Evaluating Security and Robustness for Split Federated Learning Against Poisoning Attacks

Xiaodong Wu, Henry Yuan, Xiangman Li, Jianbing Ni, *Senior Member, IEEE*, and Rongxing Lu, *Fellow, IEEE*

***Abstract*—Split federated learning (SFL) is a recently proposed distributed collaborative learning architecture that integrates federated learning (FL) with split learning (SL), offering an ingenious solution for safeguarding privacy in resource-limited environments. Despite the compelling potential of SFL and its appealing attributes, its robustness remains uncharted territory. In this paper, we investigate the security and robustness of SFL, with a specific focus on its susceptibility to malicious client-driven poisoning attacks. Specifically, we study the weaknesses of SFL against the well-known poisoning attacks designed for FL, like dataset poisoning, weight poisoning, and label poisoning. We also introduce a novel type of poisoning attacks tailored for SFL, named smash poisoning, and evaluate the robustness against smash poisoning attacks and advanced hybrid attacks (DatasetSmash, LabelSmash, and WeightSmash) that amalgamate smash poisoning with the other three methods for FL. By simulating these attacks across diverse domains over four datasets, we find that most of these attacks (including weight, WeightSmash, and LabelSmash poisoning) can disrupt the converged models with straightforward poisoning actions or have persistent negative influence on the model accuracy even after the termination of the attacks. Furthermore, our findings reveal that the robustness of SFL can be augmented by strategically adjusting the system parameters, such as client quantity, bottleneck size or split type. Finally, we verify the effectiveness of the typical defense mechanisms of poisoning attacks intended for FL and design a new defense strategy that filters out malicious smashed data to improve the robustness of SFL. We observe that the adoption of properly chosen defense mechanisms is beneficial in decreasing the security risks of SFL, but entirely eliminating the impacts of poisoning attacks in SFL is still challenging.**

***Index Terms*—Poisoning attack, split federated learning, AI robustness.**

Received 16 February 2024; revised 20 August 2024 and 13 October 2024; accepted 19 October 2024. Date of publication 4 November 2024; date of current version 23 December 2024. The associate editor coordinating the review of this article and approving it for publication was Dr. Ning Zhang. *(Corresponding author: Jianbing Ni.)*

Xiaodong Wu, Xiangman Li, and Jianbing Ni are with Department of Electrical and Computer Engineering and the Ingenuity Labs Research Institute, Queen's University, Kingston, ON K7L 3N6, Canada (e-mail: xiaodong.wu@queensu.ca; xiangman.li@queensu.ca; jianbing.ni@queensu.ca).

Henry Yuan is with the Department of Electrical and Computer Engineering, Queen's University, Kingston, ON K7L 3N6, Canada (e-mail: 19HY3@queensu.ca).

Rongxing Lu is with the Faculty of Computer Science, University of New Brunswick, Fredericton, NB E3B 5A3, Canada (e-mail: rlu1@unb.ca).

This article has supplementary downloadable material available at https://doi.org/10.1109/TIFS.2024.3490861, provided by the authors.

Digital Object Identifier 10.1109/TIFS.2024.3490861

I. Introduction

SPLIT federated learning (SFL) [1], seamlessly merges split learning (SL) [2] and federated learning (FL) [3], [4], to create a state-of-the-art learning architecture for distributed collaborative machine learning. SFL excels in scenarios where data is distributed across multiple clients, enabling localized training without the need for data sharing [5]. It culminates in the development of a highly effective and finely-tuned machine learning model. In the framework of SFL, the full model is partitioned into two components: one hosted by clients and the other controlled by a server, referred to as the main server. The flow of data samples occurs sequentially through the client models and the server models in the forward process, and the gradients move oppositely in the backward process. The main server is responsible for aggregating the server models once they are updated in each round, while the aggregation of the client models is overseen by an auxiliary server known as the fed server. The inherent strengths of SFL lie in its privacy-preserving attributes, inherited from FL [6] and computational resource efficiency brought by SL [2]. The safeguarding of data privacy is of paramount importance for sensitive domains, while alleviating the computational load on client-slide resources makes SFL suitable in resource-restriction environments. For example, in personalized healthcare, SFL emerges as a perfect solution for developing personalized prediction models based on the collected body measurements on individuals' smart phones for diagnosis and prognosis, while upholding stringent patient data confidentiality agreements.

Although SFL inherits great performance from FL and SL, there are legitimate concerns regarding the potential shared vulnerabilities [7], [8]. An exemplar pertains to the poisoning attacks [9], [10], a serious vulnerability known to compromise the robustness of FL [11]. This type of attack encompasses an array of techniques aiming at manipulating, compromising, or undermining the integrity of the training process and the resultant models [12]. In pursuit of distorting the global model's behavior during training, adversaries, who may compromise a small group of clients, can inject malicious data samples into the local datasets or manipulate their true labels [13], [14]. Thus, the models are learned from the corrupted data, resulting in biased predictions and a reduction in model performance. The scope of the attacks is also extended to the communication channels between the central server and the clients. By tampering with model updates during transmission, attackers might inject malicious model updates, thereby introducing vulnerabilities into the global model [15].

Apart from the adversarial attacks evaluated within the context of FL [16], [17], [18], there are several recent works investigating the vulnerabilities in SFL against poisoning attacks. As a pioneering work, Gajbhiye et al. [19] proposed to attack SFL with a label flipping strategy. Then, Ismail and Shukla [20] extended the traditional label flipping methods with a novel distance-based attack technique to conduct untargeted and targeted attacks on SFL. Moreover, Khan et al. [21] designed a model poisoning algorithm by adding the standard deviation perturbation to the models of the malicious clients with the assumption that attackers have the knowledge of the gradients of the benign client models. These works demonstrate that common poisoning attacks seriously threaten the security of SFL. The details of these related works are introduced in Appendix A. However, they merely apply the poisoning methods from the FL into SFL to investigate this problem, without exploring the attack performance of the sample manipulating based methods or designing methods particularly targeting the novel nature of SFL. Consider the potential risks brought from the vulnerabilities of SFL, it is of utmost importance to gain a profound understanding of the SFL system's robustness and reliability in practical implementations.

In this paper, we investigate the security and robustness of SFL, particularly focusing on the prevalent poisoning attacks in FL, as well as the new poisoning methods specifically tailored to SFL. Considering a scenario where potential attackers compromise a small portion of clients in the system, aiming to entirely destroy the training of the model (untargeted attack) or mislead the model into incorrect classification of samples in specific classes (targeted attack), we study the following four research questions pertaining to the robustness of SFL: **RQ1) Whether SFL is susceptible to the common poisoning attacks in FL, i.e., dataset poisoning, weight poisoning, and label poisoning?** Although recent research has assessed the robustness of FL against these attacks [22], [23], the performance of these attacks in SFL systems might differ significantly due to the import of SL. We find that all three types of poisoning attacks lead to a reduction in the accuracy of the trained global model in SFL under both IID and Non-IID data distribution. Notably, weight poisoning attacks cause a substantial decrease in the accuracy of the classification model by 84%, 77%, 47%, 19%, and 13% over MNIST, F-MNIST, CIFAR10, FEMNIST (IID), and FEMNIST (Non-IID) datasets, respectively. **RQ2) Is there any new poisoning attack method in SFL?** Considering the new data flow in SFL, i.e., the smashed data sent from clients to the main server, which does not exist in FL, we design a novel attack strategy named smash poisoning. This attack involves adding malicious perturbation into the smashed data on the client side before sending it to the server, subsequently exerting detrimental influence on the server model. We evaluate the accuracy degradation of the SFL systems due to smash poisoning attacks. **RQ3) Is it possible to further enhance attack effects by combining smash poisoning with dataset poisoning, weight poisoning, or label poisoning?** We find that the integration of smash poisoning and label poisoning, named LabelSmash poisoning, results in a significant accuracy reduction of 85%, 77%, 49%, 23%, and 23% over MNIST, F-MNIST, CIFAR10, FEMNIST (IID), and FEMNIST (Non-IID) datasets, respectively, acquiring a similar attack effect to weight poisoning. Unfortunately, when considering the integration of smash poisoning with weight poisoning, referred to as "WeightSmash poisoning", or the one of smash poisoning with dataset poisoning, referred to as "DatasetSmash poisoning", no noticeable enhancement of attack influence is observed compared to the single poisoning methods. **RQ4) Can the defense methods in FL have consistent effects in preventing poisoning attacks in SFL?** We evaluate nine robust aggregation methods, ranging from traditional methods like *Krum* and *Median* to several recently proposed methods, *Sparsefed* [24], *FLTrust* [25], *DnC* [26], *DeepSight* [27], and *ShieldFL* [18] and find that although *Median*, *Krum*, and *DnC* are effective at mitigating all the poisoning attacks, the performance drop is still noticeable, while others predominantly demonstrate robustness only against specific attacks.

Furthermore, we provide a thorough and comprehensive assessment of SFL's robustness in various poisoning attack scenarios, encompassing dataset poisoning, weight poisoning, label poisoning, DatasetSmash poisoning, WeightSmash poisoning, and LabelSmash poisoning. Our assessment considers diverse dimensions, including the quantity and proportion of malicious clients, variation in split types, fluctuation in the bottleneck size, the temporal scope of adversaries' attacks, and differences in model structure. Our results yield several interesting findings. We find that the SFL system with a higher ratio of malicious clients or a greater total number of clients with a fixed attacker ratio is more vulnerable to the poisoning attacks, and as the number of the layers allocated to the client side increases, the attempt of attackers to achieve a successful attack becomes more challenging. Meanwhile, our experiments reveal that if the LabelSmash poisoning attack is confined solely to the initial phase of training, the subsequent untainted training cannot restore the poisoned model to its original state, but other poisoning attacks can be rectified through untainted training. If the weight, WeightSmash, or LabelSmash poisoning attacks are launched after the convergence of models, they exhibit the capacity to completely dismantle the converged models. Besides, we observe that the bottleneck injection, which is a critical technique in SL to reduce the communication burden, is beneficial to improve the robustness of SFL. Furthermore, it is found that a model's parameter complexity may be positively correlated with its robustness against weight-based poisoning. Finally, we investigate the target attacks against SFL utilizing the label poisoning method and identify that this method can lead the model to misclassify up to 20% of samples from the target class as belonging to the base class, with only a slight decrease in total accuracy.

To the best of our knowledge, this is the first comprehensive work that assesses the robustness of SFL systems against the poisoning attacks. Our code is released at https://github.com/xxxcuss/asdf456jkk. The main contributions can be summarized as follows.

