrate its effectiveness in thwarting various types of poisoning and backdoor attacks including a defense-aware one. **FLShield** also preserves privacy of local data against gradient inversion attacks.

1. Introduction

Federated learning (FL) [8], [26] is a collaborative learning technique that allows multiple participants to jointly train a machine learning (ML) model without sharing their own training data. Each participant trains their local model and shares their local updates with the server which then aggregates the updates to generate a global model and sends it back to the participants. This technique has two important features. Firstly, it can produce a model that is trained on a vast quantity of data provided by thousands of participants without compromising their data privacy. Secondly, decentralized training allows parallelization of the computation across multiple devices resulting in a significant speed boost in the training process. Due to these attractive features, FL technique has been adopted to solve problems in many diverse domains such as autonomous vehicles [35], [36], industry 4.0 [18], [37], healthcare sectors [9], [39], etc.

The growing adoption of FL systems necessitates a comprehensive investigation of their robustness and security. Security concerns arise from the involvement of numerous clients including potential adversaries aiming to disrupt

the learning process. Designing an efficient and effective vetting scheme for such a large number of participants is challenging. Malicious clients may attack FL systems by sending local model updates trained on poisoned data [2], [3], [45] or by intentionally poisoning local models [14], [42].

Recently, two categories of attacks have garnered significant attention: *poisoning* attacks that aim to corrupt the global model and *inference* attacks targeting the theft of local participant data. Poisoning attacks are executed by malicious participants seeking to compromise the global model by sending malicious updates. Attackers may aim to produce a global model with poor performance on the primary task (untargeted poisoning attack) [14], [54] or one that performs poorly on a specific class (i.e., targeted poisoning attack) [45]. Another possible goal is to create a global model that behaves according to the attacker’s intentions when presented with a designed trigger (e.g., backdoor attacks) [2], [44]. A common defense strategy against poisoning attacks involves robust aggregation protocols capable of detecting and filtering malicious updates. However, this detection and filtering strategy relies on the assumption that malicious updates can be distinguished from benign ones—an assumption that may not hold in cases where participant data distribution varies significantly. As benign updates also diverge, separating malicious updates becomes more challenging. Moreover, a malicious participant might adapt their local training procedure to make their updates indistinguishable from benign ones. Therefore, defending against poisoning attacks necessitates a strategy that can validate local models to effectively detect and filter malicious updates. Conversely, inference attacks can be initiated by inverting gradients to reconstruct client’s training data from individual model updates [16].

Challenges: Although some existing works [10] assume server’s access to clean data for local model validation, such assumptions are often impractical—especially in privacy sensitive FL applications. Hence, implementing a validation strategy introduces two challenges: how to obtain the validation data and how to perform validation without exposing the local models. The validation data should be representative of the participants’ training data. In practice, any dataset collected by the server may not sufficiently represent the diverse distribution of clients’ data. One naive solution is selecting a set of clients as validators from the pool of all

clients and using their data for validation of local models. However, sending local models to validators risks exposing them to gradient inversion attacks [16], [56]. Another alternate approach is to validate the global model instead and accepting or rejecting the global model based on the validation performance [1]. However, the averaged global model could be infected at each round and fail the validation test resulting in a denial-of-service. Hence, neither local nor global models are ideal for validation. We name this challenge as *validation subject dilemma*. Additionally, using unvetted FL participants as validators raises concerns as malicious validators can send false results, a challenge we refer to as the *validation integrity dilemma*. These dilemmas present significant obstacles in employing validation as a defense strategy against poisoning attacks.

Proposed defense: In this paper, we propose a novel validation-based defense framework against poisoning attacks, FLShield, that is able to simultaneously solve the validation subject and validation integrity dilemmas in different FL setups. The proposed strategy solves the validation subject dilemma by using *representative models*, which are crafted using multiple local models to ensure that the inference attack will be ineffective. Each representative model is then validated, and based on the validation results, the local models that contributed to the best representative models are selected. We propose two versions of FLShield based on two representative model crafting algorithms: one with the use of *clustering* and the other with a scheme that we refer to as *bijective* model generation. The version that uses clustering-based algorithm is referred to as FLShield* and the version that uses bijective representatives is referred to as FLShield[†]. Our framework also suggests which strategy to use based on the number of participants and the level of robustness required in the system.

We solve the validation integrity dilemma by using a validation protocol that uses a new metric that we refer to as loss impact per class (LIPC). This metric can be applied to any classifier validation process in an FL system. We rank the representative models based on the minimum value extracted from the LIPC of each model and show that the top 50% ranked models are largely crafted using benign updates. Additionally, we demonstrate that the LIPC scores evaluated by the benign validators on representative models are consistent even in a non-IID scenario. We use a validation filtering mechanism in FLShield to filter tailored validation results by malicious validators and solve the validation integrity dilemma.

Evaluating FLShield against diverse attacks: We further show that the filtering technique of FLShield enables it to defend against both untargeted and targeted poisoning attacks. Different poisoning techniques, such as data poisoning or model poisoning, are equally ineffective because of the framework’s focus on validating models before aggregating them. Through experiments on datasets across multiple domains in IID and non-IID scenarios, we show that FLShield can defend against poisoning attacks of three categories including a defense-aware one: untargeted poisoning, targeted label flipping, and backdoor

attacks.

Summary of contributions: In summary, this paper makes the following contributions:

- We develop a novel FL poisoning defense framework, FLShield, that employs an effective validation strategy to solve both the validation subject dilemma and validation integrity dilemma.
- We design a new metric named LIPC for validation purpose. We show that the LIPC metric is a reliable metric to perform validation.
- Through extensive experimentation, we show that our framework can defend against poisoning attacks of three categories: untargeted poisoning, targeted label flipping, and backdoor attacks without compromising the model performance. Moreover, our framework outperforms all existing defenses in terms of robustness and performance.
- We design FLShield-aware attacks and show that FLShield is robust against those.
- Finally, we show that the representative models shared for the purpose of validation can not be used to reconstruct training data of clients using state-of-the-art gradient inversion attacks.

2. Preliminaries

2.1. Federated Learning

Federated learning (FL) is a distributed paradigm enabling a set of clients S to learn a shared global model, coordinated by a central server. Unlike conventional ML frameworks requiring centralized data, FL permits local model training without data sharing. In a training round $t \in [1, T]$, the server sends the global model G_t to a randomly sampled $S_t \subset S$ of size n . Clients $k \in S_t$ fine-tune G_t using local data D_k and send updates ω_k^t to the server which then aggregates them to update the global model G_{t+1} . FL minimizes this objective function: $\min_{\omega} \sum_{k \in S} p_k F_k(\omega)$ where F_k represents the local objective for client k , ω represents the model parameters and p_k denotes their relative impact/weight.

2.2. FL Poisoning Attacks

2.2.1. Untargeted Poisoning Attack. Malicious FL participants may submit tampered updates to indiscriminately poison the global model and reduce the main accuracy (MA) of the FL task. This attack aims to minimize:

$$\text{MA} = \mathbb{E}_{(x,y) \sim \mathcal{D}} [\text{Pr}(G(x) = y)] \quad (1)$$

where \mathcal{D} is the data distribution for the learning task, and y is the expected output for input x .

2.2.2. Targeted Poisoning Attack. Targeted poisoning attacks aim to impair global model performance on specific samples, while maintaining high accuracy elsewhere, making detection challenging. For these attacks, the adversary

aims to maximize misclassification rate (MR) on the targeted subset of data:

$$\text{MR} = \mathbb{E}_{\substack{(x,y) \sim \mathcal{D}_t \\ \mathcal{D}_t \subset \mathcal{D}}} [\Pr(G(x) \neq y)]$$

where \mathcal{D}_t are the samples targeted by the adversary.

Targeted Label Flipping Attack. This is a subclass of targeted poisoning attacks where the adversary manipulates the global model to misclassify samples from a specific class (i.e., source class) to a predetermined poison class (i.e., target class) [45]. For these attacks, the adversary aims at two metrics: minimizing global model’s recall (RCL) on source class and maximizing the attack success rate (ASR) which measures successful misclassification to the target class:

$$\begin{aligned} \text{RCL} &= \mathbb{E}_{\substack{(x,y) \sim \mathcal{D} \\ y=y_s}} [\Pr(G(x) = y)] \\ \text{ASR} &= \mathbb{E}_{\substack{(x,y) \sim \mathcal{D} \\ y=y_t}} [\Pr(G(x) = y_t)] \end{aligned} \quad (2)$$

where y_s is the source class and y_t is the target class.

Backdoor Attack. In backdoor attacks, the global model learns a malicious sub-task alongside the original task, thereby misclassifying the inputs containing some attacker-chosen patterns. The attack aims to maximize the accuracy of this malicious sub-task also termed as backdoor accuracy (BA):

$$\text{BA} = \mathbb{E}_{\substack{(x,y) \sim \mathcal{D} \\ y \neq y'}} [\Pr(G(t(x)) = y')] \quad (3)$$

Here, $t(\cdot)$ is the backdoor trigger function and the pre-determined class y' is the target class.

2.3. FL Privacy Attacks

While FL strives to protect privacy by keeping data at local devices, it still remains possible for attackers to launch privacy attacks if they gain access to the gradients [16], [50], [56]. These attacks are generally formulated as optimization problems and try to reconstruct x_r that approximates the original instance x .

3. Threat Model

Nature of the adversary: We assume that the adversary compromises a subset of the clients through Sybil attacks by creating multiple fake clients to gain control over the FL system. The Sybils have data sampled from the same distribution as the benign clients and they are coordinated by the adversary. Each Sybil leaves as soon as it performs the attack, and then a new Sybil is created. We assume that the FL coordinating server is honest. FLShield is capable of preserving privacy against an honest-but-curious server through the use of secure two-party computation which we present in Appendix D.

Goal of the adversary: The adversary has the following objectives: (1) poison the global model (poisoning objective)

and (2) reconstruct benign clients’ training data (inference objective). To achieve the poisoning objective, the adversary launches a targeted/untargeted poisoning attack, and the success is measured using metrics ASR/BA (targeted) or MA (untargeted). The adversary aims to reconstruct the clients’ training data by launching a gradient inversion attack on model updates. Generally, the adversary has access to only global model updates. However, according to the design of FLShield, when Sybils are selected as validators in a round, the adversary also gets access to the representative model updates. Hence, in the threat model, we assume that the adversary tries to invert representative model updates.

4. Challenges and Key Insights of FLShield

4.1. Challenges

In the ML domain, validation is the process of verifying a model’s potential efficacy in the test phase by evaluating its performance on a validation dataset. In centralized learning, this set is typically a portion of the training dataset not utilized for training. Contrarily, in the context of FL, the participating clients are the data sources. Hence, a naive approach in FL could be using a portion of clients’ data for validation. However, privacy concerns that led the invention of FL prevent the server from directly accessing clients’ data. Although some existing works [10] assume server’s access to clean data, such assumptions are often impractical—especially in privacy sensitive FL applications. Thus, the server must resort to the clients for validation of models. Now, resorting to clients could be insecure as malicious clients can submit falsified validation results to the server and disrupt the validation process. An effective defense mechanism must be able to discern genuine validation results from fraudulent ones. Moreover, malicious validator clients can attempt inference attacks on models received for validation—further complicating the challenges faced by FLShield. Hence, FLShield’s validation mechanism should only transmit models to validators that are not vulnerable to inference attacks. Addressing these two primary challenges is crucial for ensuring an effective defense.

4.1.1. Challenge 1: Validation Subject Dilemma. This dilemma alludes to the challenge of selecting/computing appropriate models for validation in the FL setting, balancing the need for accuracy and privacy. The chosen validation model must be resilient to inference attacks while maintaining sufficient accuracy for validation purposes.

4.1.2. Challenge 2: Validation Integrity Dilemma. This dilemma pertains to the challenge of ensuring the integrity of the validation results in the FL setting, particularly in discerning between genuine and falsified validation results. The necessity for verification of these validation results stems from the fact that the clients performing the validation could be controlled by the adversary.

