BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 72(3), 2024, Article number: e148944
DOI: 10.24425/bpasts.2024.148944

ARTIFICIAL AND COMPUTATIONAL INTELLIGENCE

FedAssess: analysis for efficient communication and security algorithms over various federated learning frameworks and mitigation of label-flipping attack

R ANUSUYA [1]* and **D KARTHIKA RENUKA**

Department of Information Technology, PSG College of Technology, Coimbatore, TN 641004, India

Abstract. Federated learning is an upcoming concept used widely in distributed machine learning. Federated learning (FL) allows a large number of users to learn a single machine learning model together while the training data is stored on individual user devices. Nonetheless, federated learning lessens threats to data privacy. Based on iterative model averaging, our study suggests a feasible technique for the federated learning of deep networks with improved security and privacy. We also undertake a thorough empirical evaluation while taking various FL frameworks and averaging algorithms into consideration. Secure multi party computation, secure aggregation, and differential privacy are implemented to improve the security and privacy in a federated learning environment. In spite of advancements, concerns over privacy remain in FL, as the weights or parameters of a trained model may reveal private information about the data used for training. Our work demonstrates that FL can be prone to label-flipping attack and a novel method to prevent label-flipping attack has been proposed. We compare standard federated model aggregation and optimization methods, FedAvg and FedProx using benchmark data sets. Experiments are implemented in two different FL frameworks – Flower and PySyft and the results are analyzed. Our experiments confirm that classification accuracy increases in FL framework over a centralized model and the model performance is better after adding all the security and privacy algorithms. Our work has proved that deep learning models perform well in FL and also is secure.

Keywords: federated learning; privacy in federated learning; deep learning; attacks in federated learning; label-flipping attack in FL.

1. INTRODUCTION

Exponential development of machine learning (ML) and artificial intelligence have made voluminous innovations in several fields like smart cities, healthcare, aviation, agriculture. A huge volume of data is collected and processed every second by various applications. Traditional machine learning algorithms collect data from all devices like sensors and aggregate it for processing. However, as the development becomes prominent, certain challenges arise when handling big data. Difficulties such as limited infrastructure, poor network connection, bandwidth, compromised privacy [1] and security have been posing challenges for traditional ML models to handle huge volumes of data. Traditional ML model owners/developers also face challenges in collecting user's private and personal data as they possess sensitive information and must adhere to rules like General Data Protection Regulation (GDPR) and Personal Data Protection Act (PDPA). These limitations are making researchers and developers shift to decentralized machine learning framework. Federated learning (FL) is an evolving decentralized machine learning model. FL allows multiple clients/data owners to train data locally across devices, such as mobile phones and edge nodes instead of aggregating all the data in a central server. To update the global model, a central server receives just the model updates. The model update happens in multiple iterations. FL is beneficial in situations when data privacy is important, or where collecting and transmitting large amounts of data is infeasible [2].

FL [3, 4] provides privacy and security as data is not leaving the device and enhances efficiency as training across multiple devices happens parallelly. All these features have caused FL to be adopted in many applications like medical data diagnosis and mobile next word prediction. Though FL enhances security and privacy still it is prone to cyber-attacks [5]. There is no centralized control over the participating devices and hence the participating devices can behave maliciously. Due to this FL is prone to adverse machine learning attacks like poisoning attacks. Bearing in mind, the recent advancement in cyber attacks and data breaches over various phases of data processing, our work FedAssess focuses on utilizing the benefits of federated learning in a multi-user setup for maintaining data privacy during data analysis. The major contributions of our proposed work are:

- We propose a distributed and collaborative architecture that enables data owners to share private data for analysis while conserving privacy.
- Conducted extensive experiments to compare various FL frameworks.
- Preserving privacy and security algorithms in FL are demonstrated.
- Provided novel strategy to mitigate label-flipping attacks in FL.

*e-mail: raa.it@psgtech.ac.in

Manuscript submitted 2023-04-05, revised 2023-12-04, initially accepted for publication 2024-01-07, published in May 2024.

The paper is organized as follows: Section 2 details the background required for this paper. Literature review and gaps identified is given in Section 3. The algorithm of the proposed work, the security algorithms and optimization algorithms implemented in the paper are explained in Section 4. Data set description and the CNN architecture used for implementation is demonstrated in Section 5. Threat analysis for the implementation carried out against label-flipping attack and proposed strategy for mitigation against label-flipping attack is detailed in Section 6. Performance analysis of the work based on various evaluation metrics is discussed Section 7. Conclusion is given in Section 8.

2. BACKGROUND

2.1. Cloud-edge computing

Communication and computational issues develop when relying solely on a centralized cloud server for data processing. Edge computing (EC) can complement cloud servers to leverage the advantages of cloud computing. In edge computing, the processing is done in the edge devices like IoT sensors or in edge servers and only the results or meta-data of the processing is sent to cloud servers. The cloud-edge architecture is a distributed system, comparable to federated learning. Ongoing research has started integrating edge computing with federated learning to improve the efficiency and security of data processing. The challenges and methods for federated-edge deployment are discussed in the article by Guanming Bao and Ping Guo [6].

2.2. Federated learning

Federated learning environment can be used in a scenario where multiple and distributed data owners/devices want to jointly train a model without exchanging data like in IoT or healthcare environments. This ensures the privacy because the data stays local. The devices that have the data act as the client. If there are N clients C_1, C_2, …, C_N, then the local models M_1, M_2, M_3, …, M_N get trained in the client end over the data D_1, D_2, …, D_N. After the training process, the update from each local model is sent to the server. The server aggregates the updates to create a global model. The process can be defined as

$$M_{\text{global}} = FN\left(M_1, M_2, \ldots, M_N\right), \tag{1}$$

where FN represents the aggregation algorithms. Thus, we complete a single iteration of federated learning and disseminate the global model to each client device for further local training. The specific number of rounds is typically chosen by the model performance, meaning that we continue the procedure until the model can attain the desired level of accuracy. Furthermore, in order to enhance privacy protection, clients have the choice to apply encryption techniques to the models prior to uploading them. Differential privacy (DP) [7] and homomorphic encryption (HE) [8] are often employed strategies to ensure security in federated learning (FL). The effectiveness of federated learning is primarily determined by the aggregation technique employed on the server side. The primary objective of federated learning is to optimize the objective function. In conventional deep learning model training, for a training dataset containing n samples (x_i, y_i), $1 \le i \le n$, the training objective is:

$$\min_{w \in \mathbb{R}^d} f(w) \quad \text{where} \quad f(w) \overset{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} f_i(w), \tag{2}$$