- We propose a structured framework for poisoning attacks orchestrated by malicious clients in SFL systems, which consists of three standard poisoning attack methods inherited from FL, namely, dataset poisoning, weight poisoning, and label poisoning, and a new smash poisoning attack specifically designed for SFL.
- We assess the security and robustness of SFL against different types of poisoning attacks, including dataset poisoning, weight poisoning, label poisoning, smash poisoning, and the new hybrid attacks, i.e., DatasetSmash poisoning, WeightSmash poisoning, and LabelSmash poisoning across MNIST, F-MNIST, CIFAR10, and FEMNIST datasets.
- We systematically perform extensive experiments across a range of settings, including varying percentage of malicious clients, distinct split types, diverse bottleneck size of the smashed data, varying temporal scope of attacks, and different model structures to identify the relationship between SFL's robustness and these various settings in adversarial SFL.
- We evaluate the effectiveness of traditional and newly proposed defense methods designed for FL against three common poisoning attacks and the three hybrid poisoning attacks within the SFL paradigm, for finding feasible strategies to enhance SFL's robustness against poisoning attacks. We also propose a novel defense strategy specially for SFL, which provides valuable insights for the future development of defense.

II. Background

In this section, we introduce the background of SFL. The introduction of FL and SL is in Appendix B.

A. SFL Background

Split federated learning [1] is a new distributed machine learning architecture, which synergistically harnesses the strengths of both FL and SL. They demonstrated that SFL can effectively reduce the computing pressure on client devices and improve the efficiency of the training process, while preserving the privacy and security of the personal data. Due to these appealing properties, SFL is suitable to be applied in resource-restricted and privacy-sensitive environments. There are two variations of SFL proposed in their work: SFLV1 and SFLV2. The details of SFLV1 are shown in Alg. 1. Assume there are m clients in the system, where each client has its local data, the same model architecture, and the optimizer for training the local part of the full model, i.e., the client model. There are two servers in the system. One is the fed server that is in charge of aggregating the parameters of the client models and the other is the main server that is responsible for training the rest part of the model, i.e., the server model, for each client in parallel and aggregating the parameters of the server models. This two-server architecture was proposed by Thapa et al. [1]. The reason for introducing a second server rather than using a single server for both aggregation tasks, is to preserve the privacy of the full model. By preventing the main server from accessing client updates, this SFL system reduces the risk of a secure yet curious main server inferring the client-side model and raw data. This design lowers the vulnerability to model inversion and model extraction attacks, as the main server only has access to the smashed data sent from clients.

Algorithm 1 Core of SFLV1 (SFL)

Input: Client's data samples $X = \{X^1, \ldots, X^i, \ldots, X^m\}$ with labels $Y = \{Y^1, \ldots, Y^i, \ldots, Y^m\}$, clients index set $C = \{1, 2, \ldots, m\}$, learning rate η, server model at round $t-1$: $\Theta_{s,t-1} = \{\theta^1_{s,t-1}, \ldots, \theta^i_{s,t-1}, \ldots, \theta^m_{s,t-1}\}$, client model at round $t-1$: $\Theta_{c,t-1} = \{\theta^1_{c,t-1}, \ldots, \theta^i_{c,t-1}, \ldots, \theta^m_{c,t-1}\}$.

Output: Server model and client model $\Theta_{s,t}$ and $\Theta^i_{c,t}$ for $i = 1$ to m at round t

```
 1: for each client $i \in C$ in parallel do
 2:     Samples forward in client models: $V^i = M^i_c(X^i; \theta^i_{c,t-1})$;
 3:     The client sends $V^i$ and $Y^i$ to the main server;
 4:     Samples forward in server models: $p^i_{t-1} = M^i_s(V^i; \theta^i_{s,t-1})$;
 5:     The main server calculates loss: $l^i_{t-1} = \mathcal{L}(p^i_{t-1}, Y^i)$;
 6:     In back-propagation, the main server calculates gradients $\frac{\partial l^i_{t-1}}{\partial \theta^i_{s,t-1}}$ and $\frac{\partial l^i_{t-1}}{\partial V^i}$;
 7:     The main server updates server models: $\theta^i_{s,t} \leftarrow \theta^i_{s,t-1} - \eta \frac{\partial l^i_{t-1}}{\partial \theta^i_{s,t-1}}$;
 8:     if all server models are updated in this round then
 9:         The main server aggregates server models and uses the obtained global model to replace each server model: $\theta^i_{s,t} = Agg(\Theta_{s,t})$ for $i = 1$ to $m$
10:     end if
11:     The main server sends the gradient $\frac{\partial l^i_{t-1}}{\partial V^i}$ to the client, where $\frac{\partial l^i_{t-1}}{\partial \theta^i_{c,t-1}}$ is calculated based on the chain rule;
12:     Clients update their local models: $\theta^i_{c,t} \leftarrow \theta^i_{c,t-1} - \eta \frac{\partial l^i_{t-1}}{\partial \theta^i_{c,t-1}}$;
13:     Clients send their local models $\theta^i_{c,t}$ to the fed server
14:     if all client models are received by the fed server then
15:         The fed server aggregates local models and uses the obtained global model to replace each client model: $\theta^i_{c,t} = Agg(\Theta_{c,t})$ for $i = 1$ to $m$.
16:     end if
17: end for
```

In each round, every client feeds the **local data $\mathbf{X^i}$** into the client model M^i_c (line 2) and then sends the corresponding middle output $\mathbf{V^i}$ (**smashed data**) and its corresponding **label $\mathbf{Y^i}$** to the main server (line 3), which uses these outputs and labels to update the server-side models (lines 3-7). When all server models are updated, the main server aggregates all server models (line 9) and backwards the gradients to each client (line 11). After that, every client updates its client model using the obtained gradients (line 12) and sends the **local model $\boldsymbol{\theta^i_{c,t}}$** to the fed server (line 13). Finally, the fed server aggregates all client-side models and updates each client model with a new global model (line 15). Compared to SFLV1, in SFLV2, the aggregation of the server models is removed. Instead, the server model is updated each time a client's smashed data is sent to the main server for processing. As outlined in [19], its robustness is worse than that of

SFLV1. This is because, in SFLV1, multiple server models are trained in parallel, which protects the server models of the benign clients, while in SFLV2, only one server model is trained sequentially. In our work, we choose SFLV1 as our system architecture to assess the robustness of the best SFL architecture so far against poisoning attacks.

III. Methodology

In this work, we propose four attacking methods to evaluate the security and robustness of SFL, including **Dataset Poisoning**, **Weight Poisoning**, **Smash Poisoning**, and **Label Poisoning**. These methods encompass tasks of both targeted and untargeted attacks. We present the threat model and problem formulation and introduce our attack methods.

A. Threat Model

Our threat model considers a scenario where a malicious attacker has the ability to compromise a small portion of clients participating in an SFL task. The goal of the attacker is to either significantly degrade the performance of the model, i.e., untargeted attacks, or lead the model to misclassify the samples belonging to specific classes into predefined classes, i.e., targeted attacks. In the untargeted attack, the attacker's strategy should be effective in undermining the model's performance once the attacker starts to poison data samples, labels, or model parameters. Furthermore, it should be resistant to recovery. Even if the compromised clients are detected and removed during the training process, mitigating the resulting damage within the trained model should remain a challenging endeavor. In contrast, in the targeted attack, the objective of the attacker is to manipulate the final model to misclassify the samples in specific classes, without compromising the overall accuracy.

The capabilities of the attackers are constrained by their knowledge of the SFL systems, which can be roughly classified into three levels. In the lowest level, the attackers only know the label information, including the range of labels, the base class, and the target class (for targeted attacks) with no knowledge of the gradients. They cannot access the gradients from the inputs and outputs of the model or the model itself. Under such circumstances, the attackers can conduct **Label Poisoning** which modifies the original labels with the purpose of compromising the final model. In the second level, the attackers can acquire the additional gradient information of the inputs and outputs on the client side, i.e., the gradients of the training samples and the smashed data. However, the parameters and the corresponding gradients of the models on the client side or the server side are still unknown. In this context, they can design **Dataset Poisoning** and **Smash Poisoning**, which modify the data points and smashed data, respectively. In the last level, the attackers are also assumed to have access to the gradient information of the model. In addition to the previous three attack methods, they are capable of deploying **Weight Poisoning**, which aims to manipulate the weights of the client models.

The attackers' capabilities are determined based on their attack strategies. In the **Label Poisoning** attack, the attackers are capable of changing the true labels of the training data owned by their controlled clients. They can either alter all labels or a portion of labels to the target classes. In the **Dataset Poisoning**, the attackers have access to the training samples from the client models. They have abundant computing resources to compute the modifications based on the partially available gradients and generate new samples before feeding them into the client model. Similarly, in the **Smash Poisoning**, the attackers can manipulate the smashed data based on the gradient information before inputting it into the server model. In the **Weight Poisoning**, the attackers have the capabilities to calculate and craft fake weights, but they are only allowed to compromise the weights of the malicious clients. In all cases, the attackers can control their clients to interact with the main server multiple times, so that these attacks are applicable in an asynchronous SFL system.

Algorithm 2 Overall Structure

Input: Every client's dataset D^i with samples X^i and labels Y^i, client index set $C = \{1, 2, \ldots, m\}$, learning rate η, attacking rate λ, attacker index a, attack iteration number R, maximum epoch T

Output: Trained model g_s, g_c

```
1:  Initialize all client models $\Theta_c = \{\theta^1_{c,0}, \theta^2_{c,0}, \ldots, \theta^m_{c,0}\}$ and
    server models $\Theta_s = \{\theta^1_{s,0}, \theta^2_{s,0}, \ldots, \theta^m_{s,0}\}$;
2:  Initial gradients of attacker's data and smashed data with
    all zero vectors $A_0$: $\nabla_d, \nabla_s \leftarrow A_0, A_0$;
3:  Set attacker's data in the beginning as the original sam-
    ples: $\hat{X}^a_0 \leftarrow X^a$;
4:  for $t = 1$ to $T$ do
5:    for each client $i \in C$ in parallel do
6:      if $i \neq a$ then
7:        SFL: $\theta^i_{s,t}, \theta^i_{c,t} = SFL(X^i, Y^i, \theta^i_{c,t-1}, \theta^i_{s,t-1}, \eta)$;
8:      else
9:        $\theta^a_{s,t}, \theta^a_{c,t}, \nabla_d, \nabla_s =$
          $PoiAtt(\hat{X}^a_{t-1}, Y^a, \nabla_d, \nabla_s, \theta^a_{c,t-1}, \theta^a_{s,t-1}, \eta, \lambda)$;
10:     end if
11:     Collect all server and client models: $\Theta_{s,t} =$
        $\{\theta^1_{s,t}, \theta^2_{s,t}, \ldots, \theta^m_{s,t}\}$, $\Theta_{c,t} = \{\theta^1_{c,t}, \theta^2_{c,t}, \ldots, \theta^m_{c,t}\}$;
12:   end for
13:   Server model aggregation: $g_{s,t} = ServerAgg(\Theta_{s,t})$;
14:   Server model assignment: $\theta^i_{s,t} \leftarrow g_{s,t}$, for $i \in C$;
15:   Client model aggregation: $g_{c,t} = ClientAgg(\Theta_{c,t})$;
16:   Server model assignment: $\theta^i_{c,t} \leftarrow g_{c,t}$, for $i \in C$;
17: end for
```