4.2. Solving The Validation Subject Dilemma

To address this dilemma, we introduce the concept of representative models which are fusions of multiple local model updates. We utilize two strategies to generate such representative models: (1) *bijjective representative models*: each local model update is treated as a base update and then accompanied by additional contributions from other local updates weighted according to their similarity to the base update. (2) *cluster representative models*: model updates are clustered into multiple groups, and updates within each group are aggregated to form a representative model.

By aggregating local model updates trained on the datasets of multiple clients, representative models yield more generalized results and are thus more suitable for validation. The integration of multiple local updates also thwarts privacy attacks launched by adversary-controlled malicious validators as it prevents successful reconstruction of training data. We perform gradient inversion attack on these representative models and observe that these attacks are unsuccessful in reconstructing any image resembling the clients’ training data (details later in section 6.5). Both of the generation strategies mentioned above demonstrate a high probability of merging benign updates with each other, and vice versa (section 6.4.1).

4.2.1. Bijjective Representative Models. The fundamental concept of bijjective representative models involves utilizing one local model update as the foundation and incorporating input from other updates based on their similarity to the foundation (i.e., the base model). Formally, let \mathcal{E} be the representative model, ω be the base update, and \mathcal{U} be the set of all updates. The bijjective representative model is defined as:

$$\mathcal{E} = G_t + (1-\tau)\omega + \tau \frac{\sum_{u \in \mathcal{U} - \{\omega\}} \left(s(\omega, u) \times u \times \frac{|\omega|}{|u|} \right)}{\sum_{u \in \mathcal{U} - \{\omega\}} \left(s(\omega, u) \times \frac{|\omega|}{|u|} \right)} \quad (4)$$

where $s(\omega, u)$ denotes the similarity between base update ω and another update u , and τ denotes the proportionate contribution ($0 < \tau < 1$) from other model updates to the representative model. We further name the term associated with τ as the *sibling contribution*. Owing to the one-to-one relationship between each representative model and local model, we coin the term *bijjective* representative model. In section 5.1.1, we discuss the requirements for the similarity function $s(\omega, u)$ and our approach to fulfill them. Sibling contribution is incorporated into the representative model to enhance its generalizability. This is because individual models trained on client-specific data subsets have weaker generalization than representative models which are composed of models trained on multiple clients’ datasets. However, for FLShield to mount a successful defense, two conditions must be fulfilled: (1) the sibling contribution need to be balanced to ensure the representative models achieve sufficient generalization without compromising the preeminent role of the base model, and (2) the benign representative

model must contain a lesser proportion of contribution from malicious updates compared to those originating from other benign updates. In sections 6.4.1 and 6.4.2, we demonstrate how fulfilling these conditions lead to a successful defense.

4.2.2. Cluster Representative Models. The technique for generating cluster representative models consists of two steps: (1) dynamically clustering clients based on their model updates, and (2) averaging the updates within each cluster to create a representative model. Existing defenses claim that benign and malicious clients can be separated by clustering on the model updates [28], [33], [43]. However, in non-IID scenarios, distinguishing them becomes challenging as data distribution shifts can cause benign model updates to seem more divergent than malicious ones. Despite this, we recognize the value of clustering as a useful tool. The non-IID data prevalent among FL clients necessitates a clustering algorithm that dynamically adjusts the number of clusters. The dynamic clustering coupled with a quality evaluation metric is expected to group clients in a way that minimizes the intermingling of benign and malicious updates. The veracity of the aforementioned claim is evidenced by experimental findings presented later in section 6.4.4. This fact holds significance as it suggests that we can distinguish benign from malicious representative models during validation, presuming benign representative models surpass malicious ones in performance. Consequently, the combination of clustering and validation mechanisms not only aids in the separation of benign and malicious clients but also ensures that the selected representative models contribute positively to the global model—enhancing its overall performance and robustness against attacks. The specifics of our clustering algorithm along with its associated hyperparameters are discussed in detail in section 5.1.2.

Suitable Use-cases. FL systems host hundreds to tens of thousands of participants. Bijjective representative model generation, while computationally demanding, ensures individual evaluation, making it apt for smaller systems. Conversely, the clustering-based approach, offering higher performance throughput, may blend benign and malicious updates in smaller cohorts, thus proving more suitable for larger systems.

4.3. Solving the Validation Integrity Dilemma

In FLShield, local clients act as validators for representative models, assess them and report validation results to the server. Nevertheless, verifying the integrity of these validation results is challenging. Variations in data distribution among clients could lead to different validation outcomes from different benign validators for the same representative model—making the identification of false results based on discrepancies from true results potentially challenging. Therefore, we introduce a new validation metric, loss impact per class (LIPC), which is computed class-wise to reduce the influence of diverse data distributions on validation results. Our experiments reveal that employing LIPC in the validation process causes honest validation results to be

more consistent with each other (section 6.4.2). We formally define the LIPC metric as follows:

$$\mathcal{L}(\mathcal{E}, v) = \left[\underset{(x,y) \in D_v, y=i}{\text{Mean}} |L(G, (x, y)) - L(\mathcal{E}, (x, y))| \mid i \in [1, c] \right] \quad (5)$$

$$\mathcal{M}(\mathcal{E}) = \left[\mathcal{L}(\mathcal{E}, G, v_i)^T \mid i \in [1, k] \right]^T \quad (6)$$

$$\mathcal{N}(v) = \left[\mathcal{L}(\mathcal{E}_i, G, v)^T \mid i \in [1, m] \right]^T \quad (7)$$

\mathcal{L} , a vector of class-wise loss value differences between the representative model and the prior round’s global model, focuses on loss impact rather than raw loss values (due to the incremental nature of FL). D_v represents the validation dataset of validator v , G denotes the global model from previous round, \mathcal{E} is the representative model under validation, L is the cross-entropy loss function, and c signifies the number of classes. \mathcal{M} is a matrix for \mathcal{E} containing all validators’ \mathcal{L} vectors. \mathcal{N} is a matrix from v containing \mathcal{L} vectors computed for all representative models. k and m represent the number of validators and representative models, respectively.

For a validation sample (x, y) and a representative model \mathcal{E} , the cross-entropy loss is given by $-\log(\mathcal{E}(x)[y])$, which represents the logit output for the correct class. A stable and benign neural network should yield similar logit values for all samples within a class, while a poisoned network is expected to produce varying logits for the source class. By computing the representative model’s \mathcal{L} for each class, we anticipate similar \mathcal{N} from benign validators even in non-IID scenarios. As shown later in section 6.4.2 and 6.2.2, \mathcal{N} from benign validators are indeed close to one another, supporting this hypothesis.

We perform thorough testing in various distribution scenarios to confirm the effectiveness of LIPC metric in ensuring validation integrity. We introduce label skew and feature skew in validators’ validation data (section 6.2) as well as vary false validation result injection strategies (section 6.3). In all cases, use of LIPC enabled FLShield to detect false validation results effectively. Moreover, using LIPC leads to the best performance amongst the metrics we consider later in section 6.4.2. These experiments substantiate the robustness and efficacy of our proposed validation approach.

5. Design and Implementation of FLShield

FLShield’s pipeline in each FL training round consists of the following steps: (1) Clients train local models and submit them to the server. (2) Server generates representative models from local model updates. (3) Validators evaluate the representative models and report \mathcal{N} to the server. (4) Server filters outlier \mathcal{N} s. (5) Server selects top 50% representative models after ranking them based on the projection of \mathcal{M} to a scalar value. (6) Local models in accepted representative models are selected. (7) Selected local model updates undergo norm-bounded clipping. (8) Clipped local model updates are averaged to obtain the new global model.

FLShield is designed to be flexible and adaptable to various FL systems, considering factors such as client

numbers, client-server relationships, and computational resources. We propose two algorithms for representative model generation: *bijjective* and *cluster*, detailed in sections 5.1.1 and 5.1.2, respectively.

Figure 1 illustrates one instantiation of the FLShield defense algorithm using *bijjective* representative model generator. Other defense algorithm components remain abstracted, providing a high-level overview of FLShield. The general framework of FLShield is outlined in the Algorithm presented in Figure 2. FLShield is divided into five components: 1) representative model generator, 2) model validation, 3) filtering, 4) clipping, and 5) aggregation. We discuss each of these components in the following sections.

5.1. Representative Model Generator

We mentioned two strategies for representative model generation in section 4.2. In this section, we explain the implementation details.

5.1.1. Bijjective Representative Model Generator. We provide an implementation of the bijjective representative generation method in the Algorithm presented in Figure 13. This approach constructs a representative model for each client using their local update as the base, and adding sibling contribution using equation 4. The similarity function must satisfy two conditions: (1) higher contribution from updates with similar direction (2) zero contribution from updates with opposing direction. We have chosen a combination of ReLU and cosine similarity as the similarity function. ReLU’s rectification meets the second criterion and cosine similarity’s constrained output range meets the first criterion.

5.1.2. Cluster Representative Model Generator. This generator using the Algorithm presented in Figure 12 applies K-Means clustering to local model updates. To identify the optimal cluster count, the algorithm utilizes silhouette score [40], a popular metric for optimal cluster determination, with min-max limits set by the system designer. After determining the cluster count, it generates representative models by averaging the updates in each cluster. Finally, it returns a client-cluster mapping and the representative models.

5.2. Model Validation

The implementation of the model validation unit is detailed in the Algorithm presented in Figure 14. For each representative model, k validators are randomly selected from the clients. The outer loop (line 2) iterates over representative models, allowing parallelization if many clients are available, leading to lower computational cost and higher throughput. In systems with fewer clients, these clients typically possess more computational resources, allowing one validator to evaluate multiple representative models despite high-latency communication.

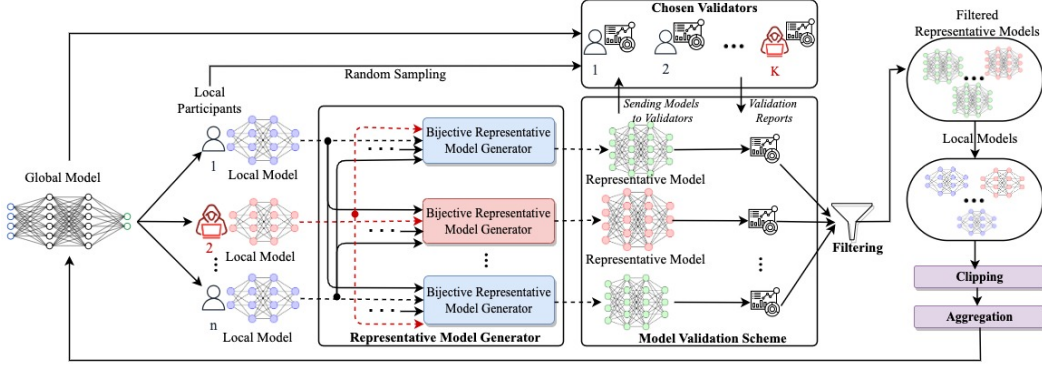


Figure 1: FLShield framework with bijective representative models.

Figure 2: FLShield Algorithm

Input: $G_0, T \triangleright G_0$ is the initial global model, T is the number of training iterations
Output: $G_T \triangleright G_T$ is the final global model

- 1: **for** each training iteration t in $[1, T]$ **do**
- 2: \triangleright Local Training
- 3: **for** each client i in $[1, n]$ **do**
- 4: $\omega_i \leftarrow \text{ClientUpdate}()$
- 5: **end for**
- 6: $M, (\mathcal{E}_1, \dots, \mathcal{E}_m) \leftarrow \text{RepresentativeGen}(\omega_{1:n}) \triangleright M : [1, n] \times [1, m]$ is the mapping function between local model updates and representative models $\mathcal{E}_1, \dots, \mathcal{E}_m$, for bijective representative it is identity mapping
- 7: $\mathcal{M}_1, \dots, \mathcal{M}_m \leftarrow \text{FedValidation}(G_t, \mathcal{E}_1, \dots, \mathcal{E}_m) \triangleright \mathcal{M}_1, \dots, \mathcal{M}_m$ are the \mathcal{M} report for each representative model by the unfiltered validators
- 8: $\mathbb{I}_r \leftarrow \text{Filtering}(\mathcal{M}_1, \dots, \mathcal{M}_m) \triangleright \mathbb{I}_r$ is the set of indices of the representative models that are accepted by the filtering scheme
- 9: $\mathbb{I} \leftarrow \{M_i : \forall i \in \mathbb{I}_r\} \triangleright \mathbb{I}$ is the set of indices of the local models that are accepted by the validation scheme
- 10: $\tilde{\omega}_{1:|\mathbb{I}|} \leftarrow \text{Clipping}(\{\omega_i : \forall i \in \mathbb{I}\}) \triangleright \tilde{\omega}_{1:|\mathbb{I}|}$ is the set of clipped local model updates
- 11: $G_{t+1} \leftarrow \text{Aggregation}(G_t, \tilde{\omega}_{1:|\mathbb{I}|})$
- 12: **end for**
- 13: **return** G_T

5.2.1. \mathcal{L} Calculation by Validator. The Algorithm presented in Figure 15 illustrates the \mathcal{L} calculation by the validator, with lower and upper bounds (n_1 and n_2) set for the number of validation samples for each class. Validators having fewer than n_1 samples are suggested to leave the value empty for that class. On the other hand, employing over n_2 samples might slow down the validation process without providing substantial defense benefits. Hence, validators are discouraged from using more than n_2 samples. Although validators might deviate from these specifications, adherence is expected to maintain consistency and prevent result discarding.