$f_i(w) = l(x_i, y_i, w)$ is the loss of the prediction on example (x_i, y_i). Using SGD [9] optimization, the weight is updated with the following formula

$$w_{t+1} \leftarrow w_t - \eta \nabla f\left(w_t; x_k, y_k\right), \tag{3}$$

where η is the learning rate and $\nabla f(w_t; x_k, y_k)$ is the loss function derivative w.r.t weight. In federated learning setup, Suppose n training samples are distributed to K clients, where P_k is the set of indices of data points on client k, and $n_k = |P_k|$, the training objective would be $\min_{w \in \mathbb{R}^d} f(w)$

$$f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w) \quad \text{where} \tag{4}$$

$$F_k(w) \overset{\text{def}}{=} \frac{1}{n_k} \sum_{i \in P_k} f_i(w). \tag{5}$$

2.3. Attacks in federated learning

FL is protected against a wide range of attacks using defense mechanisms, which reduce the likelihood of hazards. Because data access is impossible, most of these defenses avoid model corruption by guaranteeing that the model has trained to realize the underlying statistical distribution of the actual training data. That does not rule out the possibility of fraudulent data or updates being used to train the model [10, 11].

Backdoor attack: The objective of backdoor attacks [12, 13] is to modify a section of training data by adding adversarial triggers so that deep neural network models provide inaccurate predictions when the same trigger occurs on the test set.

Gradients attack: A gradient attack in federated learning is a sort of adversarial assault in which a malicious client delivers corrupted or modified model updates (gradients) to the central server. This attack aims to damage the integrity and precision of the global model by injecting biassed or incorrect updates that can mislead it during training.

Model poison attack: Poison attacks [14, 15] aim to get the FL model to output the target label that the enemy has provided. Authors in [16] used a data poison attack, for instance, by switching the labels of training data from one class to another during the local training phase in order to deceive the output of the global model. Such type of poisoning attack is known as **label-flipping attack**.

The objective of an **untargeted poisoning attack** [17] is to produce corrupted local model updates that can be injected into the system, rendering the learned parameters of the global model effectively futile. Methods like differential privacy and zero knowledge proof [18] provide mitigation to few of the above mentioned attacks.

Table 1
FL frameworks comparison

	Flower:	FATE	Pysyft Framework:
Description	`The FLOWER framework facilitates a seamless transition specifically on a substantial batch of edge devices.`	`FATE is one of the earliest open-source frameworks for FL that is ready for commercial use.`	`PySyft is a free and open-source library for federated learning and privacy protection. It allows users to do deep learning in a private and safe manner.`
Flexible	No	No	No
Ease-of-use	Yes	No	No
Security features	No	No	Yes
Model support	No	Yes	No

2.4. Federated learning frameworks

Various frameworks like PySyft [19] and Flower [20], are available for implementing federated learning. The fundamental features of a federated learning framework are client-side training, server-side aggregation, and, notably, communication. Furthermore, it is imperative to have a local simulation mode. However, the frameworks might vary significantly in other aspects such as scalability, user-friendliness, security techniques, AI algorithms, and more. Table 1 compares the frameworks PySyft, Flower and FATE.

3. RELATED WORK

The authors of the study titled "Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent" [21] discuss the issue of guaranteeing the dependability of gradient descent algorithms when confronted with the presence of malevolent or Byzantine nodes. The deliberate manipulation of gradients by these nodes might undermine the efficacy of distributed machine learning systems, hence affecting the learning process. The authors propose technique (multi-Krum or MKrum) to make gradient descent resilient against Byzantine adversaries. The primary emphasis of this paper lies in the theoretical and algorithmic dimensions of Byzantine tolerance. However, it does not extensively explore the practical obstacles associated with implementing such systems in real-world scenarios. It supports only iid data.

FoolsGold (FGold) [22] discusses the incentives for participants to conduct Sybil attacks in federated learning, including competitive or adversarial motivations, and the potential consequences for the system. The paper discusses potential strategies to mitigate Sybil attacks, such as reputation-based mechanisms and secure aggregation. However, it may not provide a comprehensive evaluation of the effectiveness, trade-offs, and implementation challenges of these strategies.

Works based on aggregation such as Median [23] and Trimmed Median [23] and RMedian [24] against Poisoning Attacks are sensitive to outliers. These methods minimize the impact of bad updates on the global model. Median and Trimmed Median can face scalability issues in large-scale distributed systems. The RMedian algorithm incurs a significant processing overhead.

Singh, A.K. *et al.*, [25] attempted to find a balance in effort between preventing poisoning and embracing diversity to aid in the development of more equitable and nondiscriminatory federated learning models. The strategy to distinguish genuine from malicious updates provides models that are more accurate than those developed using typical poisoning detection techniques. Addressing the difficulty of identifying abnormal/malicious actions from valid ones in federated learning is presented in this paper. They concentrate on cases involving small groups of clients, who are likely to be labeled as outliers or malevolent by traditional attack detection algorithms provided in the literature.

Zhao, Y. *et al.*, [26] proposed a poisoning defense method that employs generative adversarial networks to create auditing data in the training phase and removes adversaries by auditing their model correctness to detect and neutralize poisoning attempts in federated learning.

Tolpegin, V. *et al.*, [16] investigated data poisoning attacks on FL systems. They showed that label-flipping poisoning assaults are vulnerable to FL systems and that these attacks can have a considerable negative impact on the global model. They also demonstrated that as the proportion of malicious individuals increases, the detrimental influence on the global model increases, and that targeted poisoning may be achieved. It is shown that adversaries can improve the effectiveness of attacks by increasing the number of malevolent participants available in later rounds. Finally, they proposed a defense to assist an FL aggregator in distinguishing between malicious and honest participants. It is shown that this defense can detect malevolent individuals and is resistant to gradient drift.