B. Problem Formulation

Let $C = \{c_1, c_2, \ldots, c_m\}$ be the set of clients in the SFL system, where m is the total number of clients. The main server holds the server models $\Theta_s = \{\theta^1_s, \theta^2_s, \ldots, \theta^m_s\}$ and returns the gradients upon query, and the fed server aggregates the client models $\Theta_c = \{\theta^1_c, \theta^2_c, \ldots, \theta^m_c\}$. Each client c_i possesses a local dataset D^i, including a set of training data samples and labels $\{x^i_j, y^i_j\}$. In the forwarding process, D^i is fed into a local client model M^i_c with the parameter θ^i_c for every client, and the outputs of this model are the inputs of the corresponding server

Algorithm 3 Poisoning Attacks (*PoiAtt*)

```
Input: Malicious client $a$'s dataset $\hat{X}^a_{t-1}$ with true labels $Y^a$ at epoch $t$, gradients of attacker's data and smashed data $\triangledown_d, \triangledown_s$, Client $a$'s server model and client model $\theta^a_{s,t-1}, \theta^a_{c,t-1}$ at round $t-1$, learning rate $\eta$, attacking rate $\lambda$, maximum attacking iteration $R$
Output: Client $a$'s server model and client model $\theta^a_{s,t}, \theta^a_{c,t}$ at round $t$, updated gradients of attacker's data and smashed data $\triangledown_d, \triangledown_s$
1: Initial attacker's data: $\hat{X}^a_t \leftarrow \hat{X}^a_{t-1}$;
2: for $r = 1$ to $R$ do
3:     Craft poisoned data: $\hat{X}^a_t \leftarrow \hat{X}^a_t + \lambda \triangledown_d$;
4:     Client models forward: $\hat{V}^a = M^a_c(\hat{X}^a_t; \theta^a_{c,t-1})$;
5:     Craft poisoned smashed data: $\hat{V}^a \leftarrow \hat{V}^a + \lambda \triangledown_s$;
6:     Server models forward: $p^a_t = M^a_s(\hat{V}^a; \theta^a_{s,t-1})$;
7:     if Perform label poisoning then
8:         Craft fake labels: $\hat{Y}^a \leftarrow (\hat{Y}^a + 1) \mod \mathcal{Y}$, or replace the target classes with the base classes;
9:         Calculate loss with fake labels: $l^a_{t-1} = \mathcal{L}(p^a, \hat{Y}^a)$;
10:    else
11:        Calculate loss with true labels: $l^a_{t-1} = \mathcal{L}(p^a, Y^a)$;
12:    end if
13:    Back-propagation, calculate gradients: $\frac{\partial l^a_{t-1}}{\partial \theta^a_{s,t-1}}, \frac{\partial l^a_{t-1}}{\partial \hat{V}^a}, \frac{\partial l^a_{t-1}}{\partial \theta^a_{c,t-1}}$, and $\frac{\partial l^a_{t-1}}{\partial \hat{X}^a_t}$;
14:    if Perform dataset poisoning then
15:        Update data poison vectors: $\triangledown_d \leftarrow \triangledown_d + \frac{\partial l^a_{t-1}}{\partial \hat{X}^a_t}$;
16:    end if
17:    if Perform smash poisoning then
18:        Update smash poison vectors: $\triangledown_s \leftarrow \triangledown_s + \frac{\partial l^a_{t-1}}{\partial \hat{V}^a}$;
19:    end if
20:    if Perform weight poisoning then
21:        Update client models with poisoned weights: $\theta^a_{c,t-1} \leftarrow \theta^a_{c,t-1} + \lambda \frac{\partial l^a_{t-1}}{\partial \theta^a_{c,t-1}}$;
22:    else
23:        Update client models: $\theta^a_{c,t-1} \leftarrow \theta^a_{c,t-1} - \eta \frac{\partial l^a_{t-1}}{\partial \theta^a_{c,t-1}}$;
24:    end if
25:    Update server models: $\theta^a_{s,t-1} \leftarrow \theta^a_{s,t-1} - \eta \frac{\partial l^a_{t-1}}{\partial \theta^a_{s,t-1}}$;
26: end for
27: Obtain the final client model: $\theta^a_{c,t} \leftarrow \theta^a_{c,t-1}$;
28: Obtain the final server model: $\theta^a_{s,t} \leftarrow \theta^a_{s,t-1}$.
```

model M^i_s with the parameter θ^i_s. After the calculation of the loss with true labels, in the backward process, the gradients are computed and propagated from M^i_s to M^i_c to update their parameters, completing an iteration of training.

The overall problem can be formulated as the following objective function:

$$\min_{\Theta_s, \Theta_c} \sum_{i,j} \mathcal{L}(M^i_s(V^i; \theta^i_s), y^i_j), \tag{1}$$

where i and j are the indexes of the client and his/her local training sample, respectively. $\mathcal{L}$ is the training loss function, e.g., cross entropy or hinge loss, and V^i is the smashed data, which is the intermediate output from the client model M^i_c, i.e., $V^i = M^i_c(x^i_j; \theta^i_c)$.

An attacker's objective function is defined as:

$$\min_{\hat{X}^a, \hat{Y}^a, \hat{V}^a, \hat{\theta}^a_c} \mathcal{L}_{attack}\left(D_{target}; \arg\min_{\Theta_s, \hat{\Theta}_c} \sum_{i,j} \mathcal{L}(M^i_s(V^i; \theta^i_s), y^i_j)\right). \tag{2}$$

Similar to other poisoning attacks, an attacker needs to control the malicious client $a \in C$ by modifying its data sample $\hat{X}^a$, data label $\hat{Y}^a$, smashed data $\hat{V}^a$, or model parameter $\hat{\theta}^a_c$ to minimize the attack loss $\mathcal{L}_{attack}$. These four types of corruptions correspond to **dataset poisoning**, **label poisoning**, **smash poisoning**, and **weight poisoning**, respectively. Besides, the attack loss $\mathcal{L}_{attack}$ is defined as:

$$\mathcal{L}_{attack}(D_{target}, \Theta) = -\sum_{(x,y) \in D_{target}} \mathcal{L}_\Theta(x, y), \tag{3}$$

where x and y are samples and labels in the target dataset D_{target}, respectively, and Θ stands for the full model consisting of the client model Θ_c and the server model Θ_s. The test dataset is usually denoted as D_{target}.

C. Attack Design

We introduce our framework of the poisoning attacks in the SFL system, as shown in Fig. 1. There are m clients and two servers, i.e., the fed server and the main server, in the system, aiming to train a well-functioning model with local training on their individual private data without directly sharing them with the public. Both servers are benign and secure, and assume that a maximum of c clients are compromised. The overall training process of this framework is illustrated in Alg. 2. Each client first performs local training on their device and then sends the smashed data to the main server for inference (line 7), as well as sending the local client model to the fed server for aggregation (line 11). During training, each client accesses local training samples X^i, corresponding labels Y^i, the smashed data, and local model updates θ^i_c. If a client is malicious, it can manipulate these types of data by injecting false information, leading to four types of attacks, as shown in Alg. 3. Specifically, before local model training, an attacker can craft poisoned samples and the corresponding labels to manipulate the generated local model, which are referred to as **dataset poisoning** and **label poisoning** attacks, respectively. Then, in the forward phase, before sending the smashed data to the main server, an attacker may modify it with the purpose of corrupting the full model. This type of attack is known as **smash poisoning**. In the backward phase, the gradients returned from the main server may be utilized maliciously to construct poisoned weights for model updates, resulting in a **weight poisoning** attack.

1) Smash Poisoning: The smash poisoning method is targeted on the special structure of SFL. Instead of submitting original data to the server in centralized training scenarios, clients transmit the smashed data from their local models to the main server in this system, thereby rendering it susceptible to exploitation by the attackers. By minimizing the attack loss of the optimization-based attack algorithm, as shown in Eq. 2 based on the gradient ascent algorithm inspired by FGSM [28], we obtain the results, as given in Alg. 3, line 18, as:

$$\hat{V}^a_t \leftarrow \hat{V}^a_t + \lambda \frac{\partial l^a_{t-1}}{\partial \hat{V}^a}, \tag{4}$$

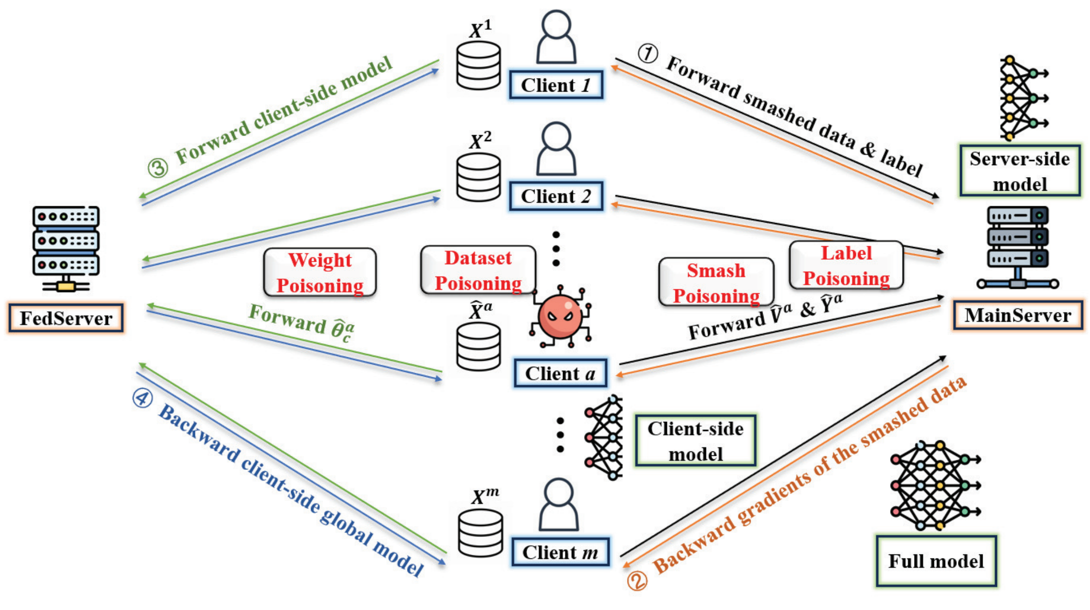

X^1
Client 1
X^2
Client 2
$\hat{X}^a$
Client a
X^m
Client m
FedServer
Server-side model
MainServer
Full model
Client-side model
③ Forward client-side model
Weight Poisoning
Dataset Poisoning
Smash Poisoning
Label Poisoning
Forward $\hat{\theta}_c^a$
Forward $\hat{V}^a$ & $\hat{Y}^a$
① Forward smashed data & label
④ Backward client-side global model
② Backward gradients of the smashed data

Fig. 1. An overview of the framework for the poisoning attacks in SFL.

where $\hat{V}_t^a$ denotes the smashed data sent from the attacker a to the main server, l_{t-1}^a is the loss function at the round $t-1$, and λ is the attacking rate, which functions similarly to the learning rate. In this algorithm, the smashed data is considered as an additional variable, which can be utilized to calculate the corresponding gradients. In each attack iteration, after obtaining the smashed data, the attacker adds the gradients multiplied by the attacking rate to the smashed data, thereby diminishing the performance of the SFL system. It is important to emphasize that, since the smashed data entirely depends on the computation involving the original data and the model's parameters, both of which are not poisoned if only smash poisoning is applied, any manipulation applied to the smashed data cannot be persist into the next iteration. As a result, the negative effects of this manipulation do not carry over from the previous iteration, thereby limiting the overall impact of smash poisoning when applied independently.