5.2.2. Filtering \mathcal{N} anomalies by the Server. The server utilizes IterativeImputation algorithm [41] (which outperforms others as shown in section 6.4.4) to fill missing values in \mathcal{L} scores before applying an outlier detection algorithm to \mathcal{N} matrices to filter potential malicious validation reports. We use elliptic envelope for outlier detection. However, according to our observation, the choice of the algorithm does not impact performance at all as long as it is a robust

one (section B.1). We further show that the absence of the outlier detector leads to failure under FLShield-aware attacks (section 6.4.4).

5.3. Filtering

The filtering unit outlined in the Algorithm presented in Figure 16 is responsible for filtering malicious representative models based on their \mathcal{M} matrix. We perform the following steps on each \mathcal{M} : (1) The matrix is averaged across the first dimension i.e. all the \mathcal{L} vector reported by the unfiltered validators for the corresponding representative model are averaged (2) The minimum value of the average \mathcal{L} vector is extracted. Afterwards, we examine the minimum value for each representative model, selecting the top 50% based on these scores across all classes. This choice reflects our threat model's assumption that malicious clients constitute less than 50% of the total clients. Untargeted poisoning attacks generally influence the \mathcal{L} vectors across all classes, whereas targeted attacks predominantly impact one class. As a result, utilizing the minimum validation scores proves to be an effective measure for detecting both targeted and untargeted poisoning attacks. Upon selecting the representative models, we utilize the mapping to identify the local model updates that contributed to these chosen representative models (line 9, FLShield Algorithm). In the remainder of the literature, we refer to the minimum of the \mathcal{L} vector as the \mathcal{L} score and the class-by-class evaluation as \mathcal{L} vector.

5.4. Clipping

In FL, malicious clients can send disproportionately large model updates to unfairly and detrimentally influence the global model. To counter this issue, FLShield incorporates a clipping technique similar to FLAME [33].

5.5. Aggregation

To obtain the new global model, we average the selected local model updates and combine them with previous round's global model.

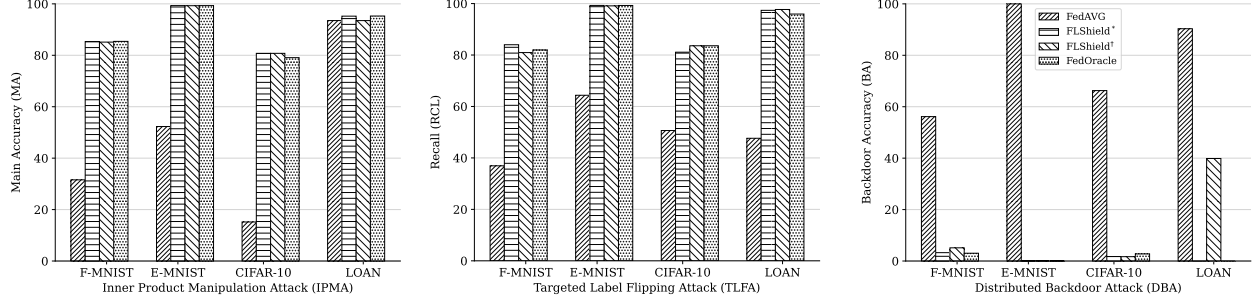


Figure 3: The performance of our defense against three different categories of poisoning attacks.

TABLE 1: Effectiveness of FLShield in comparison with state-of-the-art defenses against targeted poisoning attacks.

Defense	Targeted Label Flipping Attack (TLFA)												Distributed Backdoor Attack (DBA)			
	F-MNIST			E-MNIST			CIFAR-10			LOAN			F-MNIST		CIFAR-10	
	RCL	ASR	MA	RCL	ASR	MA	RCL	ASR	MA	RCL	ASR	MA	BA	MA	BA	MA
FedOracle	81.97	6.4	87.76	99.26	0.2	99.39	83.60	4.46	78.76	95.93	3.98	95.42	3.03	87.72	2.75	79.06
FedAvg	36.93	35.97	82.87	64.36	34.46	95.95	50.65	42.08	76.16	47.64	52.13	76.91	56.15	87.07	66.27	78.94
RFA	26.94	44.73	82.03	77.84	20.86	97.31	52.8	7.83	79.35	32.85	65.74	70.65	91.44	86.84	6.06	79.75
AFA	36.91	35.97	82.88	58.09	40.76	95.29	48.64	44.77	75.79	37.84	59.96	72.88	53.8	87.05	68.62	78.82
FLAME	42.04	41.4	81.64	78.2	20.68	97.26	48.92	45.38	72.88	82.18	6.19	90.16	6.17	86.36	4.68	78.81
FLTrust	59.69	20.95	83.95	98.65	0.69	99.32	57.2	35.26	73.77	89.14	7.22	92.82	6.5	86.99	20.15	77.45
FLShield*	84	5.43	87.05	99.28	0.17	99.29	81.10	5.27	78.96	97.37	2.56	95.41	3.25	87.64	1.73	79.75
FLShield†	80.95	6.4	87.08	99.11	0.27	99.35	83.60	6.14	78.66	97.7	2.22	95.29	5.13	87.18	1.69	77.85

6. Evaluation

6.1. Experiment Setup

Datasets. We conduct experiments with four datasets: F-MNIST [52], E-MNIST [13], CIFAR-10 [27], and LOAN [17]. The first three pertain to the image domain while the last one represents tabular data. More details of the datasets are presented in Appendix A.1.

Attacks considered. We evaluate FLShield against three categories of attacks: poisoning attacks, privacy inference attacks, and attacks that are specifically designed against FLShield. The poisoning attacks include both untargeted (i.e., inner product manipulation attack or IPMA [54]) and targeted (i.e., targeted label flipping attack or TLFA [45] and backdoor) ones. Further, we consider three backdoor attacks: distributed backdoor attack (DBA) [53], edge-case backdoor attack (ECBA) [48], and semantic backdoor attack (SBA) [2]. We also consider a gradient inversion attack (GIA) [16] to evaluate if the representative models used in FLShield can be used by the malicious validators to reconstruct training data from other clients. As mentioned earlier, we design FLShield-aware attacks where malicious participants attempt to compromise the integrity of the validation process. We design two such attacks: FLShield-aware adaptive attack (FA-Adp) and FLShield-aware advanced attack (FA-Adv). Table 5 (in Appendix) summarizes the attacks we investigate in our experiments. Section A.2 provides more details of the attacks.

Baselines. We evaluate FLShield against two baselines: (1) FL aggregation without any defense (FedAvg), and (2) FL aggregation with perfect detection and removal of poisoned updates (FedOracle). FedAvg’s performance illustrates undefended attack effectiveness while

FedOracle serves as a target baseline for any robust defense against FL poisoning.

Existing defenses considered. We compare FLShield’s performance with the following state-of-the-art defenses: FLAME [33], FLTrust [10], Adaptive Federated Averaging (AFA) [32], and Robust Federated Aggregation (RFA) [34].

Metrics. We consider the metrics main accuracy (MA), recall (RCL), and backdoor accuracy (BA) as defined earlier in section 2. These metrics evaluate the performance of the final global model. To evaluate the effectiveness of FLShield in filtering malicious model updates, we use true positive rate (TPR) and true negative rate (TNR). TPR (TNR) is computed as the ratio of malicious (benign) model updates detected as malicious (benign) and is averaged across all iterations.

Data distribution. We consider both IID and non-IID scenarios. For non-IID, we implement two strategies: one-class-expert distribution (also used in [10]) and Dirichlet distribution [31]. In the first strategy, the clients are split into 10 groups each corresponding to a class, and then each client is given an equal number of samples where 50% of the samples are from the class corresponding to the client and the rest 50% of the samples are from the other classes. For the Dirichlet distribution [31], we use the same setup as [2].

FL setup. Unless otherwise specified, we assume 40% of the clients are malicious. We show the impact of varying the percentage of malicious clients (upto 45%) on FLShield’s performance in section 6.4.5. The settings/configurations of the FL systems, classifier architecture, and hyperparameters of the attacks are described in Appendix section A.

Code. We used implementation from [48] and [22] and extended it to cover the diverse range of attacks and de-

fenses that we experimented with. Should this manuscript be accepted, we plan to make the associated code publicly available.

6.2. Performance of FLShield

We show the overall performance of FLShield against IPMA, TLFA, and DBA in Figure 3 and the performance against EBCA and SBA are reported in Table 4(b). Both FLShield* and FLShield[†] are able to achieve performances close to FedOracle for all datasets against IPMA and TLFA. While FLShield[†]'s performance degrades when experimented with DBA on the LOAN dataset, FLShield* is able to achieve a performance similar to FedOracle for all cases. In section 6.4.1 and section 6.4.2, we conduct an in-depth analysis to show how the use of representative models and LIPC score contributes to FLShield's success. In section 6.4.3, we perform an ablation study to understand how FLShield succeeds against backdoor attacks. Also, we discuss the limitations of FLShield in section 8. According to our observation, FLShield outperforms FedOracle in multiple instances. This is due to the fact that FLShield's design allows it to filter not only the malicious updates but also the updates that do not generalize well. Hence, FLShield produces global models that are often more generalized compared to that of FedOracle as FedOracle filters only the malicious updates.

6.2.1. Comparison with Existing Defenses. Table 1 compares our proposed defenses with existing defenses. The results show that both FLShield* and FLShield[†] significantly outperform the existing defenses. Note that, both TLFA and DBA are covert attacks that do not significantly affect the main accuracy (MA) of the model. State-of-the-art defenses such as FLTrust [10], RFA [34], AFA [32] fail to detect these attacks because these defenses have primarily been designed to defend against untargeted poisoning attacks only (details of IPMA results in Table 8 in Appendix). Even though FLAME [33] has been designed to detect targeted attacks (more specifically, backdoor attacks), it does not perform well against TLFA. This is because HDBSCAN is not able to separate the clusters of the benign and malicious clients in the TLFA setting. In contrast, FLShield* and FLShield[†] are able to achieve performances close to FedOracle in all datasets against both TLFA and DBA across all metrics.

6.2.2. Performance in Non-IID Settings. We experiment with two non-IID strategies as mentioned in section 6.1. Also, note that, the LOAN dataset has a natural non-IID distribution. Figure 3 shows that FLShield is able to achieve performances comparable to FedOracle for the LOAN dataset. Figure 4 shows the performance against TLFA attack on F-MNIST dataset in non-IID settings. For both one-class-expert and Dirichlet distributions, FLShield performs similar to FedOracle. This is interesting since the validators are picked from the non-IID client groups as

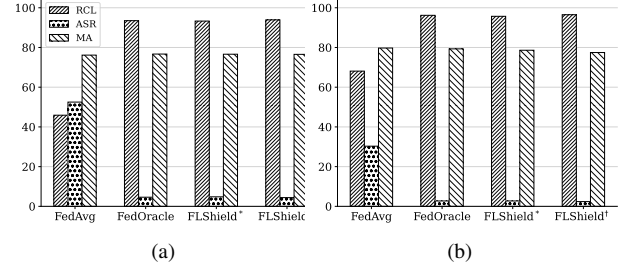


Figure 4: Evaluation of FLShield in non-IID scenario: (left) one-class-expert, (right) Dirichlet

well. More specifically, this result shows that the validation mechanism of FLShield does not require the validators to be IID. We deem the following as the primary reason for FLShield's success in non-IID scenarios.