The research conducted by Zhuoran Ma *et al.*, (ShieldFL) [27] primarily addresses the issue of countering model poisoning attacks when the attacker uses encryption to mask the harmful local gradient modifications. It is an untargeted model poisoning attack. ShieldFL employs a two-trapdoor homomorphic encryption system with the cosine similarity method to effectively identify and eliminate probable poisoned updates. The work appears to be computationally costly.

Auror [28] cluster the group of clients into malicious and normal clients and disregard the malicious group. The study investigates the analysis of independent and identically distributed (iid) data, which results in increased occurrences of both false positives and false negatives when the data vary from being independent and identically distributed (non-iid). Furthermore, it is necessary to have prior understanding of the characteristics of the training data distribution or the expected quantity of attackers within the system. Moreover, the effectiveness of the defense mechanism is compromised in cases where the attack is a single-shot attack.

Gaps identified from the literature:
Huge datasets generated by multiple sources are used by ML and AI algorithms nowadays. Due to privacy concerns, the majority of the existing systems store data privately. In traditional ML involving multiple sources if data is not shared, the model training is less efficient due to limited data. If data is combined for collaborative training, privacy is a concern. Though FL helps to address this issue, it is also vulnerable to multiple attacks. There are many existing papers which provide solutions for attacks in FL, but most methods are computationally overhead or work only for iid data. Many papers provide security algorithms but do not provide threat analysis. A few papers on mitigation are only theoretical. Our work demonstrates image classification in FL setup with security and privacy algorithms which works robustly against label-flipping attack with less computation overhead.

4. PROPOSED METHODOLOGY

We propose a distributed framework named FedAssess that enables many clients to jointly train an image classification model without compromising privacy. Our goal is to foster image classification via federated learning with improved security and privacy. We show that a decentralized system can meet the requirements for data safety without compromising performance compared to a centralized machine learning system. The work also provides mitigation methods against label-flipping attacks (Fig. 1, Algorithm 1).

Algorithm 1 FedAssess

Input: Images for classification from multiple clients

```
1: The federated learning server transmits the
   global model to the clients.
2: Every client use the global model to train the
   local data.
3: The gradients updated are communicated to
   the client with security algorithms: secure
   aggregation, secure multi-party computation and
   differential privacy.
4: FedProx and FedAvg are utilised to implement
   client-to-server communication and parameter
   aggregation on the server.
5: The model is evaluated.
6: The model is tested against Model Poisoning
   attacks and mitigation algorithm is executed.
```

4.1. Secure multi-party computation

Secure multi-party computation (SMPC) [29] is a technique that guarantees the involvement of numerous parties in a computation, where each party contributes its own set of inputs and receives the desired outputs, while ensuring the confidentiality of everyone's data. SMPC is implemented using cryptography algorithms like SecureNN [30] and SPDZ [31]. In SMPC the data that is to be protected is split into Secret shares. The shares are distributed to all the participants involved in such a way that each one does not have an idea of the other participant's share. In federated learning setup for image classification, for enhanced privacy the model and the data have to be protected. Let us assume there are 'n' participants and a model that is shared by the server. Each of the 'n' client's data is split into 'n' secret shares and shared with all the other 'n-1' clients. The model is also split into 'n' shares and shared with all the 'n' clients. Each of them now owns their own shares, as well as one share of all the other users and one share of the model. Now, computation can begin in order to train the model confidentially using the necessary cryptography protocols. Once the model has been trained, all shares can be returned to the server for decryption.

4.2. Secure aggregation

Secure aggregation [32] approaches allow a group of individuals who do not trust each other to calculate the total sum of

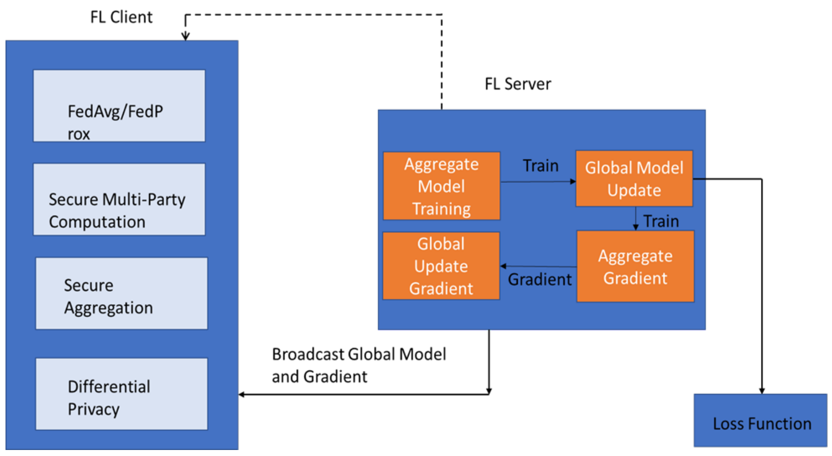

Fig. 1. Architecture of proposed FedAssess framework

their private values without disclosing the actual values. Secure aggregation for federated learning enhances federated learning by encrypting received weights before sending them to a central device. Secure aggregation prevents the server from identifying individual users' model but still being able to sum the updates from the model. Users mask the model with pair-wise random keys. Each pair of users also agrees on pair-wise seeds. The server combines the masked models obtained from the surviving users. The server then gets the secret shares of the dropped users' pairwise seeds and the secret shares of the surviving users' private seeds. Using the secret shares, the server reconstructs the pairwise and private seeds relating to dropped and surviving users and removes them from the sum of the masked models. The above process ensures that the central model has no access to the weights directly from the remote model, but only to the aggregated encrypted form. It adds an extra layer of privacy, protecting data on remote devices by preventing the central model from using weights to compromise privacy on remote devices.

4.3. Differential privacy

Differential privacy (DP) is a mechanism that enables researchers and database analysts to obtain useful information from databases holding the personal information of individuals while protecting their anonymity. This can be achieved by incorporating few interruptions into the database contents. The introduced interruption is both substantial enough to ensure privacy and limited enough to allow analysts to still obtain important information. In its simplest form, differential privacy anonymizes data by introducing noise into a dataset. It enables data specialists to conduct all statistical analysis without disclosing personally identifying information.

In differential privacy, noise is added to the data to preserve the privacy. It seeks to limit the impact of any individual's data on the final outcome. Differential privacy ensures that no inference can be made about any of the data from the final result, irrespective of whether any individual's information was included in the input for the analysis. DP gives a mathematically verifiable guarantee of privacy protection against a broad spectrum of privacy attacks.