2) Dataset Poisoning: This attack aims at poisoning the training dataset by inserting new crafted samples or manipulating the original ones with the purpose of reducing the accuracy of the trained model. Here, we use gradient information to generate fake samples. Specifically, we treat the training samples as variables akin to the parameters of the model, for obtaining the gradients of these samples. The gradients are then used to minimize the attacking loss through optimizing the training samples in the direction of gradient ascent. Formally, the computation of the malicious data samples is illustrated in Alg. 3, line 15, as:

$$\hat{X}_t^a \leftarrow \hat{X}_t^a + \lambda \frac{\partial l_{t-1}^a}{\partial \hat{X}_t^a}, \tag{5}$$

where $\hat{X}_t^a$ represents the training samples possessed by the attacker at the round t. As shown in Alg. 3 line 2, the attack iteration number R determines how many times that the modified training samples are fed into the models to generate new updates. In an asynchronous system, this interaction can enhance the quality of optimization. The dataset poisoning attack initially corrupts the client models controlled by the attacker and then propagates the negative effects during aggregation of client and server models within the two servers.

3) Weight Poisoning: Weight poisoning is targeted on directly corrupting the model parameters. In the traditional setting, the attackers have full access to the models, enabling them to identify the most vulnerable positions, e.g, layers or neurons, to insert the poisoned weights. However, in the SFL system, the main server holds a part of the model and is assumed to be genuine, so that the attacker can only modify the client models, the extent of which depends on how models are split.

Here, we utilize a simple but effective weight poisoning method based on the gradient ascent algorithm. The poisoning of the client model's parameter is shown in Alg. 3, line 21, as:

$$\theta_{c,t-1}^a \leftarrow \theta_{c,t-1}^a + \lambda \frac{\partial l_{t-1}^a}{\partial \theta_{c,t-1}^a}, \tag{6}$$

where $\theta_{c,t-1}^a$ denotes the model of the malicious client a at the round $t-1$. In each attack iteration, the parameter vector of this model is moved to the opposite direction of the normal training in the parameter space. This poisoning initiates a faster impact on the client models compared to the dataset poisoning. As a result, it has the potential to induce a more rapid collapse of the system.

4) Label Poisoning: Label poisoning is the most direct and simplest method to poison a machine learning model. To significantly degrade the performance of the trained model, the attacker replaces the legitimate labels of data samples belonging to the compromised clients with the ones that it carefully chooses or randomly selects. These clients corrupt the training process when the main server collects the labels from them to compute the training loss. In our label poisoning, we utilize the modulo rotate calculation to craft the poisoning labels, which are generated as:

$$\hat{Y}^a \leftarrow (\hat{Y}^a + 1) \mod \mathcal{Y}, \tag{7}$$

where $\hat{Y}^a$ represents the manipulated labels in the dataset of the malicious clients and $\mathcal{Y}$ denotes the number of the class categories, e.g., 10 in the MNIST dataset. By assigning incorrect labels to partial data samples, the training system is perplexed due to the completely different learning directions provided by the malicious and genuine clients. As a result, challenges arise in achieving convergence and maintaining proficient performance during evaluation.

Moreover, given the attackers' capability to assign any legitimate labels to their local training samples, targeted attacks become feasible, which are unachievable with the other three methods. The reason lies in that in these methods the true labels are provided, so the attackers need to follow the gradient optimization direction or completely oppose it. However, in the targeted attacks, the attackers aim to impact the classification of only a subset of the classes, requiring adherence to gradients for that subset, while opposing the remaining classes. Only label poisoning can be deployed to achieve targeted attacks, because it enables the attackers to determine the optimization direction for every single class. For example, to make models misclassify samples in class 0 as class 1, the attacker can assign class 1 to the samples belonging to class 0. In training, the system learns to associate the samples in class 0 with class 1, resulting in the misclassification of the samples from both classes into class 1. This result aligns with the objective of a conventional targeted attack.

5) Hybrid Poisoning Attacks: As the smash poisoning and other three attack methods (dataset poisoning, label poisoning, and weight poisoning) are self-contained and independent of each other, an attacker is able to employ multiple poisoning attacks, such as dataset poisoning and smash poisoning, in combination to achieve enhanced attack performance. Therefore, we consider hybrid poisoning attacks that integrate smash poisoning with dataset poisoning, label poisoning, or weight poisoning to obtain powerful attacks on the SFL systems:

- **DatasetSmash:** After constructing the malicious data samples according to equation 5, attackers also modify the smashed data output from their local model according to equation 4.
- **WeightSmash:** After modifying the parameters of their local model according to equation 6, attackers also modify the smashed data according to equation 4.
- **LabelSmash:** After flipping the label of specific samples according to equation 7, attackers also modify the smashed data according to equation 4.

We will check whether these combinations can generate more powerful attacks on the SFL systems and bring greater challenges for building robust SFL systems against these attacks.

IV. Experiments

In this section, we first introduce the datasets, the evaluation metrics we use, and the baseline methods we will compare against. Then, we illustrate the details of how we implement our proposed attack methods. Finally, we discuss the attack results of our proposed poisoning methods in SFL.

A. Datasets

We choose to evaluate the robustness of SFL on four image datasets: **MNIST**, **F-MNIST**, and **CIFAR10**, which were used in the paper proposing the SFL system [1], as well as **FEMNIST**, a large dataset with 62 different classes. These datasets represent image data in different domains. In [1], another dataset, i.e., HAM1000, is also tested. However, we exclude it from our experiments because it is highly unbalanced. Specifically, The top three categories in this dataset account for 66.9%, 11.1%, and 10.9%, respectively, while the remaining four categories collectively represent only 10.9%. The details of these datasets are provided in Appendix C.

B. Evaluation Metrics

In the untargeted attack, we evaluate the performance of our attack methods through the metric of classification accuracy on the test dataset. After completing the training process, we calculate the average accuracy under attack on all test samples. The detrimental impacts of the poisoning methods increase with a decrease in accuracy. Besides, in the targeted attack, the goal is to make the model misclassify samples in the target class as belonging to the base class. We evaluate the true positive rate (TPR) for the target class and the true negative rate (TNR) for the base class. TPR and TNR are defined as:

TPR = (True Positives) / (True Positives + False Negatives),

TNR = (True Negatives) / (True Negatives + False Positives).

For the target class, if TPR decreases after poisoning, it indicates a reduction in the correct classification of the samples belonging to the target class. For the base class, if TNR drops after attacking, there is an increase in the number of the samples incorrectly classified as the base class. To clarify further, we also calculate the wrong classification rate (WCR) defined as:

WCR = (#Target class samples predicted as the base class)/(#All target class samples).

C. Baseline Methods

For the untargeted attack task, we compare the performance of our four attack strategies and hybrid poisoning attacks against a state-of-the-art adversarial attack method designed for SFL systems: Khan et al. [21]. For the targeted attack task, we evaluate our label poisoning attack in comparison with a recently proposed method: Ismail [20].

Khan et al. [21] is a model poisoning method designed for analyzing the security of SFL. It assumes that the attacker has access to the gradient updates from the benign clients. By utilizing these gradients, the attacker applies standard deviation perturbation to the updates of malicious clients to do model poisoning.

Ismail and Shukla [20] is a novel distance-based label flipping method. It selects the class whose data distribution is the farthest from the group of the target class as the base

TABLE I
MODEL ACCURACY UNDER DIFFERENT POISONING METHODS ON MNIST, F-MNIST, CIFAR-10, AND FEMNIST

Dataset	Normal	Khan	Dataset	Weight	Smash	Label	DatasetSmash	WeightSmash	LabelSmash
MNIST	95.63	87.03	80.78	**10.52**	93.92	78.22	81.15	**11.34**	**10.39**
F-MNIST	87.61	74.43	75.19	**10.03**	87.11	67.88	76.31	**10.17**	**10.37**
CIFAR10	58.78	13.19	48.39	**11.97**	58.70	48.69	46.59	**11.77**	**10.00**
FEMNIST	77.50	65.64	72.04	58.37	73.39	61.02	63.76	56.90	**53.98**

class, and then conducts the targeted attack by flipping the label of the attacker's data from the target class to the base class.

D. Implementation Details

We build the classification model with ResNet-18 [29], which is applied in [1] to evaluate SFL. Its structure is shown in Fig. 4. For the layer splitting in our system, we set the type $L2$ as the default split. Specifically, $L2$ partitions the initial three convolution layers to the client model and allocates the remaining layers to the server model. The aggregation for the main server and the fed server is performed through the average operation by default. We use SGD as the optimizer with a learning rate $\eta = 0.01$ and an attacking rate $\lambda = 1$ to train the system. The implementation of the designed attack methods on SFL utilizes PyTorch on the RTX A6000 platform. The dataset is randomly split into multiple subsets of equal size with each subset assigned to a client. Unless otherwise stated, we assume an IID training data distribution among clients by default, with a total of five clients in the system, including one malicious client.

E. Evaluation Results

1) Attack Performance on Distinct Datasets: We first deploy the three common poisoning attacks, our proposed smash poisoning method and Khan's model poisoning attack on four datasets to evaluate the robustness of SFL systems. The results are shown in Tab. I. In columns 4-7, dataset poisoning, weight poisoning, smash poisoning, and label poisoning are applied separately. From Tab. I, it is found that Khan's attack is only effective on CIFAR10. When dealing with easier classification tasks like the ones on the MNIST and F-MNIST, it shows much weaker attack performance. Compared with their method, our strategies have varying impacts on decreasing the accuracy to different extents. Specifically, Dataset poisoning, smash poisoning, and label poisoning only have limited or few effects, but weight poisoning can have much stronger negative effects on the accuracy. It can even completely disrupt the entire system on MNIST, F-MNIST, and CIFAR10. This demonstrates that the SFL systems have well robustness against most traditional poisoning attack methods. The reason weight poisoning in particular can achieve an attacking success is that it can directly poison the models of all clients with a greater scaled corrupted model. This model can overwrite other client models during the aggregation period on the fed server.