- *The \mathcal{N} matrices of validators are independent from validation data distribution.* We flattened each \mathcal{N} matrix to a vector and projected the vectors after performing PCA in Figure 5(a). The \mathcal{N} matrices are taken from a one-class-expert setting with the F-MNIST dataset and as such there are 10 groups of clients with each group defined by the dominant class sample in clients' validation data. The figure shows that the \mathcal{N} matrices of validators are similar irrespective of the group they belong to. In fact, the intra-group mean distance of \mathcal{N} matrices is 0.0126 which is close to the inter-group mean distance of \mathcal{N} matrices of 0.0128. Hence, it is evident that the \mathcal{N} matrix is not influenced by the validation data distribution. This is because the \mathcal{N} matrix is computed in a class-wise manner. For a particular class, a validator from the corresponding group has more samples of the class than a validator from another group. If the \mathcal{N} was computed as only a vector containing the loss impact of all the representative models on the validators' data, then it would be biased by the dominant class of the validator. Figure 5(b) shows the PCA projection of \mathcal{N} vectors in this alternate scenario. Some groups of validators are reporting similar results to a group member but disparate results from a non-member. The similarity of original \mathcal{N} matrices implies the absence of disagreement among the validators which in turn enables FLShield to defend well in non-IID scenarios.

6.3. FLShield-aware Attacks by Malicious Validators

To evaluate FLShield's robustness, we assume an enhanced adversary which has the knowledge of FLShield and has additional capabilities to: (1) manipulate the \mathcal{N} matrices computed by malicious validators, and (2) estimate the \mathcal{N} matrices computed by benign validators.

The enhanced adversary aims to achieve two objectives: (1) *Stealth objective*: ensures that the \mathcal{N} matrices by the malicious validators cannot be distinguished from that of the benign validators by FLShield's outlier detection algorithm,

and (2) *Infiltration objective*: ensures that the malicious representative models rank in the top 50%. However, it is important to note that finding a set of crafted validation results that meet both objectives could be difficult for the adversary. This is because, for example, malicious representative models able to poison the FL system may not be stealthy. If these objectives cannot be achieved simultaneously, we can conclude that any attack that is aware of FLShield would still fail to poison the FL system. We design two enhanced adversaries to meet these two objectives at the same time:

(1) FLShield-aware adaptive attack (FA-Adp): In this approach, the adversary tries to achieve the two objectives through the formulation of an optimization problem. Such adaptive attacks have been designed to prove the effectiveness of other defenses in the literature, e.g., in FLTrust [10]. In this tactic, we define a loss function combining the two objectives. The optimal \mathcal{N} matrices are then derived by utilizing the stochastic gradient descent method to minimize the loss function. The strategy details are presented in Appendix C due to space constraints.

(2) FLShield-aware advanced attack (FA-Adv): In this approach, the malicious validators first compute the true \mathcal{N} matrices and then attempt to achieve their stealth and infiltration objectives by making minimal alterations to these original validation results. The magnitude of alteration can be quantified as the distance between the original and the crafted \mathcal{N} matrix by each malicious validator. While finding the minimal change necessary to meet the infiltration objectives is mathematically intractable, a suboptimal solution can be derived by following the steps detailed in the Algorithm presented in Figure 17 in Appendix.

Evaluation Results: Table 2 compares the performance of FLShield against FA-Adp and FA-Adv attacks including the case when there is no malicious validator. In the no malicious validator scenario, when a malicious participant is selected as a validator it does not manipulate the validation results. The attacks in this experiment are performed on the CIFAR-10 dataset. The results show that both versions of FLShield-aware attacks achieve limited success in terms of RCL, BA, TPR, and TNR. The only anomaly is the performance of FLShield* against the FA-Adv attack that reduces TPR and TNR by 10% under DBA. Figures 8(e), 8(f) display representative models ranked by \mathcal{L} score under DBA and TLFA respectively. DBA exhibits a smaller benign-malicious model gap compared to TLFA. FA-Adv’s strategy, aiming to narrow this gap, only finds limited success when minimum alterations don’t yield out-of-distribution \mathcal{N} . However, such occurrences are rare, and the limited infiltration is inadequate for an effective backdoor. The recall is higher in some defense-aware attack scenarios compared to non-defense-aware ones. However, recall is not a robust metric for evaluating the defense’s effectiveness due to various influencing factors in an FL system. Differences in benign model updates selected in cases like None and FA-Adp or FA-Adv lead to different recall values, even if they are close. TPR/TNR, in contrast, serves as a direct measure

Malicious Validators →		None	FA-Adp	FA-Adv	None	FA-Adp	FA-Adv
Metrics ↓	Attacks ↓	FLShield*			FLShield†		
RCL	TLFA	81.10	83.10	81.60	83.60	84.80	83.70
BA	DBA	1.73	1.72	2.47	1.69	1.48	1.87
TPR	TLFA	100	100	100	100	100	100
	DBA	100	100	90	100	100	100
TNR	TLFA	91	90.75	90.75	86.67	86.67	86.67
	DBA	100	100	90	86.67	86.67	86.67

TABLE 2: Evaluation of FLShield under FA-Adp and FA-Adv

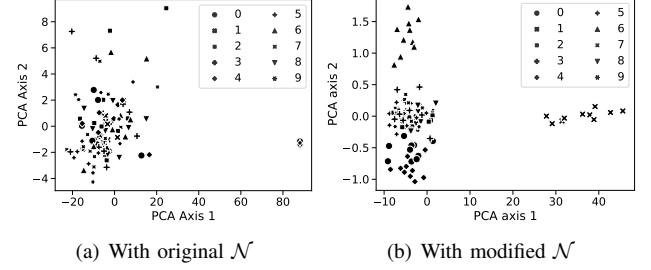


Figure 5: PCA projection of \mathcal{N} (left) and a modified version of \mathcal{N} (right) of 100 validators in the one-class-expert setting. Each validator belongs to one of the 10 groups denoted by the class index of the dominant class in their training data.

of the defense’s ability to eliminate poisoned model updates.

In section 6.4.2, we perform an ablation study to closely understand the reason behind the success of FLShield against malicious validator attacks.

6.4. Ablation Study

6.4.1. Why does FLShield work?. The following list of observations coupled with visualizations from experimental results shed light on the reasons behind the success of FLShield.

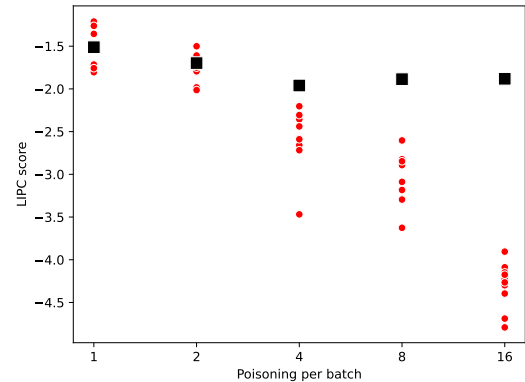


Figure 6: LIPC score of malicious representative models vs poisoned sample per batch (PSPB) in DBA. The square dot indicates the median LIPC score for the corresponding setting.

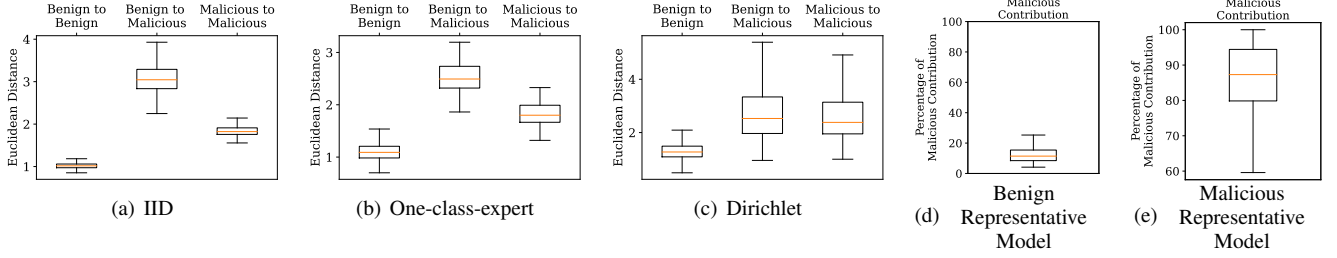


Figure 7: Euclidean distance between updates for three distribution settings: (a) IID, (b) one-class-expert, and (c) Dirichlet. Percentage of malicious contribution to (d) benign representative models and (e) malicious representative models.

- *The similarity between two benign (malicious) updates is greater than the similarity between a benign update and a malicious update.* Models trained with similar objectives are expected to bear resemblance with each other. Figure 7(a), 7(b), and 7(c) present results from the experiments to support this. The three boxplots show the range of distances between two benign updates, benign and malicious updates, and two malicious updates in three distribution settings: IID, one-class-expert, and Dirichlet. In all cases, the mean benign-benign distance is smaller than the mean malicious-benign distance. The experiments consider TLFA on F-MNIST dataset.

- *The contribution of benign (malicious) local models to a benign (malicious) representative model is greater than the contribution of the malicious (benign) local models.* This is because the contributions are weighted based on the contributor’s similarity with the base model. Figure 7(d) and 7(e) show the contribution of malicious participants to the benign and malicious representative models, respectively.

- *The performance of a representative model declines with an increase in the ratio of the malicious updates’ contribution to that model.* In other words, more malicious contribution implies worse performance in terms of \mathcal{L} scores. Figure 8(a) presents evidence of this behavior by showing the \mathcal{L} scores of 100 bijective representative model updates. The scores reflect a single iteration of FL with CIFAR-10 dataset and Dirichlet distribution where the adversary is launching TLFA.

- *All the representative models that have a malicious majority contribution rank at the bottom 50% of all the representative models in terms of performance.* The decreasing \mathcal{L} scores with increasing malicious contributions in Figure 8(a) complies with the above.

6.4.2. How does FLShield differentiate malicious validation reports from benign ones?. We perform an ablation study on the validation components of FLShield and conduct a comprehensive inspection to evaluate the efficacy of our filtering mechanisms. Our analysis helps us identify some key reasons behind FLShield’s ability to filter malicious validation results.

- *The use of \mathcal{L} as validation metric and the incorporation of sibling contribution into the representative models make the benign validation reports congruent and difficult to*

manipulate. Figure 8(b) casts the validation reports into two dimensions using PCA and portrays the congruency among the benign validators and the anomalous nature of the tailored validation reports created by the malicious validators. FLShield is able to achieve 100% TPR when sibling contribution rate τ is set to 75%, i.e., base model contributes the rest 25%. We use this as the default ratio for our experiments. The effects of varying sibling contributions in terms of RCL, TPR, and TNR are demonstrated in Figure 8(c). The results also show that when local models are validated directly, i.e., when τ is set to 0%, FLShield’s defense fails. Sibling contributions’ success can be attributed to the improved generalization ability brought to the representative models as described in section 5.1.

- *Stealthy poisoning can be identified using \mathcal{L} metric whereas accuracy-based metric fails.* To demonstrate the significance of \mathcal{L} metric, we design another metric *accuracy difference per class* (ADPC) and define it as the difference between accuracy values output by the global model and the representative model for each class. Next, we run experiments with an altered version of FLShield[†] that uses this new metric instead of \mathcal{L} . Figure 8(e) and 8(d) presents the results from an experiment with DBA. The results show that malicious and benign representative models are indistinguishable when measured with ADPC. The experiment results shown in Table 3(b) further ascertains that ADPC is not a suitable metric to use in FLShield. The experiment has been performed on CIFAR-10 dataset with FLShield[†] used in aggregation and the adversary launches TLFA.