While training machine learning models on various data sets, DP typically consists of two injections: clipping the maximum model parameter updates and adding noise to the model parameters. DP is an effective method for obscuring sensitive data. In deep learning, we employ neural network for prediction on user-provided input data. It is made up of multiple aspects, such as a model with architecture, loss value, weights, etc. The idea is to prevent the network from memorizing the data. Hence, when we wish to add noise to the data to make it DP, we can add it to the input data, the model training weights, and the loss function. The goal of privacy preserving data analysis is to obtain as much helpful information as is attainable while compromising privacy as little as possible. To formalize this concept, consider a database D, which is simply a set of data points, and a probabilistic function M acting on databases. The function is said to be (ε, δ)-differentially private if for all subsets of possible outputs $S \subseteq \mathrm{Range}(M)$, and for all pairs of databases D and D' that differ by one element,

$$\Pr[M(D) \in S] \leq \exp(\varepsilon)\Pr[M(D') \in S] + \delta, \tag{6}$$

ε denotes privacy leakage. Equation (6) shows that if one datapoint in the database is changed, the results of M will have a distribution that is essentially unchanged when both are very small positive values. In other words, adding a single user's data to a differential private analysis will not probably change the results.

4.4. Optimization algorithms in federated learning: FedAvg and FedProx

One of the important challenges normal federated learning framework faces is the communication cost as the model updates are sent in each round from all the participating rounds. There are a few communication optimization algortihms like FedAvg and FedProx.

4.4.1. FedAvg

Federated averaging (FedAvg) [33] can be used in a distributed environment with many clients. The data is kept locally in the participating device. The global model is distributed by the central server to k clients chosen at random. The selected clients train the model with the current state gradient w_t and updates the gradient parameter to w_{t+1} federated averaging (FedAvg) is a technique for distributed training with a large number of clients that is efficient in terms of communication. The communication rounds are lesser than in FedSGD(Aggregation model based on SGD) as the updates are not sent in each round to the server. FedAvg updates the global model by averaging the model updates from each client, whereas FedSGD updates the global model by averaging the gradients of each client's model. The pseudocode of FedAvg is given in Algorithm 2.

Algorithm 2 FedAvg

```
The K clients are indexed by k; B is the local mini-
batch size, E is the number of local epochs and $\eta$ is
the learning rate.
**Server: Initialize** $w_0$
**for** each round t=1,2,.. **do**
    m <- max(C.K,1)
    $S_t$ <- *(random set of m clients)*
    **for** each client k $\in s_t$ in parallel **do**
        Update $w^k_{t+1}$ <- ClientUpdate(k, $w_t$);
        $m_t \leftarrow \sum_{k\in S_t} n_k$
        $w_{t+1} \leftarrow \sum_{k\in S_t} \frac{n_k}{m_t} w^k_{t+1}$
    **end for**
**end for**
ClientUpdate(k, w):
$\mathcal{B} \leftarrow$ ( split $\mathcal{P}_k$ into batches of size $B$)
**for** each local epoch $i$ from 1 to $E$ **do**
    **for** batch $b \in \mathcal{B}$ **do**
        $w \leftarrow w - \eta\nabla\ell(w;b)$
    **end for**
**end for**
return $w$ to server
```

4.4.2. FedProx

FedProx [34] can be viewed as a generalization and re-parameterization of FedAvg. FedAvg is a federated learning algorithm that computes the average of model parameters from multiple clients. FedProx, on the other hand, is an optimization method that adds a proximal term to the loss function in FedAvg to improve the privacy and communication efficiency of federated learning. Although it makes only minor modifications to the re-parameterization method, these modifications have important implications both in theory and in practice. In FedProx variable amount of work and iteration is used. To address heterogeneity [17], along with local loss term a proximal term is also used. In FedProx, the proximal term encourages the model to stay close to the previous iteration parameters, reducing the amount of information communicated between clients and the server. On each device k, the local model w_{t+1}^k is updated by minimizing the sum of the local loss $F_k(w)$ and the proximal term

$$\min_w h_k(w; w_t) = F_k(w) + \frac{\mu}{2} \|w - w_t\|^2, \tag{7}$$

where μ is the hyperparameter controlling the regularization strength, w is the the parameter vector of the local model and w_t is the parameter vector of the global (server) model. In FedProx the number of epochs is not uniform throughout and it is identified with γ_k^t.

The pseudocode of FedProx is given in Algorithm 3.

```
Algorithm 3 FedProx
Input: K, T, $\mu, \gamma$, $w_0$, N, $p_k$, k = 1,....N
  for t=0 to T - 1 do do
      Server selects a subset $S_t$ of K devices at random (each device k is chosen with probability $p_k$);
      Server sends $w_t$ to all chosen devices;
      Each chosen device $k \in S_t$ finds a $w_{t+1}^k$ which is a $\gamma$-inexact minimizer of:
          $w_{t+1}^k \approx \arg\min_w h_k(w; w_t) = F_k(w) + \frac{\mu}{2}\|w - w_t\|^2$;
      Each device k $\in S_t$ sends $w_{t+1}^k$ back to the server
      Server aggregates the w's as
          $w_{t+1} = 1/K \sum_{k \in S_t} w_{t+1}^k$
  end for
```

5. IMPLEMENTATION

In this paper, we present a decentralized and collaborative architecture for image classification. The federated learning architecture used for implementation involves a central server and multiple clients. The clients have the data and execute the model. Our objective is to show that deep CNN federated learning allows for data exchange among multiple clients for classification without compromising privacy. Our work focuses on the security and machine learning aspect while the various available software design architecture and patterns are discussed in detail in [35]. The CNN architecture is not our primary focus, and there are other architectural options that can marginally boost or decrease overall performance. A typical architecture of a convolutional neural network (CNN) includes the following elements:

Convolutional layers: Responsible for applying filters to extract features.

Pooling layers: Reduce the data spatial dimensions to reduce computation and over-fitting.

Activation functions: Add non-linearity to the network so that it can learn complex representations.

Fully connected layers: Integrate the previous layer information into output.

Regularization methods: Helps to prevent over-fitting.

Loss function: Utilizing a metric such as cross-entropy, calculate the difference between the expected and actual outputs.