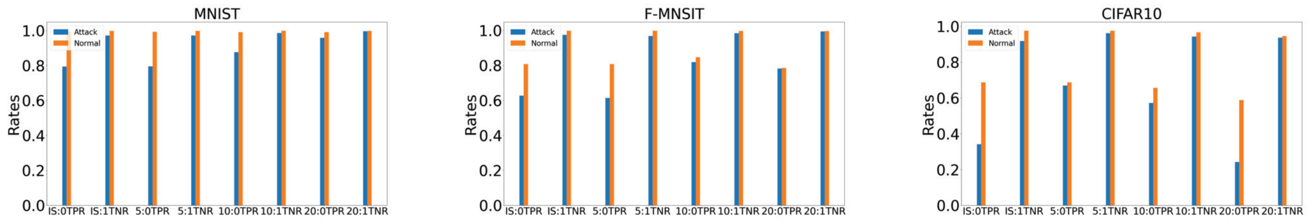
As previously mentioned in Section III-C, part 1), the effect of smash poisoning cannot be directly sustained within the system unlike dataset poisoning or weight poisoning, because the smashed data is completely dependent on the original data and the weight of the model, which are not manipulated in the smash poisoning. This explains why its attacking effect is relatively modest. Therefore, in most of the remaining experiments, this attack is not considered independently. Furthermore, we evaluate our three hybrid poisoning attack strategies, i.e., DatasetSmash, WeightSmash, and LabelSmash, to determine whether hybrid attack methods are more powerful or not. Their attacking results are given in the columns 8-10 of Tab. I. The attack performance of DatasetSmash and WeightSmash shows either a slight decrease (on FEMNIST) or remains relatively stable (on other datasets) when compared to dataset poisoning and weight poisoning. This suggests that the enhancement does not significantly impact the attack methods that are already very effective or ineffective. However, for label poisoning, the enhanced attack can reduce classification accuracy to a level of random guessing on most datasets, except FEMNIST. This illustrates that smash poisoning is able to improve attack performance when employed in combination with certain prevalent poisoning methods.

Address RQ1: The SFL system is susceptible to common poisoning attacks in FL, i.e., dataset poisoning, weight poisoning, and label poisoning. Among them, weight poisoning is the most powerful one, which can have a reduction of model's accuracy up to 85%.

Address RQ2: Considering the new data flow in SFL, we designed a new poisoning attack method called "smash poisoning", which adds malicious perturbations into the smashed data output from the client model. However, its performance when applied independently is not satisfactory.

Address RQ3: Combining smash poisoning with dataset poisoning, weight poisoning, and label poisoning can further enhance attack effects. This enhancement impacts the three attack strategies to varying degrees, with label poisoning seeing the most significant improvement. With the help of smash poisoning, it can reduce the model's accuracy to around 10%.

2) Attack Under Practical Settings: As recommended by Shejwalkar et al. [30], poisoning attacks in practical settings should be evaluated at relatively low poison rates, such as 0.1%. Following this approach, we reduced the poison rate from over 20% to below 0.1% and assessed the robustness of the SFL system against our poisoning methods on the FEMNIST dataset to simulate a real-world scenario. Specifically, we set the total number of clients to 1,020, with just one attacker among them. The results are presented in Tab. III. Compared to the scenario with a 20% poison rate (i.e., 68 attackers among 340 clients), the system demonstrates greater



(a) (b) (c)

Fig. 2. Targeted attack results about TPR for the target class and TNR for the base class prediction. (IS means under Ismail's attack [20]).

robustness at the lower poison rate of 0.1%, with the largest accuracy drop being only 6.15%, from 75.37% to 69.02%. This observation aligns with the findings in [30], which suggest that the average aggregation rule can achieve high accuracy even when there is only one compromised client.

3) Attack Under Non-IID Data Distribution: Using the sampling method provided in the official release of FEMNIST,[1] we created a Non-IID version of the dataset to simulate a scenario with explicitly Non-IID data distribution. The normal training performance and the SFL system's performance under different attacks are presented in Table IV. Compared to the IID scenario, the SFL system's accuracy decreases on Non-IID data, dropping from 77.50% to 73.89% when no attacks are present. However, the system's vulnerability under our proposed attacks remains similar to that observed with IID data distribution. This consistency indicates that our attacks are able to effectively degrade the SFL system's performance, regardless of the data distribution.

4) Targeted Attacks: We evaluate the vulnerability of SFL in targeted attacks. As mentioned in Section III-C, part 4), we apply label poisoning in this task. Specifically, in the local datasets of the malicious clients, true labels of samples belonging to the target class 0 are changed to the base class 1, aiming to mislead the system model. To evaluate robustness, we investigate the impact of varying the number of clients on TPR for the target class and TNR for the base class under three different settings. We also test the robustness of the SFL system under Ismail's attack [20], which is a distance-based label flipping targeted attack method, as the baseline with one malicious client in five clients. As depicted in Fig. 2, on three datasets, our label poisoning attack consistently decreases both TPR and TNR, indicating that after attacking, fewer samples with class 0 are correctly classified and more samples are wrongly predicted as class 1. Compared to our attack, Ismail's attack [20] causes a larger reduction on TPR and TNR on MNIST and CIFAR10, indicating improved performance. However, it is slightly worse than our label poisoning on F-MNIST.

To further check whether the decrease in TNR mainly results from the misclassification of class 0 samples, the wrong classification rate (WCR) is calculated and illustrated in Tab. V. It is shown that WCRs increase in the presence of malicious clients compared to the normal training, indicating successful targeted attacks. As a baseline, Ismail's attack [20] is generally better than our label poisoning attack. In comparison, Ismail's attack increases the WCR from 0.193 to 0.196 on MNIST and from 0.048 to 0.256 on CIFAR10. Besides, targeted attacks require a certain degree of stealth, such that their influence on the classification of other classes remains minimal. To quantify this, we compute the average accuracy of the model derived from the normal training and compare it with the accuracy obtained in the attack scenarios. In Tab. VI, the prediction accuracy of our attacks experiences acceptable reduction, which reveals the concealment and hazardousness of the targeted attack towards SFL systems. However, there is a noticeable decrease on the average accuracy of the model under Ismail's attack [20] (e.g., 8.84% on CIFAR10), compared to our attack (e.g., 3.29% on CIFAR10). This suggests that its enhanced effectiveness comes at the cost of reduced concealment.

5) Attack With Bottleneck Injection: We present the attack performance with varying bottleneck layer sizes in the Appendix E.

F. Ablation Study

1) Attack Performance With Different Number and Percentage of Compromised Clients: We investigate the scalability of SFL when a portion of clients are compromised. In SFL, the convergence and accuracy under massive clients are key performance indicators, which may be affected by the aggregation method, client set size, model size, etc. Here, we mainly explore the impacts that the number of total clients and the percentage of malicious clients have on the performance of SFL. As shown in Fig. 3, we first obtain the accuracy of models without malicious clients across a spectrum of client set size. With the increase of the number of clients, there is a slight decrease of the model accuracy, but the accuracy still remains high, meaning that SFL has the capability to accommodate a substantial number of clients while maintaining its operational efficacy. Then, we train the SFL system with the three common poisoning attack methods and their hybrid versions. The percentage of malicious clients is set to 20%. Similar to the previous results, weight poisoning, WeightSmash, and LabelSmash are able to destroy the system regardless of the number of clients. In terms of the remaining three methods, which exhibit limited negative effects on the system in scenarios with sparse compromised clients, the increase of the total number enlarges their capability in poisoning the system. These findings demonstrate that a

[1]https://github.com/TalwalkarLab/leaf

TABLE II
MODEL ACCURACY UNDER DIFFERENT POISONING METHODS WITH DISTINCT MALICIOUS CLIENT PERCENTAGE (20 CLIENTS IN TOTAL)

	Malicious Client Percentage (%)	Normal	Dataset	Weight	Label	DatasetSmash	WeightSmash	LabelSmash
MNIST	20%	94.89	74.19	**10.62**	71.09	74.37	**10.43**	**11.34**
	25%	94.89	77.33	**13.35**	63.16	75.68	**10.65**	**11.25**
	30%	94.89	74.08	**11.31**	63.72	74.44	**11.15**	**11.02**
	35%	94.89	77.88	**12.38**	60.2	75.88	**10.75**	**11.35**
	40%	94.89	77.15	**13.51**	58.34	74.05	**10.56**	**11.36**
	45%	94.89	72.9	**12.25**	48.92	75.28	**10.23**	**11.34**
	50%	94.89	72.19	**13.36**	43.08	74.79	**10.61**	**11.03**
F-MNIST	20%	85.86	83.93	**10.25**	63.28	82.98	53.94	**10.01**
	25%	85.86	83.86	**10.02**	62.55	83.71	47.43	**10.36**
	30%	85.86	83.66	**10.00**	53.56	83.92	40.11	**10.05**
	35%	85.86	84.10	**9.99**	48.81	83.73	39.60	**9.86**
	40%	85.86	83.62	**11.18**	45.80	83.37	33.98	**10.03**
	45%	85.86	83.45	**9.93**	43.58	83.35	16.79	**10.26**
	50%	85.86	83.40	**9.97**	39.67	83.26	**10.36**	**10.17**
CIFAR-10	20%	54.23	53.87	21.08	31.46	52.92	19.14	**10.14**
	25%	54.23	53.17	18.60	30.87	52.90	17.48	**10.28**
	30%	54.23	52.98	18.00	30.83	52.80	15.21	**9.99**
	35%	54.23	52.82	17.41	28.99	52.40	14.66	**10.01**
	40%	54.23	52.88	16.06	28.41	53.32	**13.79**	**10.09**
	45%	54.23	51.96	14.72	25.05	52.36	**13.37**	**10.00**
	50%	54.23	51.59	**10.74**	22.43	52.55	**13.38**	**9.79**

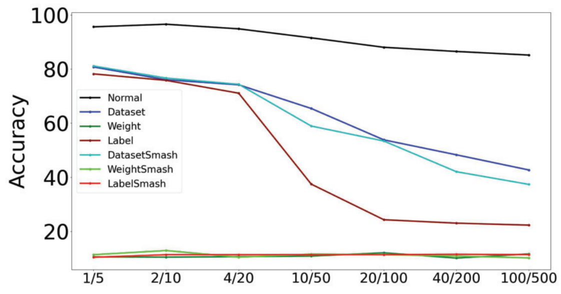


Fig. 3. Accuracy under distinct client number against different poisoning methods on MNIST. The attacker ratio (#malicious clients/#total clients) is set as the same, i.e., 20%. 'Normal' means all clients are genuine.

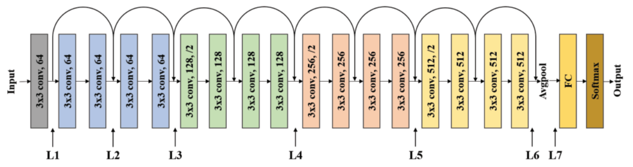


Fig. 4. Different splitting types in SFL with ResNet-18.

large number of clients can significantly amplify the potential security vulnerabilities of SFL systems.

We also investigate the impact of the percentage of malicious clients, while keeping the client set size fixed at 20 and show the results in Tab. II. Starting from 4 attackers in 20 clients, we increase the percentage up to 50% with a step size of 5%. In addition to the stable performance exhibited by weight, WeightSmash, and LabelSmash poisoning methods, a reduction in accuracy is observed when implementing label poisoning. This is because increasing the number of attackers results in more manipulation of labels, thereby creating more pronounced confusion for the model in the learning process. However, the performance decrease is invisible in the data manipulation, i.e., dataset poisoning or DatasetSmash, which is different from the attack results in the centralized training system, where dataset poisoning with 50% poisoning rate can destroy the trained model. The reason is that in centralized scenarios, the clean samples in the dataset get mixed with the poisoned ones, so that the misleading effects from the poisoned samples are persist throughout the entire training process. On the contrary, in SFL, the entire dataset is split into multiple subsets, hold by genuine and malicious clients. In each training epoch, some models are trained solely on the clean data, which helps alleviate the negative effects caused by the poisoned models. Therefore, SFL exhibits superior robustness performance against dataset poisoning compared to traditional training systems, owing to its distributed learning structure.