- *Due to the robustness of the \mathcal{L} metric, the choice of outlier detector algorithm to filter malicious validation reports does not matter significantly.* We experimented with Elliptic envelope, Isolation forest, Local outlier factor and found no difference in terms of performance among them (section B.1). Note that, not using an outlier detector in the presence of malicious validators leads to failure which signifies the importance of outlier detection. The experimental results are presented in Table 3(a) for CIFAR-10 dataset where FLShield[†] defends against TLFA with malicious validators performing FA-Adv.

6.4.3. How does FLShield defend against backdoor attacks?. We perform an ablation study on various parameters of the backdoor attacks to understand the reasons behind

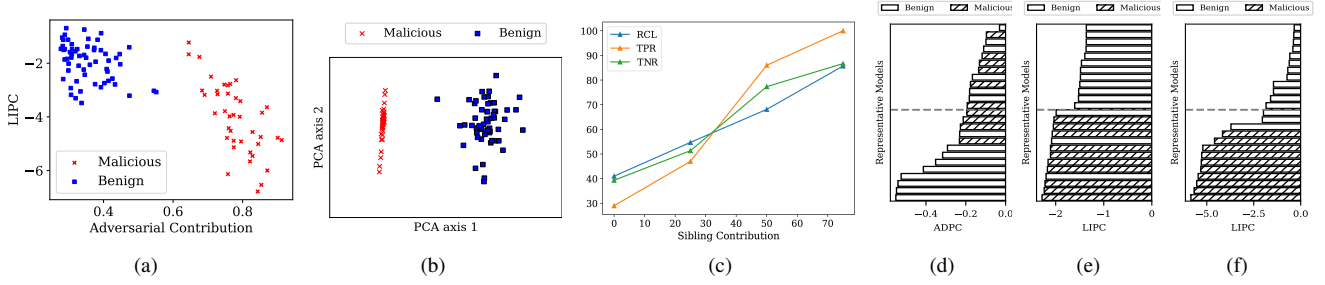


Figure 8: (a) LIPC vs. adversarial contribution, (b) visualization of validation reports after PCA projection, (c) performance with varying sibling contributions, and (d-f) representative models ranked by metrics: (left) ADPC, (middle) LIPC under DBA, (right) LIPC under TLFA.

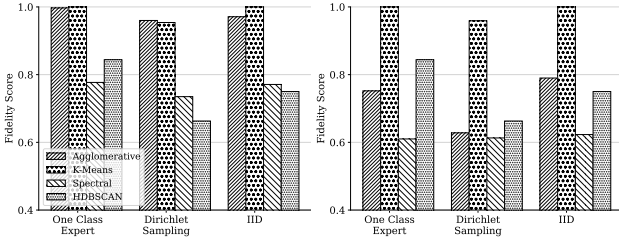


Figure 9: Fidelity score comparison of multiple clustering algorithms with: (left) dynamic clustering, (right) fixed number of clusters (2). Dataset: F-MNIST, attack: TLFA.

FLShield’s success against backdoor attacks.

- *LIPC score of malicious representative models is negatively correlated with poisoned sample per batch (PSPB) of the backdoor attack.* We conducted experiments under DBA with varying numbers of PSPB and reported the LIPC score of the malicious representative models in Figure 6. It is evident that the more backdoored samples are introduced to the training the more the LIPC score decreases from the median value. Backdoor poisoning involves both clean samples and backdoored samples to reduce the loss on the main task and backdoored task respectively. The increase of backdoored samples means the decrease of clean samples which in turn results in diminished loss reduction on the main task and that is reflected in the poor LIPC score of the malicious representative models.

- *The required PSPB to bypass FLShield is ineffective at setting up backdoors in the global model.* In Figure 6, it is shown that the malicious representative models get ranked in the top 50% with an extremely low poisoned sample per batch. However, if the PSPB is below 8, the poisoning is ineffective even in FedAvg i.e. the final global model does not perform well on the backdoor task (BA=3.31% at PSPB 4). Thus, it is clear that any effective backdoor poisoning requires a certain level of PSPB which in turn results in a discriminative LIPC score of the malicious representative models enabling FLShield to defend against backdoor attacks.

6.4.4. Explanation of Design Choices.

- *Would FLShield work if the mean of \mathcal{L} is computed instead of minimum before ranking the representative models?* Experimental results presented in Table 3(c) show that using mean instead of minimum leads to an overall loss of RCL, TPR, and TNR. The experiment is run on CIFAR-10 with FLShield[†] as defense and the adversary is launching TLFA.

- *Does the choice of clustering algorithm matter in FLShield*?* We conduct experiments employing a variety of clustering algorithms, including Agglomerative, K-means, Spectral, and HDBSCAN. To compare their effectiveness, we design a metric named *fidelity score* which is defined in the following way— for each cluster, we first determine whether it has a benign/malicious majority (ground-truth) and then the member count ratio of the majority is computed by dividing the count by the total number of participants in the cluster. Hence, fidelity score is a suitable metric to determine the overall purity of the clusters. The results are presented in Figure 9. It shows that K-Means have the best overall fidelity score among all which we use in FLShield as the default clustering algorithm.

- *Does the choice of imputation technique matter in filling out the missing values in the \mathcal{N} matrices?* We undertake an evaluation of different imputation techniques, namely KNNImpute, SimilarityWeightedAveraging, IterativeImpute, Mean, and Median as implemented in [41], by selectively removing a random fraction of the validation results and presenting the findings in Figure 10. Most algorithms perform well even mean and median where the missing value is simply replaced with mean and median respectively. This again goes to show the congruency of the validation reports. Iterative imputation slightly outperforms the others overall and as such, we use it as the default imputation in FLShield.

- *Why doesn’t FLShield aggregate representative models to obtain the global model?* The representative models selected after outlier removal still include some malicious contribution (see Figure 7(d)) and thus may still leave some poisoning footprints in the global model. Experimental results reported in Table 4(a) show that with the presence of malicious contribution of only 8.6%, aggregating the

	RCL	TPR	TNR
No Detector	15.70	10.00	26.67
Default	83.40	100.00	86.67

(a)

Metric used	RCL	% of Malicious Representatives Accepted
ADPC	25.34	26
LIPC	88.66	0

(b)

	RCL	TPR	TNR
Mean	73.00	80.60	73.73
Minimum	83.60	100.00	86.67

(c)

# of Validators	BA	TPR	TNR
10	3.61	97.50	81.67
15	3.46	100.00	88.89
20	2.64	100.00	83.33
25	1.89	100.00	86.67

(d)

TABLE 3: (a) Result Comparison between two scenarios: (1) no outlier detector (2) default version of FLShield.(b) Performance comparison between two validation metrics: LIPC vs Accuracy-difference-per-class.(c) Performance comparison between mean and minimum (default) as the projection step in Algorithm 16 Line 4. (d) BA, TPR, and TNR vs. number of validators.

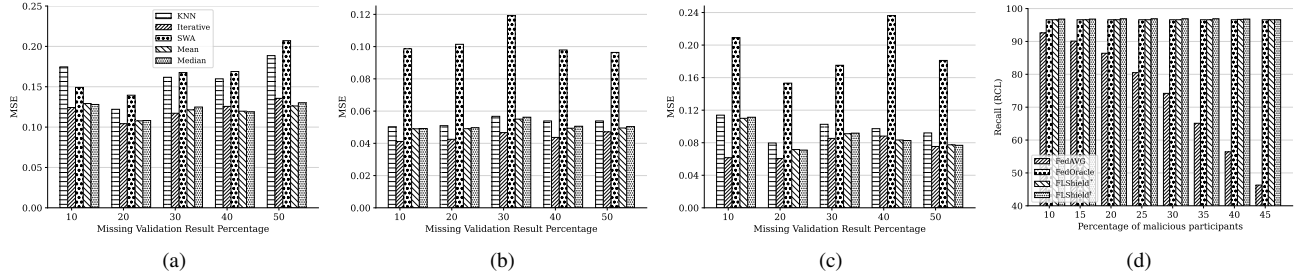


Figure 10: (a-c) Comparison of imputation techniques' performance with varying missing validation results in different data distribution: (from left to right) IID, one-class-expert, Dirichlet. (d) Effect of varying malicious client participation

Aggregation of	Malicious contribution %	TPR	TNR	BA
Representative	8.60	99	86	51
Individual	N/A	100	86.67	0.43

(a)

Defense	ECBA BA	SBA BA
FedAvg	85.71	100
FedOracle	8.67	0
FLShield*	9.18	0
FLShield [†]	7.65	0

(b)

TABLE 4: (a) Aggregation of representative models vs. individual local models. Dataset: CIFAR-10, Attack: DBA. (b) Performance evaluation of ECBA and SBA on CIFAR-10

representative models instead of the individual local models may increase the backdoor accuracy to 51%.

• *Does the number of validators selected in each iteration matter?* Table 3(d) presents comparison among experiments run with different number of validators ranging from 10 to 25 with a step size of 5. With only 10 validators, there is a slight drop in performance and in other cases, the TPR is 100% which again demonstrates the robustness of FLShield. The experiment is run on the CIFAR-10 dataset against DBA.

6.4.5. Varying Percentage of Malicious Clients. We vary the percentage of malicious clients in the TLFA setting and measure the performance of FLShield. We use F-MNIST dataset for this experiment and vary the percentage of malicious clients from 10% to 45% in steps of 5%. Figure 10(d) shows how the percentage of malicious clients influences FLShield* and FLShield[†]. As shown in the figure, unlike FedAvg, there is no noticeable drop in the performance.

6.5. Gradient Inversion Attack

Since sharing gradients in FL may leak private information [16], [22], [56], [60], we evaluate FLShield against a state-of-the-art gradient inversion attack [16] leveraging the attack implemented in [22]. Note that, in FLShield, only the representative gradients are sent to the validators. However, the malicious validators may still try to conduct gradient inversion attacks on the representative models to extract private local data. To eliminate this concern, we evaluate the robustness of the representative gradients on the CIFAR-10 dataset with ResNet-18 architecture. As mentioned in [16], there are two adversary knowledge assumptions: (1) private labels and (2) batch normalization statistics. The detailed attack setting is provided in Appendix A.2.

Figure 11 visualizes the images reconstructed from local gradients (first row), cluster representative models in FLShield* (second row), and bijective representative models in FLShield[†]. For FLShield* we launch the attack under different assumption. With respect to FLShield[†] we launch the attack under the strongest attacker assumption (the attacker has knowledge of private labels and batch normalization statistics) and variate the number of clients. The results show that the reconstructed images with representative gradients in FLShield are unrecognizable even in the strongest attack setup. We use the learned perceptual image patch similarity (LPIPS) score [58] to measure the performance of the gradient inversion attack in Table 9 in Appendix.

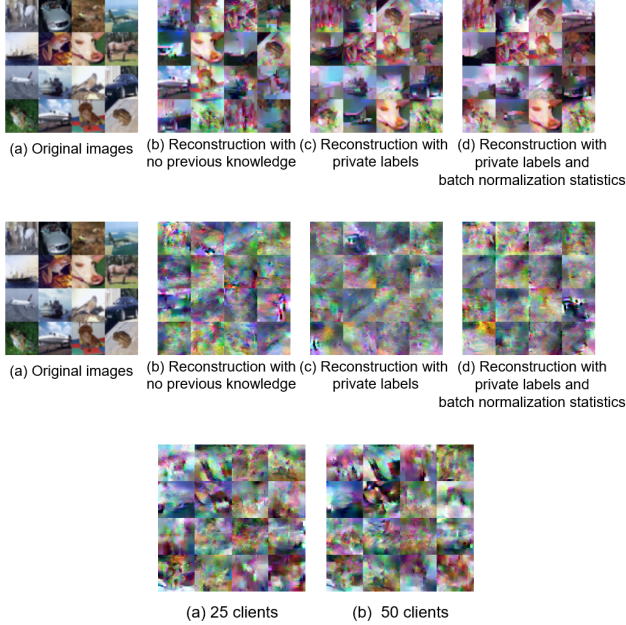


Figure 11: Comparison of images reconstructed from local gradients (first row), representative gradients in FLShield* (second row), and FLShield[†] (third row).