Optimizer: Based on the gradient of the loss function, such as stochastic gradient descent (SGD), modify the network weights.

The design of a CNN might vary based on the task and the data, however these components are typical of many cutting-edge models. This CNN model learning phase is comprised of multiple communication rounds in which the central server interacts synchronously with the clients. Initially, the CNN model is setup with random weights w_0. We assume that K clients are available, with each client storing n_k private images locally. Each round consists of 3 rounds:

- The server distributes a global model ‘g’ with initial weights w_0 for a random subset of s_t clients.
- The weights are modified using the local objective minimization and averaging techniques.
- Lastly, to update the parameters of the global model g, the server receives updates from all involved clients and computes an average model w_t in accordance with equation (8).

$$w^t \leftarrow \sum_k^K \frac{n_k}{n} w_k^t. \tag{8}$$

These are the w^t parameters that were modified at round t, w_k^t are the parameters sent by client k at round t, n_k is the number of data points stored on client k, and n is the total number of data points that participated in collaborative training. The work is divided into three sections: CNN architecture, federated learning setup, and label-flipping attack configuration. For training a model that is utilized for image classification, MNIST dataset is used. All of the experiments were designed to train the model federatively utilizing ten virtual workers or agents, as well as to introduce and prevent malicious attacks by one or more agents.

Dataset description: The Modified National Institute of Standards and Technology dataset is known by the abbreviation MNIST dataset. This dataset includes 70 000 handwritten single digits between 0 and 9 in small square 28 by 28 pixel grayscale images. So, we are dividing this dataset into 60 000 training photos and 10 000 testing images. The training-testing ratio would be 85:15.

For demonstrating the security algorithms over FL frameworks, our proposed methodology includes the implementation in CNN for image classification for MNIST Dataset over the PySyft and Flower framework. The implementation for the efficient communication of the clients and server are demonstrated using the FedAvg and FedProx and for privacy preservation the Differential privacy is implemented/employed. Threat analysis is carried out on this implementation against label-flipping attack and mitigation strategy is demonstrated in the next section.

6. THREAT ANALYSIS

6.1. Threat and adversary model in label flipping attack

Participants in a federated learning system share model parameters with the centralized server rather than their dataset. There is no one to check each participant's dataset, which creates a vulnerability when the global model trains over each participant's dataset. An attacker can manipulate the training data such that the model learns incorrect patterns in label-flipping attack. Label-flipping attacks have the potential to weaken the resilience of machine learning models, leading to a substantial drop in their performance when used for real-world data. Label-flipping attacks can have significant effects in applications that utilize machine learning models for security-sensitive tasks, such as malware detection and fraud detection. A compromised model may fail to detect malicious activities, leading to security breaches. The objective of a targeted attack is to deliberately influence the behavior of the model in order to get it to incorrectly classify a particular set of instances.

An adversary participant can misclassify intentionally for example "Digit 1" data samples of the local training data as "Digit 7". The Scenario is explained in Fig. 2 [15].

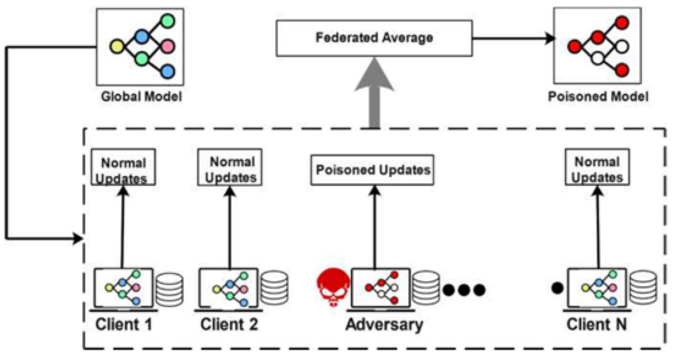

Fig. 2. Federated learning with label-flipping attacks

Since malevolent players can choose which class the model should identify, label-flipping is a targeted data poisoning attack. Each malevolent client C_i alters their dataset by changing c_{src} to c_{tar}, given a source class c_{src} and a target class c_{src} from C. As a result, the final global model M accuracy suffers, as it is more likely to misclassify images during testing.

6.2. Attack simulation

Attacking is simulated by establishing two PySyft federated loaders, one with genuine (benign) training data and the other with identical training data in the same sequence, but with the class labels swapped on one or all instances of a class (malicious data). The training logic iterates over both loaders, but batches from the benign and malicious data loaders are distributed to virtual benign and malicious agents, respectively. Training is carried out through a series of "timestamp" iterations, with the global model being disseminated to all agents and each agent receiving its initial batch of training data at the start of each iteration. During many training epochs per timestamp, each agent local model is incrementally updated. The model update parameters are calculated at the end of each timestamp by comparing the trained agent models to the global model that was originally broadcast to the agents.

The updates of malicious agents are boosted by a factor of 10, which is evenly divided across the malicious agents before the model parameter updates are collected and applied to the global model. For example, with only one hostile agent, model parameter updates would be boosted ten-fold, whereas, with two agents, each would be boosted five-fold, and so on. The changes are added to the global model before being delivered to the agents for the next scheduled iteration of training after they have been aggregated via weighted average.

6.3. Threat mitigation and prevention against label-flipping attack

The malicious agent seeks to attack the base model by increasing the weight deltas, which implies they must be relatively large compared to the weight deltas of benign agents to have an influence when all of the agents weight deltas are added together. To detect the anomalous agents in our experiment, a two-step technique implemented is given below.

6.3.1. Calculating the average distance

1. For each layer the distance between the weight deltas is calculated using the matrix multiplication method.
2. The weight deltas between two agents inside a layer are x1 and x2.
3. The p-norm distance between each vector pair $\in [0, \infty]$ is calculated using the parameter p.
4. Find the mean of the Euclidean distance, which indicates how dissimilar the means of two agents in a single layer are on average.

6.3.2. Ranking system

1. Score each layer agents based on how close or far they are to other agents.
2. The punishment metric is utilized, and each agent ranks the other agents according to how far they are from it.
    - The one who is closest to you gets a higher rank than the one who is far away.
    - The top 30%, average 40%, and worst 30% of the rankings are divided into three groups.
    - Punishment increases from the top to the bottom (remains the same within the group).
3. After calculating the score for each agent across all levels at a time stamp. Check if the score is less than a THRESHOLD

= Mean + SD, If Check is true, Tag agent as non-malicious Else, Tag Agent as malicious

4. The mean+SD provided insight into how different the larger score is from the lesser ones, which was useful in determining the threshold.