In addition, we assess the performance of the SFL systems under poisoning attacks with the fixed number of compromised clients, while the participating clients increase. We establish the scenario in which only one malicious client exists, varying the total number of clients from 5 to 10. As illustrated in Tab. VII, the performance of normal training with different numbers of clients is consistent. Similar to the previous results, weight poisoning, WeightSmash, and LabelSmash result in complete distortion of the classification. Moreover, for the remaining three methods: dataset poisoning, label poisoning, and DatasetSmash, there is an growing of the model accuracy with the increase of the total client number. This indicates that by enlarging the number of the clean clients in SFL systems, the effects of the poisoning attacks can be partially mitigated, which provides a way to enhance the robustness of SFL.

2) Attack With Different Split Types: In SL, one of the most key setting is determining how to partition the full model into client-side and server-side models. The decision has a significant influence over both the computational load on the clients and the system's convergence. Here, we analyze its effect on the vulnerability of SFL. Since we use Resnet-18 as the full model, we suggest dividing the model by segmenting residual layers. As shown in Fig. 4, there are

TABLE III
MODEL ACCURACY UNDER PRACTICAL SETTINGS ON FEMNIST

#clients	Normal	Dataset	Weight	Smash	Label	DatasetSmash	WeightSmash	LabelSmash
68/340	77.50	72.04	58.37	73.39	61.02	63.76	56.90	53.98
1/1020	75.37	72.80	70.21	69.23	72.35	72.38	69.02	69.20

TABLE IV
MODEL ACCURACY UNDER NON-IID DATA DISTRIBUTION ON FEMNIST

Distribution	Normal	Dataset	Weight	Smash	Label	DatasetSmash	WeightSmash	LabelSmash
IID	77.50	72.04	58.37	73.39	61.02	63.76	56.90	53.98
Non-IID	73.89	68.19	61.17	72.57	62.00	64.58	50.98	51.23

TABLE V
WCR FROM THE TARGET CLASS 0 TO THE BASE CLASS 1 UNDER TARGETED ATTACKS WITH DIFFERENT NUMBER OF CLIENTS (ATTACKER RATIO IS 20%)

#Client	MNIST	F-MNIST	CIFAR10
5 clients, 1 malicious, Ismail	0.196	0.171	0.256
5 clients, 1 malicious	0.193	0.191	0.048
5 clients, normal training	0.000	0.009	0.032
10 clients, 2 malicious	0.204	0.170	0.103
10 clients, normal training	0.000	0.003	0.039
20 clients, 4 malicious	0.203	0.190	0.198
20 clients, normal training	0.000	0.004	0.050

TABLE VI
AVERAGED ACCURACY UNDER TARGETED ATTACKS WITH DIFFERENT NUMBER OF CLIENTS (ATTACKER RATIO IS 20%)

#Client	MNIST	F-MNIST	CIFAR10
5 clients, 1 malicious, Ismail	93.34	83.24	53.16
5 clients, 1 malicious	94.81	86.18	58.71
5 clients, normal training	97.73	88.29	62.00
10 clients, 2 malicious	95.80	84.98	51.88
10 clients, normal training	97.61	87.93	57.42
20 clients, 4 malicious	92.68	79.39	48.33
20 clients, normal training	96.41	85.85	52.54

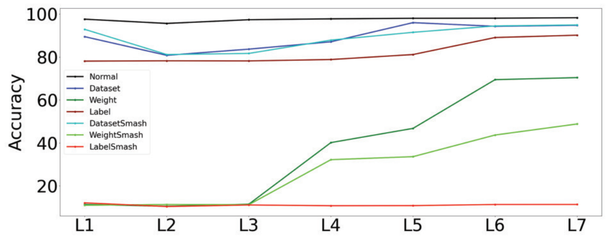


Fig. 5. Accuracy under different poisoning attack methods with distinct split types on MNIST.

7 different split plans, denoted as $L1$ to $L7$, each involving an increasing allocation of several layers to the clients. For example, in $L3$, the first 5 convolution layers are assigned to clients and the remaining layers are split to the server model. Additionally, the difference between $L6$ and $L7$ is that $L7$ incorporates the average pooling process into the client model. The trainable parameter size of the client side model for each split configuration is shown in Tab. VIII.

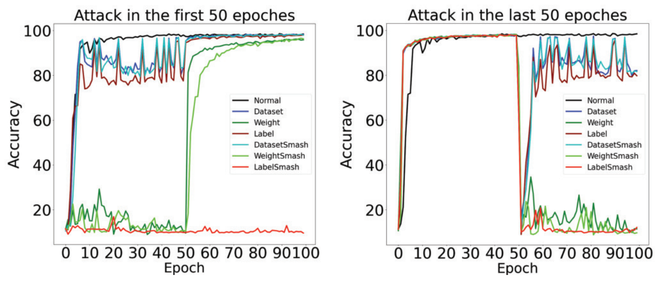
As illustrated in Fig. 5, we first present the performance of normal training with different split types (the black line). Each demonstrates high accuracy, showing the superior adaptability of SFL across diverse splitting configurations. Then, the proposed poisoning attacks are implemented with the same splitting configuration in SFL. It is observed that from $L1$ to $L7$, with an escalating number of layers allocated to the client side, weight poisoning and WeightSmash are the only two methods exhibiting sensitivity to the model split types. This phenomenon can be attributed to the substantial dependency of both algorithms on the specific conditions of model splitting. When assigning more layers to the clients, the system has better classification performance, indicating greater robustness against model poisoning methods. This outcome arises from the fact that an increasing number of layers brings a greater quantity of parameters available for manipulation, thus rendering the poisoning tasks more challenging for attackers. If the poisoning is confined to sparse layers alone, the attackers have the capability to completely eliminate the useful information within the parameters, thereby rendering the remaining model maintained by the server ineffective in its learning process. On the contrary, tampering with a greater number of parameters necessitates a substantially elevated degree of manipulation, imparting a heightened level of complexity to the model poisoning.

3) Attack Performance by Different Attacking Timing: In an attack-aware system, the operators have the knowledge about attackers' malicious behaviors and have deployed some effective detection mechanisms to protect the system. Once detecting the suspected behaviours of the outlier clients, the operators can remove them from the system or substitute them with new trusted clients. Therefore, it is meaningful to evaluate whether the system can seamlessly recover from its prior poisoned states and achieve a level of performance that is comparable to a system trained in a clean scenario from the scratch.

We assess the performance of SFL under the attacks of different poisoning methods happening in the first half period of training and show the results in Fig. 6 (a). During the attack phase, dataset poisoning, label poisoning, and DatasetSmash poisoning can achieve effective yet unstable reductions in the model's accuracy. In contrast, the remaining three methods severely compromise the system's performance. Then, if at the 50^{th} epoch, all malicious clients are identified and replaced with genuine ones, the system is trained in a completely reliable environment. It is shown that, after epoch 50, the damage caused by dataset poisoning, label poisoning, and

TABLE VII
MODEL ACCURACY UNDER DIFFERENT POISONING METHODS WITH INCREASING NUMBER OF CLIENTS (ONE MALICIOUS CLIENT) ON MNIST

#Client	Normal	Data	Weight	Label	DatasetSmash	WeightSmash	LabelSmash
5	95.63	80.78	**10.52**	78.22	81.15	**11.34**	**10.39**
6	97.63	80.92	**10.94**	81.86	83.22	**11.38**	**10.69**
7	97.43	83.18	**10.51**	82.61	84.36	**12.94**	**10.08**
8	96.78	83.63	**11.37**	82.84	84.85	**12.40**	**10.80**
9	97.47	83.96	**11.41**	84.71	85.53	**12.13**	**12.22**
10	96.59	89.08	**10.93**	85.64	86.21	**12.14**	**11.36**



(a) (b)

Fig. 6. Accuracy under different poisoning methods attacking only part of the full training period on MNIST (average over 3 runs).

TABLE VIII
THE TRAINABLE PARAMETER SIZE OF THE CLIENT SIDE MODEL FOR EACH SPLIT CONFIGURATION

Split Type	#Parameters
L1	3k
L2	77k
L3	152k
L4	679k
L5	2.7M
L6	11.1M
L7	11.1M

DatasetSmash poisoning are rapidly alleviated, and the detrimental effects resulting from weight poisoning and WeightSmash poisoning are ultimately eradicated. However, LabelSmash poisoning proves to be exceptionally destructive, as 50 epochs of subsequent normal training fail to restore the system to optimal functionality, underscoring the persistently disruptive consequences of this attack.

In an alternative scenario, the attackers may not conduct poisoning at the beginning of training, but waiting for a period of time in order to induce more abrupt and severe damage. This deliberate delayed attack pose a challenge to the robustness of SFL after the convergence of the model. We evaluate this by initiating poisoning attacks on the system after 50 epochs of normal training. In Fig. 6 (b), after the 50th epoch, all six poisoning methods yield similar attacking performance compared to the scenario where attacks are launched at the beginning of training. This indicates that the system, even after reaching a state of convergence and performing well, is still vulnerable to meticulously planned poisoning attacks.

4) Attack Under Different Model Structure: To evaluate the robustness of different neural networks against our proposed attacks, we replaced the original ResNet-18 model with ResNet-152 and VGG16. We assumed a scenario with 20 malicious attackers among 100 clients. The number of parameters in these models, along with their performance under our proposed attacks, is presented in Tab. IX. The results indicate that more complex models tend to achieve higher accuracy. Specifically, VGG16 demonstrates the highest accuracy and the best robustness, particularly against Weight and WeightSmash poisoning attacks, which significantly impact the other two models. This suggests that a model's parameter complexity may be positively correlated with its robustness against weight-based poisoning.

V. DEFENSE

We have demonstrated that the SFL system is vulnerable to both common poisoning algorithms and specially designed attack methods based on the structure of SFL. Nevertheless, these findings were acquired without any defense mechanisms. This section is devoted to discuss whether the current defense methods in FL can mitigate the detrimental impacts of the poisoning attacks in SFL.

A. Performance of Different Defense Mechanisms

In our work, we primarily focus on the robust aggregation mechanisms as the representatives of various defense mechanisms. We choose the defense algorithms from simple parameter number filters, e.g., median, to complex model vector filters like Krum. We also evaluate several state-of-the-art designs like Sparsefed [24], FLTrust [25], DnC [26], DeepSight [27], and ShieldFL [18]. These robust aggregation methods are briefly introduced in Appendix D.