7. Related Work

Data Poisoning Attack against FL. Data poisoning is an adversarial attack that manipulating training datasets by injecting poisoned data to control the behavior of the trained model. This attack have been demonstrated to be successful on many machine learning systems [4], [11], [12], [23], [25], [29], [30], [51]. FL is also vulnerable to data poisoning. In FL, this attack involve maliciously manipulating or poisoning the data contributed by specific clients in order to compromise the overall model’s performance [2], [3], [45], [48], [53]. Depending on the attacker’s goal, this kind of attacks can be categorized into untargeted attacks [14] and targeted attacks [2], [3], [53]. Untargeted attacks aim to make the learned model unusable by degrading the performance of the model, while targeted attacks only influence the model’s behavior towards misclassifying certain target inputs.

Robust Aggregation Frameworks. A number of robust FL aggregation frameworks have been proposed [1], [2], [5], [10], [15], [24], [32], [33], [38], [43], [44], [55] to mitigate the threat of Byzantine participants. There are mainly two categories of methods: (1) identifying and filtering out malicious clients, and (2) adding noise to local updates.

Identifying and filtering out malicious clients is a commonly employed method. Some methods [5], [12], [32], [43], [55] mitigate malicious clients by removing local updates which are far away from the majority. However, these methods are highly sensitive to outliers in the client updates and may exclude benign clients with significantly different updates. FLTrust [10], which makes the assumption that the defender has access to a server-side clean dataset, leverages

cosine similarity to identify malicious intent in client updates deviating from a benign server model. However, the challenge lies in assembling such a clean dataset.

Certain methods allow clients to act as validators to detect malicious updates. Zhao et al. [61] propose assessing the integrity of local models using client’s local data, marking poorly performing models as malicious, but this method is unable to detect targeted attacks that don’t diminish the overall performance of the model. BaFFLe [1] lets selected clients to validate the aggregated model but also shares the same problem.

Recent works have introduced many novel statistical techniques to identify abnormal updates. FLDetector [59] let the server predict local model updates based on the historical updates and then identify the malicious updates by comparing the predicted updates and the received updates. However, it is not necessary for the malicious clients to keep changing their local data distribution. Deepsight [38] remove the local updates that trained with datasets where a single class dominates the majority of the labels by estimating distribution of clients’ local data. Nevertheless, it’s easy for an attacker to construct a training set to make the malicious clients evade the detection and the benign clients who only have homogeneous data will suffer from bad model performance. Adding noise to local updates have been explored as a potential option [2], [33], [44] but the noise injection causes utility loss to the trained model.

Gradient Inversion Attacks. Gradient Inversion Attack (GIA) [16], [50], [56] aims to reconstruct training samples from gradient for image classification tasks. Adversaries leverage various reconstruction techniques, such as optimization algorithms or machine learning models, to analyze the gradients and infer the underlying training samples. In FLShield representative gradients are sent to clients for validation. Malicious clients may try to reconstruct the training samples from the these gradients.

8. Limitations

One limitation of FLShield is that it has been designed to work in FL systems with a classification task. The extension of FLShield’s validation mechanism to other tasks, e.g., text generation and recommendation systems seem challenging. It remains a future work to investigate how the insights gained from FLShield could be used in those tasks. In FLShield, all model updates selected post-validation receive equal weight, despite their respective scores. We did not integrate an adaptive weighting scheme that might assign a higher weight to models achieving better validation scores. Although this has not resulted performance drop in terms of any metric, investigating alternative schemes may be advantageous for an FL system that uses an incentive mechanism to stimulate the provision of high-quality data. Nevertheless, we leave this exploration as a future work.

9. Conclusion

We introduce a novel validation-based defense framework for Federated Learning (FL) `FLShield` which is adept at resolving the *validation dilemmas* without creating any system vulnerabilities and exhibits robustness against three types of poisoning attacks. Further, `FLShield` can efficiently defend against two defense-aware attacks, specifically, FA-Adp and FA-Adv. We also perform gradient inversion attack experiments, demonstrating that malicious validators are incapable of reconstructing training data from the shared representative models. This design aligns well with existing privacy precautions against inference attacks. We demonstrate that employing a validation process is the sole reliable defense strategy against poisoning attacks in FL. Despite our framework being tailored for training classifiers, we anticipate its potential extension to other machine learning tasks, an area we earmark for future research.

References

- [1] ANDREINA, S., MARSON, G. A., MOLLER, H., AND KARAME, G. BaFFLe: Backdoor detection via feedback-based federated learning. In *Proceedings - International Conference on Distributed Computing Systems* (2021), vol. 2021-July.
- [2] BAGDASARYAN, E., VEIT, A., HUA, Y., ESTRIN, D., AND SHMATIKOV, V. 20-AISTATS-C-18-17-How To Backdoor Federated Learning. *AISTATS* (2018).
- [3] BHAGOJI, A. N., CHAKRABORTY, S., MITTAL, P., AND CALO, S. Analyzing federated learning through an adversarial lens. In *36th International Conference on Machine Learning, ICML 2019* (2019), vol. 2019-June.
- [4] BIGGIO, B., NELSON, B., AND LASKOV, P. Poisoning attacks against support vector machines, 2012.
- [5] BLANCHARD, P., EL MHAMDI, E. M., GUERRAOU, R., AND STAINER, J. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems* (2017), vol. 2017-December.
- [6] BONAWITZ, K., EICHNER, H., GRIESKAMP, W., HUBA, D., INGERMAN, A., IVANOV, V., KIDDON, C., KONEČNÝ, J., MAZZOCCHI, S., MCMAHAN, B., ET AL. Towards federated learning at scale: System design. *Proceedings of Machine Learning and Systems 1* (2019), 374–388.
- [7] BONAWITZ, K., IVANOV, V., KREUTER, B., MARCEDONE, A., MCMAHAN, H. B., PATEL, S., RAMAGE, D., SEGAL, A., AND SETH, K. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (2017), pp. 1175–1191.
- [8] BRENDAN MCMAHAN, H., MOORE, E., RAMAGE, D., HAMPSON, S., AND AGÜERA Y ARCAS, B. Communication-Efficient Learning of Deep Networks from Decentralized Data. *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017* (2 2016).
- [9] BRISIMI, T. S., CHEN, R., MELA, T., OLSHEVSKY, A., PASCHALIDIS, I. C., AND SHI, W. Federated learning of predictive models from federated electronic health records. *International journal of medical informatics 112* (2018), 59–67.
- [10] CAO, X., FANG, M., LIU, J., AND GONG, N. Z. Fltrust: Byzantine-robust federated learning via trust bootstrapping. In *28th Annual Network and Distributed System Security Symposium, NDSS 2021, virtually, February 21-25, 2021* (2021), The Internet Society.
- [11] CHEN, X., LIU, C., LI, B., LU, K., AND SONG, D. Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning. *arXiv e-prints* (Dec. 2017), arXiv:1712.05526.
- [12] CHEN, Y., SU, L., AND XU, J. Distributed Statistical Machine Learning in Adversarial Settings. *ACM SIGMETRICS Performance Evaluation Review 46*, 1 (2019).
- [13] COHEN, G., AFSHAR, S., TAPSON, J., AND VAN SCHAIK, A. Emnist: Extending mnist to handwritten letters. In *2017 international joint conference on neural networks (IJCNN)* (2017), IEEE, pp. 2921–2926.
- [14] FANG, M., CAO, X., JIA, J., AND GONG, N. Z. Local Model Poisoning Attacks to Byzantine-Robust Federated Learning. *Proceedings of the 29th USENIX Security Symposium* (11 2019), 1623–1640.
- [15] FUNG, C., YOON, C. J., AND BESCHASTNIKH, I. The limitations of federated learning in sybil settings. In *RAID 2020 Proceedings - 23rd International Symposium on Research in Attacks, Intrusions and Defenses* (2020).
- [16] GEIPING, J., BAUERMEISTER, H., DRÖGE, H., AND MOELLER, M. Inverting gradients - how easy is it to break privacy in federated learning? In *Advances in Neural Information Processing Systems* (2020), H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., pp. 16937–16947.
- [17] GEORGE, N. All lending club loan data, 2019. URL <https://www.kaggle.com/datasets/zaurbegiev/my-dataset>.
- [18] HAO, M., LI, H., LUO, X., XU, G., YANG, H., AND LIU, S. Efficient and privacy-enhanced federated learning for industrial artificial intelligence. *IEEE Transactions on Industrial Informatics 16*, 10 (2019), 6532–6542.
- [19] HARDY, S., HENECKA, W., IVEY-LAW, H., NOCK, R., PATRINI, G., SMITH, G., AND THORNE, B. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. *arXiv preprint arXiv:1711.10677* (2017).
- [20] HE, K., ZHANG, X., REN, S., AND SUN, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 770–778.
- [21] HITAJ, B., ATENIESE, G., AND PEREZ-CRUZ, F. Deep models under the gan: information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security* (2017), pp. 603–618.
- [22] HUANG, Y., GUPTA, S., SONG, Z., LI, K., AND ARORA, S. Evaluating gradient inversion attacks and defenses in federated learning. In *NeurIPS* (2021).
- [23] JAGIELSKI, M., OPREA, A., BIGGIO, B., LIU, C., NITA-ROTARU, C., AND LI, B. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *2018 IEEE Symposium on Security and Privacy (SP)* (2018), pp. 19–35.
- [24] JEBREEL, N., AND DOMINGO-FERRER, J. FI-defender: Combating targeted attacks in federated learning, 2022.
- [25] KOH, P. W., AND LIANG, P. Understanding black-box predictions via influence functions, 2017.
- [26] KONEČNÝ, J., MCMAHAN, H. B., YU, F. X., RICHTÁRIK, P., SURESH, A. T., AND BACON, D. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492* (2016).
- [27] KRIZHEVSKY, A., HINTON, G., ET AL. Learning multiple layers of features from tiny images.
- [28] LI, X., QU, Z., ZHAO, S., TANG, B., LU, Z., AND LIU, Y. Lomar: A local defense against poisoning attack on federated learning. *IEEE Transactions on Dependable and Secure Computing* (2021).
- [29] LIU, Y., MA, S., AAFER, Y., LEE, W.-C., ZHAI, J., WANG, W., AND ZHANG, X. Trojaning attack on neural networks. In *NDSS* (2018), The Internet Society.
- [30] MEI, S., AND ZHU, X. Using machine teaching to identify optimal training-set attacks on machine learners. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence* (2015), AAAI’15, AAAI Press, p. 2871–2877.
- [31] MINKA, T. Estimating a dirichlet distribution, 2000.
- [32] MUÑOZ-GONZÁLEZ, L., CO, K. T., AND LUPU, E. C. Byzantine-robust federated machine learning through adaptive model averaging, 2019.
- [33] NGUYEN, T. D., RIEGER, P., CHEN, H., YALAME, H., MÖLLER, H., FEREDOONI, H., MARCHAL, S., MIETTINEN, M., MIRHOSEINI, A., ZEITOUNI, S., KOUSHANFAR, F., SADEGHI, A.-R., AND SCHNEIDER, T. Flame: Taming backdoors in federated learning, 2021.
- [34] PILLUTLA, K., KAKADE, S. M., AND HARCHAOU, Z. Robust Aggregation for Federated Learning.
- [35] POKHREL, S. R., AND CHOI, J. A decentralized federated learning approach for connected autonomous vehicles. In *2020 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)* (2020), IEEE, pp. 1–6.
- [36] POKHREL, S. R., AND CHOI, J. Federated learning with blockchain for autonomous vehicles: Analysis and design challenges. *IEEE Transactions on Communications 68*, 8 (2020), 4734–4746.