7. RESULTS AND DISCUSSION

7.1. Evaluation metrics

The metrics Precision, Recall, F1-Score, and False Positive Rate are used to assess the effectiveness of the proposed model's categorization and are defined as follows.

Accuracy: The Accuracy score is the proportion of correctly categorized images relative to the total number of images.

$$\text{Accuracy} = \frac{\text{TN}+\text{TP}}{\text{TN}+\text{TP}+\text{FP}+\text{FP}}. \tag{9}$$

Precision: The percentage of classification that were genuinely accurate is known as precision.

$$\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}}. \tag{10}$$

Recall: Recall measures a model ability to properly identify all instances of a certain class within a dataset. For example, if we want to measure the model's performance for classifying the hand written image 5, recall measures the percentage of actual image of '5' in the dataset that is actually classified as '5'

$$\text{Recall} = \frac{\text{TP}}{\text{TP}+\text{FN}}. \tag{11}$$

F1-Score: This metric helps to identify how many times a model made correct prediction and is a measure of accuracy.

$$\text{F1-score} = \frac{2*\text{Precision}*\text{Recall}}{\text{Precision}+\text{Recall}}. \tag{12}$$

A True Positive (TP) is a picture that a machine learning model properly identifies as being a member of a certain class or category. For example, the hand-written image of number '9' if properly classified by the model as '9', then it is TP.

A machine learning model that wrongly classifies a picture as being a member of a certain class or category is known as a False Positive (FP).

A True Negative (TN) in image classification is an image that a machine learning model properly identifies as not being within a specific class or category.

A False Negative (FN) in image classification is when an image is labelled as negative (not falling under a certain class) even when it does in fact fall under that class.

The accuracy is compared between various security algorithms and framework and here are the result analysis.

From Fig. 3, we conclude that the Pysyft framework works better in terms of accuracy than the Flower framework. Pysyft framework gives an accuracy of about 99% and Flower framework has an accuracy of 92%. Pysyft can be used to implement some privacy preserving algorithms where as it is not possible in Flower framework.

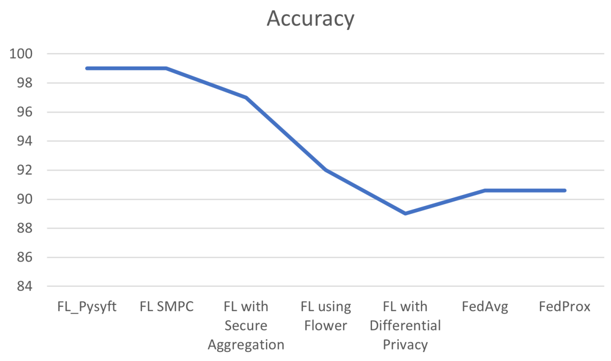

Fig. 3. Accuracy comparision

When privacy algorithms applied to the FL framework, the accuracy does not drastically change and remains the same for a few algorithms such as secure multi party computation with an accuracy of 99%. Whereas the other algorithms gave an accuracy of 97% for Secure Aggregation and 89% for Differential Privacy. The noise that is added to the gradients is what causes the accuracy for DP to decrease. Though there is a decrease, it makes it harder for an attacker to identify individual data in the training set thus providing more privacy.

From the graph in Fig. 4, we can understand that federated learning implemented in Pysyft framework executes the highest accuracy of 99% and federated learning with DP executes the lowest accuracy of 89%.

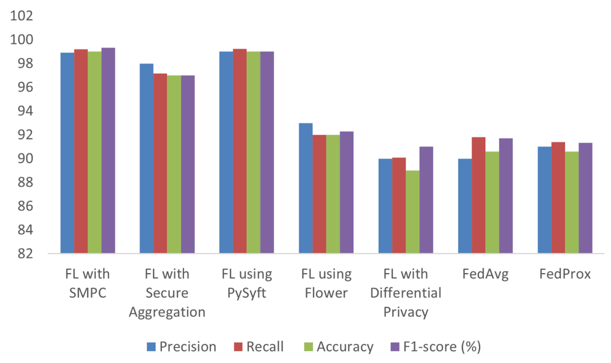

Fig. 4. Comparision based on evaluation metrics

The loss and accuracy of when algorithms FedProx and FedAvg were implemented is shown below.

Figures 5 and 6 represent the accuracy graph of training the model on 3 clients using FedProx and FedAvg algorithms respectively. All the clients were executed with same configuration on i5-9400 machines. The accuracy is approximately around 90% for both the algorithms. For label-flipping attack, we have calculated the misclassification rates:

$$\text{Misclassification Rate} = \frac{\text{Number of misclassified class 5 -> images}}{\text{Total number of class 5 images}} * 100, \tag{13}$$

$$\text{Global model accuracy} = \frac{\text{No. of Correct Predictions}}{\text{Total no. of images in test dataset}} * 100. \tag{14}$$

Attack success rate (ASR) is the percentage of samples from the source class that were misclassified as belonging to the target class. ASR value should be low.

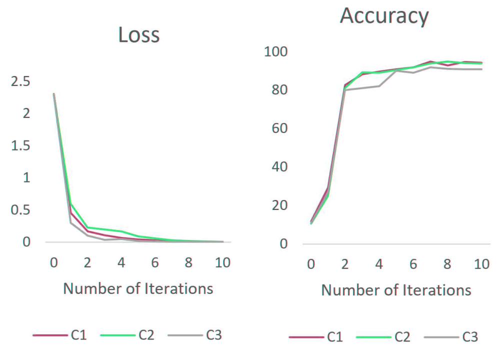

Fig. 5. Loss and accuracy of FedProx

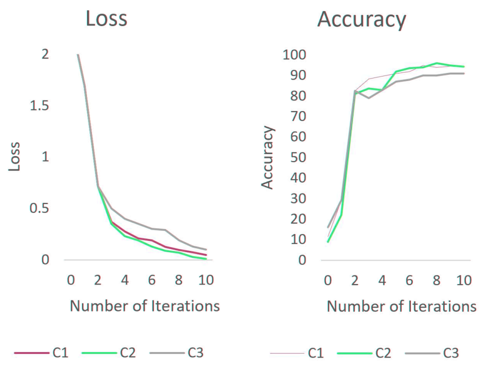

Fig. 6. Loss and accuracy of FedAvg

We deployed one malevolent agent and nine benign agents in our scenario. The dataset was spread evenly among all agents, and the malicious agent training data was adjusted for targeted attack by changing all class 5 sample labels to class 7. The datasets were also switched between timestamps to ensure that both benign and malicious agents received a fresh set of training examples, while the malicious agent data was distorted by changing its class from 5 to 7. We gradually increased the number of malicious clients and found our work robust even when there is increase in number of clients. The attack success rate of our work and FGold is compared and the results are shown in the next section.