In the first case, we apply these defense mechanisms only to the client models and show their effectiveness in Tab. X. DeepSight is not considered in this case because it requires the output from the last classification layer, which is available only in the server model. The classification performance under these mechanisms, as observed from normal training, is comparable to that achieved with average aggregation, indicating that they do not negatively impact the standard training process. Then, we apply our six poisoning methods to the system, along with the defense mechanisms, and record the average accuracy. Firstly, experiencing significant performance degradation caused by poisoning attacks, *Trim_mean* fails to provide effective safeguarding for the SFL system. This is because *Trim_mean* may trim some weights from the genuine clients, thereby increasing the potential risk of magnifying the effects of poisoning attacks. Secondly, *Median* and *Krum* demonstrate good defense performance under all types of attacks, due to

TABLE IX
ACCURACY UNDER DIFFERENT MODEL STRUCTURE ON MNIST. (#PARA REFERS TO THE NUMBER OF PARAMETERS IN THE MODEL)

Model Structure	#Para	Normal	Dataset	Weight	Label	DatasetSmash	WeightSmash	LabelSmash
ResNet-18	11.1m	88.06	53.82	12.05	24.28	53.39	11.35	11.05
ResNet-152	58.3m	91.19	63.68	25.04	58.47	62.67	12.65	11.16
VGG16	134.3m	91.36	83.25	90.80	72.54	81.01	83.07	15.57

the fact that they are designed to exclude the deviant updates and preserve the updates from genuine clients.

Thirdly, as an improved extension of *Krum*, *Bulyan* mitigates the negative influence of poisoning under three less effective attacks, i.e., dataset poisoning, label poisoning, and DatasetSmash, but it completely fails under the attacks of the remaining three methods. This failure indicates that the parameters poisoned by these three powerful attacks are capable of blending into the distribution of normal parameters, thereby remaining effective in the average-based defenses. However, the median-based mechanisms, such as *Median* and *Krum*, directly remove these poisoned parameters, effectively eliminating their impacts. Similar performance trends are observed in the evaluation of *Sparsefed*, which utilizes the top-k values extracted from the client updates for global model update. The results indicate that weight poisoning and WeightSmash can still leave poisoned weights in the top-k updates, leading to a decrease in final accuracy. However, compared to median-based methods, *Sparsefed* performs better under label poisoning and LabelSmash poisoning attacks.

Fourthly, for *FLTrust* and *ShieldFL*, both of which use cosine distance-based mechanisms, the defense performance is notably effective against dataset-related poisoning attacks. However, their performance deteriorates significantly against other attacks, particularly WeightSmash and LabelSmash. This suggests that alterations in sample data can cause substantial fluctuations in cosine similarity between the attacker's model and others, while changes in weights and labels have less impact. In contrast, as a singular value decomposition (SVD)-based mechanism, *DnC* exhibits superior defense performance across all six attack methods, underscoring the effectiveness and advantages of employing SVD for safeguarding SFL.

In the second case, the defense mechanisms are applied to both client models and server models to build a more secure scenario. The performance of the attacking methods is shown in Tab. XI. In general, adding aggregation defense on the server greatly improves the robustness of the SFL system with a maximum accuracy increase of 72.32%. Among these defense mechanisms, *DeepSight* performs exceptionally well against Dataset and DatasetSmash attacks but is less effective against other types. *ShieldFL*, on the other hand, shows significant improvement in defending against WeightSmash and LabelSmash attacks, making it a robust mechanism capable of defending against most of our poisoning attacks. Its performance is comparable to that of *Median*, *Krum*, and *DnC*. Besides, although being mitigated, Weight poisoning, WeightSmash poisoning, and LabelSmash poisoning still can have a leading attacking performance among all evaluated attacking strategies. In summary, by selecting appropriate robust aggregation algorithms and applying them to both the client and server side, SFL is capable of reducing the impacts that poisoning attacks impose, but complete elimination of these impacts might be challenging. Hence, there remains a need to investigate and develop new robust aggregation mechanisms for the SFL system.

Address RQ4: Defense methods in FL do not consistently prevent poisoning attacks in SFL. Among the nine robust aggregation methods we tested, only *Krum*, *Median*, *DnC*, and *ShieldFL* effectively mitigate all the aforementioned poisoning attacks to some extent. However, these methods still experience a noticeable drop in system performance when confronted with certain poisoning techniques, such as LabelSmash.

B. Defense Performance Under Different Number of Clients

Similar to Section IV-F, part 1), we also examine the scalability of the defense mechanisms in the SFL system. We choose to evaluate the performance of *Median* and *Krum* across varying numbers of clients on MNIST, since they are very effective in defending against attacks according to the results in Tab. XI. We assume that both client and server sides use the same defense mechanism, i.e., *Median* or *Krum*, with 20% of clients being malicious. The number of clients is scaled from 5 to 100. The results for *Median* and *Krum* are illustrated in Fig. 7 and Fig. 8, respectively. Both mechanisms show a decrease in accuracy with an increasing number of clients across all attack types. This trend confirms that *Median* and *Krum* effectively mitigate the adverse effects of poisoning attacks, regardless of the system's scale, as compared to the results shown in Fig. 3.

C. Defense Strategy for SFL Systems

The defense mechanisms discussed above are designed for traditional FL systems and may not fully address the unique characteristics of SFL. Since the main difference between FL and SFL is that clients send smashed data instead of model updates to the server, we design a defense strategy called **Smash Defense**, which filters out potential malicious clients based on the smashed data. In **Smash Defense**, during each epoch, after clients forward their original samples and send the smashed data to the main server, we apply an existing defense mechanism to the smashed data. This helps construct a trust set, which identifies trustworthy clients. Models from clients that are not included in the trust set are excluded during the model aggregation phase on both the client and server sides. The remaining models can then be processed using any suitable defense mechanism.

To evaluate the performance of our **Smash Defense**, we apply *DnC* [26] to the smashed data, given its superior effectiveness. Initially, we tested this approach without incorporating any additional defense methods. The results,

TABLE X
MODEL ACCURACY UNDER DISTINCT POISONING ATTACK METHODS WITH DIFFERENT DEFENSE MECHANISMS ON MNIST (ONLY CLIENT SIDE)

Defense Method	Normal	Dataset	Weight	Label	DatasetSmash	WeightSmash	LabelSmash
Median	97.64	62.03	85.82	66.22	62.85	78.46	62.11
Trim_mean	97.77	9.94	10.92	10.03	11.41	10.70	11.15
Krum	97.60	62.63	89.77	70.96	62.65	79.25	61.23
Bulyan	97.88	83.11	11.17	77.22	80.43	10.08	10.16
Sparsefed	97.85	62.37	9.66	81.92	62.06	10.63	56.28
FLTrust	96.85	88.67	47.85	58.84	85.94	11.35	31.68
DnC	97.24	92.54	85.60	80.93	90.82	80.61	80.46
ShieldFL	97.12	95.49	90.52	79.42	91.30	11.37	10.69

TABLE XI
MODEL ACCURACY UNDER DISTINCT POISONING ATTACK METHODS WITH DIFFERENT DEFENSE MECHANISMS ON MNIST (BOTH CLIENT AND SERVER)

Defense Method	Normal	Dataset	Weight	Label	DatasetSmash	WeightSmash	LabelSmash
Median	97.98	80.20	**97.35**	95.52	79.92	**96.10**	78.70
Trim_mean	86.53	79.27	19.91	59.67	76.72	10.27	50.91
Krum	97.68	92.28	90.69	87.07	87.97	86.81	85.18
Bulyan	98.08	**97.99**	26.07	77.27	**97.47**	25.77	41.79
Sparsefed	76.88	81.22	26.69	83.57	77.54	21.98	66.88
FLTrust	96.60	95.43	52.23	60.72	89.11	12.45	38.88
DnC	97.26	94.77	88.71	89.34	89.48	81.32	82.63
DeepSight	91.66	87.43	31.60	62.33	87.14	30.12	13.42
ShieldFL	97.24	95.56	91.70	87.60	93.10	76.26	83.01

TABLE XII
MODEL ACCURACY UNDER DISTINCT POISONING ATTACK METHODS WITH OUR SMASH DEFENSE MECHANISMS ON MNIST (BOTH CLIENT AND SERVER SIDE)

Defense Method	Normal	Dataset	Weight	Label	DatasetSmash	WeightSmash	LabelSmash
avg	95.63	80.78	10.52	78.22	81.15	11.34	10.39
DnC+avg	96.82	80.91	**87.63**	86.64	81.17	**80.34**	**80.33**
krum	97.68	92.28	90.69	87.07	87.97	86.81	85.18
DnC+krum	97.30	81.02	90.09	93.49	79.59	81.45	80.87
bulyan	98.08	97.99	26.07	77.27	97.47	25.77	41.79
DnC+bulyan	96.72	83.84	**91.37**	**92.58**	80.37	**82.86**	**82.61**
sparsefed	76.88	81.22	26.69	83.57	77.54	21.98	66.88
DnC+sparsefed	94.19	78.03	**90.54**	90.44	78.24	**88.82**	**80.48**
DnC	97.26	94.77	88.71	89.34	89.48	81.32	82.63
DnC+DnC	97.13	81.30	93.67	95.57	80.08	81.59	81.60

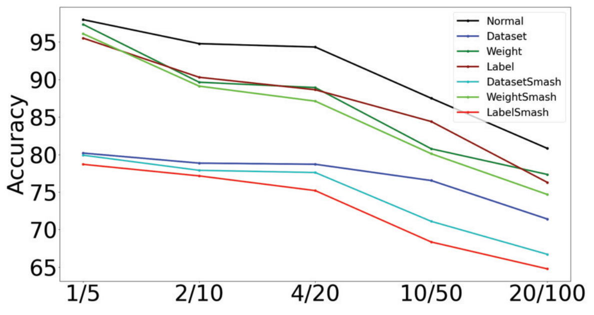


Fig. 7. Accuracy under varying client numbers with the *Median* mechanism applied.

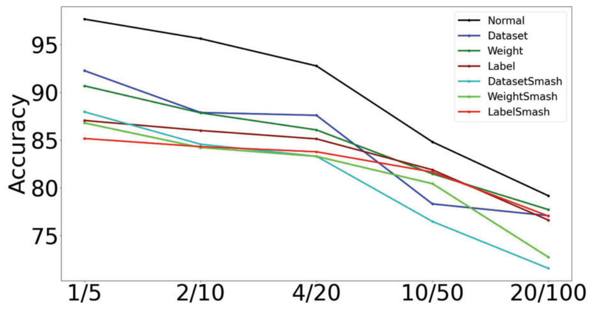


Fig. 8. Accuracy under varying client numbers with the *Krum* mechanism applied.

shown in Tab. XII, demonstrate that **Smash Defense** significantly enhances the SFL system's resilience against Weight, WeightSmash, and LabelSmash poisoning attacks. It increases accuracy from 10.52% to 87.63%, achieving an improvement of up to 77.11% compared to using only average aggregation. Furthermore, we explored the combined effectiveness of **Smash Defense** with other defense mechanisms, such as *Krum*, *Bulyan*, *SparseFed*, and *DnC*. As presented in Tab. XII, **Smash Defense** enhances the defensive performance of these methods against weight-based and label-based poisoning attacks. However, its effectiveness is reduced when addressing dataset-based poisoning attacks due to the stealthiness of these attacks. Overall, **Smash Defense** serves as a promising strategy, providing valuable insights and potential improvements for developing more robust defense mechanisms.