- [37] QU, Y., POKHREL, S. R., GARG, S., GAO, L., AND XIANG, Y. A blockchained federated learning framework for cognitive computing in industry 4.0 networks. *IEEE Transactions on Industrial Informatics* 17, 4 (2020), 2964–2973.
- [38] RIEGER, P., NGUYEN, T. D., MIETTINEN, M., AND SADEGHI, A.-R. DeepSight: Mitigating backdoor attacks in federated learning through deep model inspection. In *Proceedings 2022 Network and Distributed System Security Symposium* (2022), Internet Society.
- [39] RIEKE, N., HANCOX, J., LI, W., MILLETARI, F., ROTH, H. R., ALBARQOUNI, S., BAKAS, S., GALTIER, M. N., LANDMAN, B. A., MAIER-HEIN, K., ET AL. The future of digital health with federated learning. *NPJ digital medicine* 3, 1 (2020), 1–7.
- [40] ROUSSEEUW, P. J. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics* 20 (1987), 53–65.
- [41] RUBINSTEYN, A., AND FELDMAN, S. fancyimpute: An imputation library for python.
- [42] SHEJWALKAR, V., AND HOUMANSADR, A. Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning. In *NDSS* (2021).
- [43] SHEN, S., TOPLE, S., AND SAXENA, P. Auror: defending against poisoning attacks in collaborative deep learning systems. *Proceedings of the 32nd Annual Conference on Computer Security Applications* (2016).
- [44] SUN, Z., KAIROUZ, P., SURESH, A. T., AND MCMAHAN, H. B. Can you really backdoor federated learning?, 2019.
- [45] TOLPEGIN, V., TRUEX, S., GURSOY, M. E., AND LIU, L. Data poisoning attacks against federated learning systems. In *European Symposium on Research in Computer Security* (2020), Springer, pp. 480–501.
- [46] TRUEX, S., BARACALDO, N., ANWAR, A., STEINKE, T., LUDWIG, H., ZHANG, R., AND ZHOU, Y. A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM workshop on artificial intelligence and security* (2019), pp. 1–11.
- [47] TRUONG, N., SUN, K., WANG, S., GUITTON, F., AND GUO, Y. Privacy preservation in federated learning: An insightful survey from the gdpr perspective. *Computers & Security* 110 (2021), 102402.
- [48] WANG, H., SREENIVASAN, K., RAJPUT, S., VISHWAKARMA, H., AGARWAL, S., SOHN, J.-Y., LEE, K., AND PAPALIOPOULOS, D. Attack of the tails: Yes, you really can backdoor federated learning. *Advances in Neural Information Processing Systems* 33 (2020), 16070–16084.
- [49] WEI, K., LI, J., DING, M., MA, C., YANG, H. H., FAROKHI, F., JIN, S., QUEK, T. Q., AND POOR, H. V. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security* 15 (2020), 3454–3469.
- [50] WEI, W., LIU, L., LOPER, M., CHOW, K.-H., GURSOY, M. E., TRUEX, S., AND WU, Y. A framework for evaluating gradient leakage attacks in federated learning, 2020.
- [51] XIAO, H., BIGGIO, B., BROWN, G., FUMERA, G., ECKERT, C., AND ROLI, F. Is feature selection secure against training data poisoning?
- [52] XIAO, H., RASUL, K., AND VOLLGRAF, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747* (2017).
- [53] XIE, C., HUANG, K., PIN-YU, C., AND LI, B. Dba : Distributed Backdoor Attacks. *8th International Conference on Learning Representations, {ICLR} 2020* (2020).
- [54] XIE, C., KOYEJO, O., AND GUPTA, I. Fall of empires: Breaking byzantine-tolerant sgd by inner product manipulation. In *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference* (22–25 Jul 2020), R. P. Adams and V. Gogate, Eds., vol. 115 of *Proceedings of Machine Learning Research*, PMLR, pp. 261–270.
- [55] YIN, D., CHEN, Y., RAMCHANDRAN, K., AND BARTLETT, P. Byzantine-robust distributed learning: Towards optimal statistical rates. In *35th International Conference on Machine Learning, ICML 2018* (2018), vol. 13.
- [56] YIN, H., MALLA, A., VAHDAT, A., ALVAREZ, J. M., KAUTZ, J., AND MOLCHANOV, P. See through gradients: Image batch recovery via gradinversion, 2021.
- [57] ZHANG, C., LI, S., XIA, J., WANG, W., YAN, F., AND LIU, Y. {BatchCrypt}: Efficient homomorphic encryption for {Cross-Silo} federated learning. In *2020 USENIX annual technical conference (USENIX ATC 20)* (2020), pp. 493–506.
- [58] ZHANG, R., ISOLA, P., EFROS, A. A., SHECHTMAN, E., AND WANG, O. The unreasonable effectiveness of deep features as a perceptual metric, 2018.
- [59] ZHANG, Z., CAO, X., JIA, J., AND GONG, N. Z. Fldetector: Defending federated learning against model poisoning attacks via detecting malicious clients, 2022.
- [60] ZHAO, B., MOPURI, K. R., AND BILEN, H. idlg: Improved deep leakage from gradients, 2020.
- [61] ZHAO, L., HU, S., WANG, Q., JIANG, J., SHEN, C., LUO, X., AND HU, P. Shielding collaborative learning: Mitigating poisoning attacks through client-side detection, 2020.

Appendix

1. Details of Experiment Setup

All experiments have been conducted using the PyTorch framework. The backdoor attack codes and all the existing defenses’ codes except FLTrust [10], FLAME [33] and AFA [32] were directly used from implementations by Xie et al. [53]. The TLFA and IPMA attacks, FLTrust, FLAME, and AFA defenses have been re-implemented for comparison purposes.

1.1. Datasets. We evaluate the performance of our defense on classification tasks trained on four datasets of whom three are image datasets (F-MNIST [52], E-MNIST [13], CIFAR10 [27]) and the rest is a tabular dataset (LOAN [17]).

1.2. Attacks Considered for FLShield Evaluation. We first describe the specific attack strategies for each and then provide an overview of the attack parameters used in our experiments.

Inner Product Manipulation Attack [54]. This attack assumes the adversary has access to the benign updates of the FL system. Specifically if the benign updates are w_i^t , for $i \in [c_t n + 1, n]$ at iteration t then the malicious clients send the following as the poisoned update:

$$w_i^t = -\frac{\epsilon_{att}}{[c_t n + 1, n]} \sum_{j \in [c_t n + 1, n]} w_j^t \quad (8)$$

We set the attack strength ϵ_{att} to 1.0. To measure the performance of the attack, we use the MA of the target class as the metric.

Targeted Label Flipping Attack [45]. This attack aims to misclassify a specific class of samples (*source* class) to another class (*target* class). The malicious clients flip the

TABLE 5: Attack taxonomy

Poisoning Attacks	Untargeted Poisoning Attacks	Inner Product Manipulation Attack (IPMA) [54]		
	Targeted Poisoning Attacks	Targeted Label Flipping Attack (TLFA) [45]		
		Backdoor Attacks	Distributed Backdoor Attack (DBA) [53]	
			Edge-case Backdoor Attack (ECBA) [48]	
			Semantic Backdoor Attack (SBA) [2]	
Privacy Inference Attack	Gradient Inversion Attack (GIA) [16]			
FLShield-aware Attacks	FLShield-aware adaptive attack (FA-Adp)			
	FLShield-aware advanced attack (FA-Adv)			

TABLE 6: FL system parameters

Parameter	E-MNIST	F-MNIST	CIFAR-10	LOAN
# of Clients	100			51
# of Clients per Round	25		10	20
# of Rounds	150		400	300
# of Samples per Client	600		500	Variable
# of Classes	10			9
batch size	64			
learning rate	0.1			0.001

TABLE 7: Attack Parameters

Parameters*	F-MNIST	E-MNIST	CIFAR-10	LOAN
No. of Malicious Clients		40		20
PSPB		20		10
Attack iterations		[36, 150]		[201, 300]

* All parameters apply to TLFA, DBA, and IPMA except PSPB for IPMA

labels of the samples from the source class to the target class in their local dataset and train the malicious local model. The (source, target) class pairs for the datasets F-MNIST, E-MNIST, CIFAR-10, and LOAN are (coat, shirt), (digit 5, digit 3), (automobile, truck), and (current, charged off) respectively.

Distributed Backdoor Attack [53]. In this attack, the adversary split the trigger pattern into multiple parts, and each client injects one of the partial triggers into a fraction of their training samples. For our experiments, we use the same trigger pattern as in [53], and we split the trigger pattern into 4 parts. The target class for the datasets F-MNIST, E-MNIST, CIFAR-10, and LOAN are pullover, digit 2, bird, and 'Does not meet the credit policy. Status: Charged Off' respectively.

Edge-case Backdoor Attack [48]. Edge-case backdoors are generated by altering label data points that, while usually correctly classified by the model, are under-represented, or unlikely to be part of the regular training or test data. We followed the experimental setup of [48]. The adversary uses

TABLE 8: Effectiveness of FLShield in comparison with state-of-the-art defenses against Inner Product Manipulation Attack

Defense	F-MNIST	CIFAR-10
	MA	MA
FedOracle	85.41	79.08
FedAvg	31.56	15.28
RFA	87.15	80.41
AFA	81.23	65.77
FLAME	86.59	79.62
FLTrust	86.03	80.07
FLShield*	85.33	80.76
FLShield†	85.13	80.76

images of the planes class from Southwest Airlines as edge-case samples and labels them as truck.

Semantic Backdoor Attack [2]. In this attack, the adversary aims to poison the model with the goal for it to produce an attacker-chosen output on visual semantic features (e.g. green cars). We followed the experiment setup of [2] and conduct the image classification task on CIFAR dataset. We selected the green car images as backdoor and labeled them as bird.

Gradient Inversion Attack [16]. An attacker aims to find the reconstructed samples x_r that minimize the loss function L_{grad} between the gradient from client k , $\nabla G_k(x_k)$, and the reconstructed gradient $\nabla G(x_r)$, according to the optimization problem $\underset{x_r}{\operatorname{argmin}}(L_{grad}(\nabla G(x_r), \nabla G_k(x_k))) + R_{aux}(x)$ where R_{aux} represents auxiliary knowledge used for regularization. We implement the attack proposed in [16] with [22]. The attack necessitates a smaller batch size for success, thus we utilize a batch of 16. We optimize the attack for 5,000 iterations using Adam, with an initial learning rate of 0.1.

1.3. FL Data Distribution. For image datasets, We consider an FL system containing 100 clients. For IID data distribution, we assign each client an equal number of samples from the training set. The distribution strategies for the two non-IID scenarios are described in section 6.2.2. For the LOAN dataset, we split it into 51 segments each corresponding to one of the states in the US. The *addr_state* attribute denotes the state where the loan applicant is from. This splitting mechanism provides a *natural* non-IID distribution.

1.4. Models. For CIFAR-10 and LOAN dataset, we use a lightweight Resnet-18 model [20] and a Soft Decision Tree following the implementation in [53]. For F-MNIST and E-MNIST, we train a standard convolutional neural network (CNN) as used in [53].

1.5. Experiment Parameters. The FL system parameters are summarized in Table 6. The attack parameters are given in Table 7. For FLShield*, the minimum and maximum number of clusters k_1 and k_2 are set to 2 and $\lfloor n/2 \rfloor$ respectively. For validation of FLShield, the minimum and maximum number of samples per class n_1 and n_2 are set to 10 and 30 respectively. For each client, we hold out 30% of its data for the validation task. However, the resulting reduction of the training data does not impact the performance at all as demonstrated in section 6.2.

Figure 12: Clustering-based Representative Generation Algorithm

Input: $\omega_1, \omega_2, \dots, \omega_n$ \triangleright local model updates,
 G_t \triangleright the global model
Output: m \triangleright the mapping of each client to a cluster,
 $\mathcal{E}_1, \dots, \mathcal{E}_m$ \triangleright the representative models of the m clusters
1: $\triangleright k_1$ and k_2 are the lower and upper bounds of the number of clusters respectively
2: $a_1, \dots, a_m \leftarrow \text{DynamicClustering}(\omega_{1:n}, k_1, k_2)$ $\triangleright a_i$ is a set containing indices of updates in cluster i
3: **for** each i in $[1, m]$ **do**
4: $\mathcal{E}_i \leftarrow G_t + \text{Mean}_{j \in a_i} \omega_j$
5: $m_j \leftarrow i \quad \forall j \in a_i$
6: **end for**
7: **return** $m, \mathcal{E}_1, \dots, \mathcal{E}_m$

Figure 13: Bijective Representative Generation Algorithm

Input: $\omega_1, \omega_2, \dots, \omega_n$ \triangleright local model updates,
 G \triangleright the global model
Output: $\mathcal{E}_1, \mathcal{E}_2, \dots, \mathcal{E}_n$ \triangleright the bijective representative models
1: **for** each i in $[1, n]$ **do**
2: $\mathcal{E} \leftarrow \text{BijectiveRepresentativeGen}(G_t, \omega, \tau)$
3: $\triangleright \text{BijectiveRepresentativeGen}(\cdot)$ uses equation 4 to calculate \mathcal{E}
4: **end for**
5: **return** $\mathcal{E}_1, \mathcal{E}_2, \dots, \mathcal{E}_n$

2. Additional Experiment Results

2.1. Performance Comparison between the use of different outlier detection algorithm in FLShield. We performed an experiment to compare the performance of three outlier detection algorithms - Local Outlier Factor, Isolation Forest, and Elliptic Envelope - in FLShield in all 3 different distribution scenarios. In all scenarios, the TPR is reported to be 100% and TNR is 86.67%. This demonstrates that the choice of outlier detection algorithm does not impact the TPR and TNR values.