The malicious agent weight delta boosting factor is calculated. The boosting factor in this example is 10 because the weights are evenly distributed. In this case, two trials were carried out with attack prevention disabled and with attack prevention enabled.

7.2. Attack prevention disabled

The global model accuracy on the test dataset is represented by the orange line, whereas the global model misclassification accuracy on class 5 is represented by the blue line in Fig. 7.

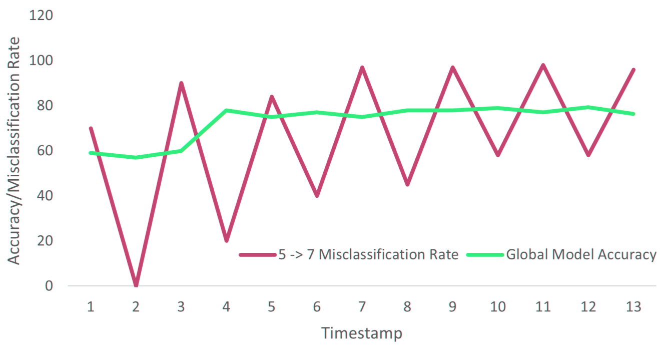

Fig. 7. Accuracy/misclassification rate of attack

The global model is quite confident (approximately 75%) in its misclassification at timestamp 1, while the overall accuracy is only about 60%. The weight deltas from benign agents override the influence of the malicious agent delta in timestamp 2, resulting in an extremely low misclassification rate. With higher timestamps, the oscillating behavior is repeated, with the global model misclassification confidence stabilizing. However, the global model misclassification does not totally stabilize even after 13 timestamps. At the end of the training, the overall accuracy of the test dataset reaches 76%.

7.3. Attack prevention enabled

The behavior is absolutely different in this instance. The attack detection and prevention approach identifies malicious updates and prevents them from influencing the global model. Figure 8 shows the situation. At timestamp 1, the global model is moderately confident in misclassifying class 5 to class 7 on the test data, but its misclassification confidence quickly drops in higher timestamps, reaching low confidence of roughly 3% by the end of timestamp 13. The accuracy of the whole global model is similarly trending upwards, reaching 85 percent by the end of timestamp 13. The analysis shows that the hostile agent had no effect on the global model, proving that attack prevention worked.

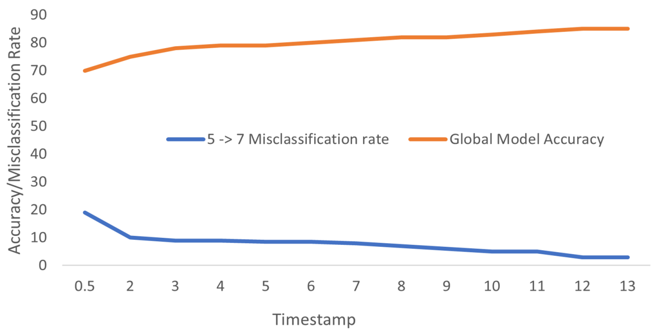

Fig. 8. Accuracy/misclassification rate of attack when attack is enabled

7.4. Comparison with existing approach

The work is tested against existing method for Poisoning Attack FoolsGold (FGold) [22] on the metrics ASR. The result of analysis is showed in Fig. 9. Our work proves to be stable even when

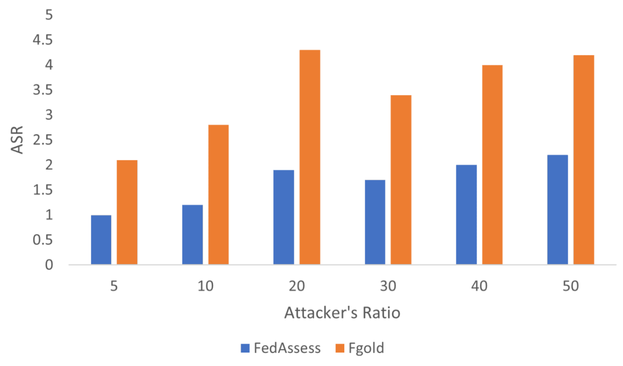

Fig. 9. Analysis of attack success rate

the number of malicious clients increases and attack success rate seems to be low when compared to FoolsGold [22].

8. CONCLUSION

In this article, federated learning has been implemented for benchmark data set and we have found the accuracy has improved compared to a centralized machine learning system. Our work has analyzed FL with additional security features and implemented algorithms for communication efficient Federated Learning. The proposed model demonstrates label-flipping attack in FL and developed a novel method for mitigating model poisoning attack. All our empirical results prove that our work is robust against label-flipping attack and the model performs well, even after adding all the security algorithms. We have demonstrated the work in two different FL frameworks. Our future work focuses on quantum computing based federated learning which provides better performance in FL with any ML model and is more secure.

REFERENCES

[1] K. Kuźniewski, K. Matusiewicz, and P. Sapiecha, "The high-level practical overview of open-source privacy-preserving machine learning solutions," *International Journal of Electronics and Telecommunications*, pp. 741–747, 2022, doi: 10.24425/ijet.2022.143880.

[2] X. Zheng and Z. Cai, "Privacy-preserved data sharing towards multiple parties in industrial iots," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 5, pp. 968–979, 2020, doi: 10.1109/JSAC.2020.2980802.

[3] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019, doi: 10.1145/3298981.

[4] M.H. Ur Rehman and M.M. Gaber, *Federated learning systems: Towards next-generation AI*. Springer Nature, 2021, vol. 965, doi: 10.1007/978-3-030-70604-3.