VI. Discussion

While our experimental results demonstrate the overall effectiveness of poisoning attacks in degrading the model's performance across various scenarios, there are notable limitations that must be considered. One such limitation is the reduced efficacy of these attacks at lower poisoning ratios, as highlighted in Section IV-E, part 2). At low poisoning ratios, the malicious data contained in the training set represents a smaller fraction of the entire dataset, providing less perturbation power to effectively influence the model. As a result, the attack has a diminished ability to manipulate the model's learning process, leading to weaker or less consistent degradation in performance. This limitation becomes particularly noticeable in scenarios where the available data is already large and diverse, making it difficult for a small amount of poisoned data to significantly shift the model's decision boundaries or internal representations. To address this limitation, potential attacks could be explored to improve the selection of poisoned data samples by identifying those that are more critical to the model's decision-making process, thus amplifying the impact of even a small number of poisoned points. Another attach approach is to develop more advanced perturbation techniques, such as adversarial poisoning, where an attacker could continuously adjust the poisoned samples during training to maximize their effect. Additionally, hybrid methods that combine poisoning attacks with other adversarial techniques, such as model inversion or data extraction, could further enhance the potency of poisoning attacks, even at lower ratios. These avenues of exploration could mitigate the current limitation and improve the effectiveness of poisoning attacks in future scenarios.

Additionally, while the evaluated attacks are effective in controlled experimental settings, their practicality in real-world applications requires further consideration. Specifically, executing a successful poisoning attack against split federated learning systems depends on various factors, including the attack ratio, the attacker's level of access to the target model, and the availability of data samples and intermediate smashed data. In real-world split federated learning systems, attackers may face significant barriers in gaining access to enough clients or the exact model parameters needed to implement a poisoning attack. Moreover, split federated learning often involves secure data-sharing protocols, encryption, and differential privacy techniques, which further complicate the attacker's ability to manipulate the model during training. Beyond data access, the computational resources required for large-scale poisoning present another significant hurdle. Attackers need substantial computing power to successfully execute poisoning attacks across multiple devices, making such attacks less practical for all but the most resource-rich adversaries. To enhance the practical applicability of these attacks, future research could examine attack scenarios under more realistic and constrained conditions. For instance, in mobile end-user application scenarios, attackers may face limited computational resources, making large-scale poisoning difficult to execute. Similarly, in highly confidential environments, attack opportunities may be further constrained by privacy requirements, such as limited attack ratios and tighter monitoring of data flow. Our work has demonstrated the effectiveness of poisoning attacks on the general image classification tasks; further research on adaptive poisoning strategies that operate within the resource and security limitations of real-world systems could help solidify the understanding of split federated learning's resilience to such threats and identify areas where defenses need strengthening.

VII. Conclusion

In this paper, we conducted a meticulous and comprehensive investigation into the security and robustness of the SFL system, offering novel insights into its vulnerabilities against both conventional and specially designed poisoning attacks. Specifically, we found that SFL is susceptible to three traditional poisoning techniques in FL: dataset poisoning, weight poisoning, and label poisoning. Additionally, we introduced a novel approach named smash poisoning, tailored to exploit the unique architecture of SFL and the hybrid attack methods, DatasetSmash, LabelSmash, and WeightSmash, by combining smash poisoning with traditional poisoning attacks. Our findings reveal that LabelSmash and WeightSmash exhibit the ability to accomplish their objectives by attacking only a fraction of the initial training process or by disrupting the stability of already converged models. Furthermore, we explored which elements of the SFL system are significant to its vulnerability. This is the first comprehensive work that assesses the robustness of SFL under different settings of split types and bottleneck size. Finally, we evaluated the effectiveness of defense methods originally developed for FL systems in countering poisoning attacks within SFL and introduced a new defense strategy specifically tailored for the SFL system. Our work sets the stage for advancing the security and reliability of SFL systems in the evolving landscape of distributed machine learning and opens new research problems, such as the analysis of theoretical performance bounds for SFL and the design of robust aggregation technologies for SFL.

References

[1] C. Thapa, P. C. M. Arachchige, S. Camtepe, and L. Sun, "SplitFed: When federated learning meets split learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 36, 2022, pp. 8485–8493.

[2] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar, "Split learning for health: Distributed deep learning without sharing raw patient data," 2018, *arXiv:1812.00564*.

[3] Y. Yang, B. Hui, H. Yuan, N. Gong, and Y. Cao, "PrivateFL: Accurate, differentially private federated learning via personalized data transformation," in *Proc. USENIX Secur.*, 2023, pp. 1595–1612.

[4] Y. Yang et al., "Fortifying federated learning against membership inference attacks via client-level input perturbation," in *Proc. 53rd Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2023, pp. 288–301.

[5] X. Liu, H. Li, G. Xu, Z. Chen, X. Huang, and R. Lu, "Privacy-enhanced federated learning against poisoning adversaries," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 4574–4588, 2021.

[6] K. Wei et al., "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 3454–3469, 2020.

[7] T. Chu, A. Garcia-Recuero, C. Iordanou, G. Smaragdakis, and N. Laoutaris, "Securing federated sensitive topic classification against poisoning attacks," 2022, *arXiv:2201.13086*.

[8] C. Zhou et al., "PPA: Preference profiling attack against federated learning," 2022, *arXiv:2202.04856*.

[9] W. J.-W. Tann and E.-C. Chang, "Poisoning online learning filters by shifting on the move," in *Proc. 53rd Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2023, pp. 239–251.

[10] N. Baracaldo et al., "Benchmarking the effect of poisoning defenses on the security and bias of deep learning models," in *Proc. IEEE Secur. Privacy Workshops (SPW)*, May 2023, pp. 45–56.

[11] H. Mozaffari, V. Shejwalkar, and A. Houmansadr, "Every vote counts: Ranking-based training of federated learning to resist poisoning attacks," in *Proc. USENIX Secur.*, 2023, pp. 1–19.

[12] Z. Yang et al., "Data poisoning attacks against multimodal encoders," in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 39299–39313.

[13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.

[14] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "XlNet: Generalized autoregressive pretraining for language understanding," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 5753–5763.

[15] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," 2012, *arXiv:1206.6389*.

[16] A. N. Bhagoji, S. Chakraborty, P. P. Mittal, and S. Calp, "Analyzing federated learning through an adversarial lens," in *Proc. 36th Int. Conf. Mach. Learn.*, May 2019, pp. 634–643.

[17] J. Zhang, J. Chen, D. Wu, B. Chen, and S. Yu, "Poisoning attack in federated learning using generative adversarial nets," in *Proc. 18th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun. 13th IEEE Int. Conf. Big Data Sci. Eng. (TrustCom/BigDataSE)*, Aug. 2019, pp. 374–380.

[18] Z. Ma, J. Ma, Y. Miao, Y. Li, and R. H. Deng, "ShieldFL: Mitigating model poisoning attacks in privacy-preserving federated learning," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 1639–1654, 2022.

[19] S. Gajbhiye, P. Singh, and S. Gupta, "Data poisoning attack by label flipping on splitfed learning," in *Proc. RTIP2R*, 2022, pp. 391–405.

[20] A. T. Z. Ismail and R. M. Shukla, "Analyzing the vulnerabilities in SplitFed learning: Assessing the robustness against data poisoning attacks," 2023, *arXiv:2307.03197*.

[21] M. A. Khan, V. Shejwalkar, A. Houmansadr, and F. M. Anwar, "Security analysis of SplitFed learning," in *Proc. 20th ACM Conf. Embedded Networked Sensor Syst.*, Nov. 2022, pp. 987–993.

[22] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," in *Proc. Eur. Symp. Res. Comput. Secur.*, 2020, pp. 480–501.

[23] C. Fung, C. J. M. Yoon, and I. Beschastnikh, "Mitigating sybils in federated learning poisoning," 2018, *arXiv:1808.04866*.

[24] A. Panda, S. Mahloujifar, A. N. Bhagoji, S. Chakraborty, and P. Mittal, "SparseFed: Mitigating model poisoning attacks in federated learning with sparsification," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2022, pp. 7587–7624.

[25] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "FLTrust: Byzantine-robust federated learning via trust bootstrapping," 2020, *arXiv:2012.13995*.

[26] V. Shejwalkar and A. Houmansadr, "Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2021, pp. 1–19.

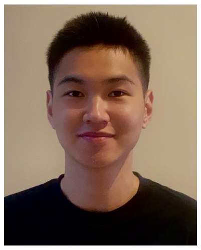
[27] P. Rieger, T. D. Nguyen, M. Miettinen, and A.-R. Sadeghi, "DeepSight: Mitigating backdoor attacks in federated learning through deep model inspection," 2022, *arXiv:2201.00763*.

[28] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, *arXiv:1412.6572*.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[30] V. Shejwalkar, A. Houmansadr, P. Kairouz, and D. Ramage, "Back to the drawing board: A critical evaluation of poisoning attacks on production federated learning," in *Proc. IEEE Symp. Security Privacy (SP)*, Oct. 2022, pp. 1354–1371.

Xiaodong Wu received the bachelor's degree in computer science from the University of Electronic Science and Technology of China and the master's degree in computer science from the King Abdullah University of Science and Technology. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, Queen's University, Kingston, Canada. His research interests include machine learning, adversarial attack, AI security and privacy issues, and trustworthy AI.



Henry Yuan received the bachelor's degree in computer engineering from Queen's University, Kingston, Canada, where he is currently pursuing the Master of Engineering (M.Eng.) degree with the Department of Electrical and Computer Engineering. His research interests include artificial intelligence (AI), privacy, and cybersecurity.

Xiangman Li received the bachelor's degree in applied science from Queen's University, Kingston, Canada, where she is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering. Her research interests include machine learning, AI security and privacy issues, and decentralized learning.

Jianbing Ni (Senior Member, IEEE) received the Ph.D. degree in electrical and computer engineering from the University of Waterloo, Waterloo, Canada, in 2018. He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, Queen's University, Kingston, Canada. His research interests include applied cryptography and network security, with current focus on edge computing, artificial intelligence, the Internet of Things, and blockchain technology.

Rongxing Lu (Fellow, IEEE) received the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Waterloo, Canada, in 2012. He was a Post-Doctoral Fellow with the University of Waterloo from May 2012 to April 2013. He was an Assistant Professor with the School of Electrical and Electronic Engineering, Nanyang Technological University (NTU), Singapore, from April 2013 to August 2016. He is currently a Mastercard IoT Research Chair, a University Research Scholar, and a Professor with the Faculty of Computer Science (FCS), University of New Brunswick (UNB), Canada. His research interests include applied cryptography, privacy enhancing technologies, and the IoT-big data security and privacy. He also serves as the Chair of the IEEE Communications and Information Security Technical Committee (ComSoc CISTC) and the Founding Co-Chair of the IEEE TEMS Blockchain and Distributed Ledgers Technologies Technical Committee (BDLT-TC).