3. FLShield-aware attack Implementation

3.1. FLShield-Aware Adaptive Attack (FA-Adp). Formally, the attacker solves the following optimization problem:

$$\begin{aligned}
 & \underset{l}{\operatorname{argmax}} (f_1(l) - \lambda_1 \times f_2(l) - \lambda_2 \times f_3(l)) \\
 & \mathcal{N}_{mal} = \{\mathcal{N}_j \mid j \in \mathcal{V}_m\} \\
 & f_1(\mathcal{N}_{mal}) = \sum_{x \in \mathcal{E}_m} \left| \min_{cl} \left(\text{Mean}_{i \in \mathcal{V}_m} (\mathcal{N}_i[x, :]) \right) \right|^2 \\
 & f_2(\mathcal{N}_{mal}) = \sum_{x \in \mathcal{E}_b} \left| \min_{cl} \left(\text{Mean}_{i \in \mathcal{V}_m} (\mathcal{N}_i[x, :]) \right) \right|^2 \\
 & f_3(\mathcal{N}_{mal}) = \sum_{i \in \mathcal{V}_m} |\mathcal{N}_i - \text{Mean}_{j \in \mathcal{V}_b} (\mathcal{N}_j)|
 \end{aligned} \tag{9}$$

In the above equation: \mathcal{E}_m is the set of malicious representative models, \mathcal{E}_b is the set of benign representative models, \mathcal{V}_m is the set of malicious validators, \mathcal{V}_b is the set of benign validators, λ_1 and λ_2 are the regularization parameters for the second and the third term respectively. f_1, f_2, f_3 are the three objectives of the optimization problem. f_1 (f_2) refers to the sum of the \mathcal{L} scores of the malicious (benign) representative models. Increasing f_1 and reducing f_2 achieves

Figure 14: Validation Algorithm

Input: $\{\mathcal{E}_1, \mathcal{E}_2, \dots, \mathcal{E}_m\}$ \triangleright representative models,
 G \triangleright global model,
 S \triangleright set of clients,
 c \triangleright number of classes
Output: $\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_m$ \triangleright \mathcal{M} report of each representative model
1: **for** i in $[1, m]$ **do**
2: randomly sample k validators $v_1, v_2, \dots, v_k \in S$
3: **for** v in v_1, v_2, \dots, v_k **do**
4: send both G and e to current validator
5: calculate $\mathcal{L}(\mathcal{E}, G, v)$ using Algorithm 15
6: send $\mathcal{L}(\mathcal{E}_i, G, v)$ back to the server
7: **end for**
8: **end for**
9: calculate $\mathcal{M}(\mathcal{E})$ for each representative model \mathcal{E} using equation (6)
10: calculate $\mathcal{N}(v)$ for each validator v using equation (6)
11: use *Imputation* to fill the missing values in each \mathcal{N} and \mathcal{M}
12: use *OutlierDetection* on all \mathcal{N} to filter out-of-distribution values
13: use remaining \mathcal{N} to update each \mathcal{M}
14: **return** $(\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_m)$

Figure 15: LIPC Calculation Algorithm

Input: e \triangleright the representative model to be validated,
 G \triangleright the global model at previous iteration,
 v \triangleright the validator
Output: \mathcal{L} \triangleright vector of the representative model
1: $D_1, D_2, \dots, D_c \leftarrow \text{SplitByClass}(D)$ $\triangleright D$ is validation dataset of v
2: **for** each i in $[1, c]$ **do**
3: **if** $|D_i| > n_1$ **then**
4: $D'_i \leftarrow \text{RandomSample}(D_i, \min(n_2, |D_i|))$ $\triangleright n_1$ and n_2 are the lower and upper bounds of the number of samples of each class respectively
5: $\mathcal{L}_i \leftarrow \text{Mean}_{(x,y) \in D'_i} (\text{Loss}(G(x), y)) - \text{Mean}_{(x,y) \in D'_i} (\text{Loss}(e(x), y))$
6: **else**
7: $\mathcal{L}_i \leftarrow nan$ $\triangleright nan$ (not-a-number) value assigned to the class with less than n_1 samples
8: **end if**
9: **end for**
10: **return** \mathcal{L}

the infiltration objective. f_3 refers to the sum of the absolute difference between the \mathcal{L} vector of each malicious validator and the mean \mathcal{L} vector of the benign validators. Reducing f_3 achieves the stealth objective.

The optimization problem is solved using the gradient descent algorithm, where the attack updates the \mathcal{N}_{mal} matrices by calculating the partial derivative according to the equation 9 repeating this process for 1000 iterations. During a grid search over the regularization parameters λ_1 and λ_2 , the attack simulates the steps of FLShield selecting the hyperparameter combination that maximizes the change in the \mathcal{L} score which is then used to craft the malicious \mathcal{N} matrices.

3.2. FLShield-aware Advanced Attack (FA-Adv). In this strategy, the adversary calculates the \mathcal{N} matrices that

Figure 16: Filtering Algorithm

Input: \mathcal{M} \triangleright matrices of the representative models
Output: \mathbb{I} \triangleright indices of the selected representative models
1: \triangleright calculate mean \mathcal{L} of all unfiltered validators for each \mathcal{M}
2: $\overline{\mathcal{M}} \leftarrow \text{Mean}_v(\mathcal{M}[v, :]) \quad \forall \mathcal{M}$
3: \triangleright extract minimum value from each $\overline{\mathcal{M}}$ and find the top 50% representative models based on the minimum
4: $\mathbb{I} \leftarrow \text{argsort}(\min(\overline{\mathcal{M}}_1, \dots, \overline{\mathcal{M}}_m))$
5: **return** $\mathbb{I}_1, \mathbb{I}_2, \dots, \mathbb{I}_{\lfloor n/2 \rfloor}$

Figure 17: FA-Adv Algorithm

Input: \mathcal{N} of all the benign validators
 V_m : malicious validators, V_b : benign validators, \mathcal{E}_m : malicious validators, \mathcal{E}_b : benign validators,
Output: \mathcal{N} for all malicious validators

- 1: $score_x \leftarrow \min_{cl} (\mathcal{E}_x) \forall x \in \mathcal{E}_m \cup \mathcal{E}_b$
- 2: $score_{x'} \leftarrow \min_{x \in \mathcal{E}_m \cup \mathcal{E}_b} (score_x)$
- 3: $x_1, x_2, \dots, x_m \leftarrow \text{argsort}(|score_x - score_{x'}|)$
- 4: **for** each x in $\{x_1, x_2, \dots, x_m\}$ **do**
- 5: $cl \leftarrow \underset{cl}{\text{argmin}} \left(\underset{v \in V_m \cup V_b}{\text{Mean}} (\mathcal{E}_x[v, :]) \right)$
- 6: **if** x is benign **then**
- 7: $y \leftarrow$ benign representative with the highest \mathcal{L} score that will be accepted
- 8: **else if** x is malicious **then**
- 9: $y \leftarrow$ malicious representative with the lowest \mathcal{L} score that will be filtered
- 10: **end if**
- 11: $cl' \leftarrow \underset{cl}{\text{argmin}} \left(\underset{v \in V_b}{\text{Mean}} (\mathcal{E}_x[v, :]) \right)$
- 12: $\mathcal{L}(x, v_m)[cl] \leftarrow \frac{\sum_{i \in V_b \cup V_m} \mathcal{L}(y, i)[cl'] - \sum_{i \in V_b} \mathcal{L}(x, i)[cl]}{|V_b|} \quad \forall v \in V_m$
- 13: **end for**
- 14: **return** (Updated \mathcal{N} for all malicious validators)

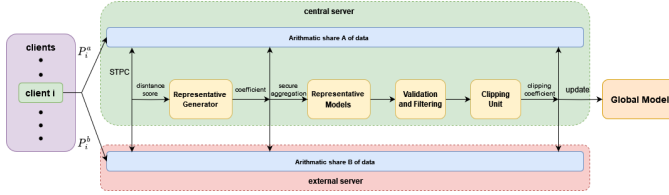


Figure 18: Server-side Privacy Strategy

meet the infiltration objective with the minimal alteration. We can safely assume that the \mathcal{L} score given to the malicious representative models is higher than the \mathcal{L} score given to the benign representative models. The malicious validators in this attack bridge the gap between the \mathcal{L} scores of the malicious and benign representative models by giving a lower score to the benign representative models in the class malicious representative models are underperforming and a higher score to the malicious representative models. We formalize this strategy in the Algorithm presented in Figure 17

4. Server-side Privacy

Existing Defenses. Existing defenses employ one of the three following defense strategies: differential privacy (DP) [21], [46], [49], homomorphic encryption [19], [57] and secure multi-party computation (SMC) [6], [7], [8], [46]. DP ensures privacy by adding noise to local updates but can compromise the model’s performance if too much noise is added. Homomorphic encryption allows the central server to compute encrypted data, but its practical application is limited by substantial computational overhead. SMC enables global model updates with aggregated local updates but is only effective when participants are honest. [47].

Our Defense. Inspired by FLAME [33], we utilize secure two-party computation (STPC) to prevent the threat from malicious central server. The overhead on the runtime

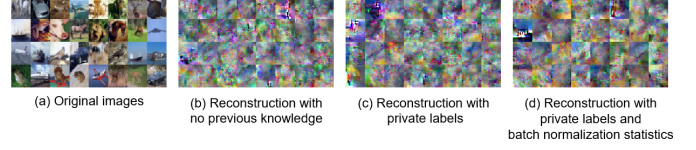


Figure 19: Images reconstructed from representative gradients in FLShield* when batch size is 32

caused by STPC is acceptable given it can help maintain privacy.

The idea is to prevent access to all local updates by a single central server. In order to achieve this goal, a third-party server called External Server is introduced. Figure 18 shows the pipeline of our method. In every training round, client $i \in \{1 \dots n\}$ first splits its update P_i into 2 arithmetic shares P_i^a and P_i^b , where $P_i = P_i^a + P_i^b$. Then, P_i^a and P_i^b are sent to the central server and the external server separately. This division of data allows both servers to possess partial access to the local updates. Once the central server and the external server receive the partial updates, they can employ STPC to collaboratively generate a representative model and send the generated model to the validators. Last, the servers can clip the local updates based on the output of the validators and update the global model together via STPC.

	None	Private labels	Private labels + BatchNorm Statistics
Attack on a single client’s gradient with batch size = 16			
Avg. LPIPS ↓	0.48	0.48	0.46
Best LPIPS ↓	0.13	0.12	0.09
LPIPS std.	0.12	0.13	0.14
Attack on cluster representative gradient with batch size = 16			
Avg. LPIPS ↓	0.61	0.61	0.61
Best LPIPS ↓	0.53	0.51	0.48
LPIPS std.	0.03	0.04	0.04
Attack on cluster representative gradient with batch size = 32			
Avg. LPIPS ↓	0.60	0.59	0.60
Best LPIPS ↓	0.46	0.50	0.45
LPIPS std.	0.03	0.03	0.04

TABLE 9: We evaluate the gradient inversion attack against cluster representative gradient in FLShield* on a subset of 50 CIFAR-10 images. (↓: lower values suggest more privacy leakage). The averaged and the best results for the metric of reconstruction quality are provided as average-case and worst-case privacy leakage.