[5] P. Kairouz *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021, doi: 10.1561/2200000083.

[6] G. Bao and P. Guo, "Federated learning in cloud-edge collaborative architecture: key technologies, applications and challenges," *J. Cloud Comput.*, vol. 11, no. 1, p. 94, 2022, doi: 10.1186/s13677-022-00377-4.

[7] C. Dwork, "Differential privacy: A survey of results," in *Theory and Applications of Models of Computation: 5th International Conference, TAMC 2008, Proceedings 5* China: Springer, 2008, pp. 1–19, doi: 10.1007/978-3-540-79228-4_1.

[8] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the forty-first annual ACM symposium on Theory of computing*, 2009, pp. 169–178, doi: 10.1145/1536414.1536440.

[9] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction", *Advances in Neural Information Processing Systems*, vol. 26, 2013, [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2013/file/ac1dd209cbcc5e5d1c6e28598e8cbbe8-Paper.pdf

[10] L. Lyu, H. Yu, and Q. Yang, "Threats to federated learning: A survey," *arXiv preprint arXiv:2003.02133*, 2020, doi: 10.48550/arXiv.2003.02133.

[11] N. Rodríguez-Barroso, D. Jiménez-López, M. V. Luzón, F. Herrera, and E. Martínez-Cámara, "Survey on federated learning threats: Concepts, taxonomy on attacks and defences, experimental study and challenges," *Inf. Fusion*, vol. 90, pp. 148–173, 2023, doi: 10.1016/j.inffus.2022.09.011.

[12] P. Rieger, T.D. Nguyen, M. Miettinen, and A.-R. Sadeghi, "Deepsight: Mitigating backdoor attacks in federated learning through deep model inspection," *arXiv preprint arXiv:2201.00763*, 2022, doi: 10.48550/arXiv.2201.00763.

[13] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 2938–2948. [Online]. Available: https://proceedings.mlr.press/v108/bagdasaryan20a.html

[14] Z. Chen, P. Tian, W. Liao, and W. Yu, "Towards multi-party targeted model poisoning attacks against federated learning systems," *High-Confidence Computing*, vol. 1, no. 1, p. 100002, 2021, doi: 10.1016/j.hcc.2021.100002.

[15] S. Awan, B. Luo, and F. Li, "Contra: Defending against poisoning attacks in federated learning," in *Computer Security–ESORICS 2021: 26th European Symposium on Research in Computer Security, Darmstadt, Germany, October 4–8, 2021, Proceedings, Part I 26*. Springer, 2021, pp. 455–475, doi: 10.1007/978-3-030-88418-5_22.

[16] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," in *Computer Security–ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Proceedings, Part I 25*. UK: Springer, 2020, pp. 480–501, doi: 10.1007/978-3-030-58951-6_24.

[17] V. Valadi, X. Qiu, P.P.B. de Gusmão, N.D. Lane, and M. Alibeigi, "Fedval: Different good or different bad in federated learning," *arXiv preprint arXiv:2306.04040*, 2023, doi: 10.48550/arXiv.2306.04040.

[18] R. Anusuya, D. Karthika Renuka, S. Ghanasiyaa, K. Harshini, K. Mounika, and K. Naveena, "Privacy-preserving blockchain-based ehr using zk-snarks," in *Computational Intelligence, Cyber Security and Computational Models. Recent Trends in Computational Models, Intelligent and Secure Systems: 5th International Conference, ICC3 2021, Revised Selected Papers*. India: Springer, 2022, pp. 109–123, doi: 10.1007/978-3-031-15556-7_8.

[19] A. Ziller *et al.*, "Pysyft: A library for easy federated learning," *Federated Learning Systems: Towards Next-Generation AI*, pp. 111–139, 2021, doi: 10.1007/978-3-030-70604-3_5.

[20] D.J. Beutel *et al.*, "Flower: A friendly federated learning research framework," *arXiv preprint arXiv:2007.14390*, 2020, doi: 10.48550/arXiv.2007.14390.

[21] P. Blanchard, E.M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, vol. 30, 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/f4b9ec30ad9f68f89b29639786cb62ef-Paper.pdf

[22] C. Fung, C.J.M. Yoon, and I. Beschastnikh, "The limitations of federated learning in sybil settings," in *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*. San Sebastian, USA: USENIX Association, 2020, pp. 301–316. [Online]. Available: https://www.usenix.org/conference/raid2020/presentation/fung

[23] N.M. Jebreel, J. Domingo-Ferrer, D. Sánchez, and A. Blanco-Justicia, "Defending against the label-flipping attack in federated learning," *arXiv preprint arXiv: 2207.01982*, vol. abs/2207.01982, 2022, doi: 10.48550/arXiv.2207.01982.

[24] A.F. Siegel, "Robust regression using repeated medians," *Biometrika*, vol. 69, no. 1, pp. 242–244, 1982.

[25] A.K. Singh, A. Blanco-Justicia, J. Domingo-Ferrer, D. Sánchez, and D. Rebollo-Monedero, "Fair detection of poisoning attacks in federated learning," in *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2020, pp. 224–229, doi: 10.1109/ICTAI50040.2020.00044.

[26] L. Zhao *et al.*, "Shielding collaborative learning: Mitigating poisoning attacks through client-side detection," *IEEE Trans. Dependable Secur. Comput.*, vol. 18, no. 5, pp. 2029–2041, 2020, doi: 10.1109/ TDSC.2020.2986205.

[27] 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. Forensic Secur.*, vol. 17, pp. 1639–1654, 2022, doi: 10.1109/TIFS.2022.3169918.

[28] S. Shen, S. Tople, and P. Saxena, "Auror: Defending against poisoning attacks in collaborative deep learning systems," in *Proceedings of the 32nd Annual Conference on Computer Security Applications*, 2016, pp. 508–519, doi: 10.1145/2991079.2991125.

[29] M. Hirt and D. Tschudi, "Efficient general-adversary multi-party computation," in *Advances in Cryptology-ASIACRYPT 2013: 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part II 19*. Springer, 2013, pp. 181–200, doi: 10.1007/978-3-642-42045-0_10.

[30] S. Wagh, D. Gupta, and N. Chandran, "Securenn: 3-party secure computation for neural network training." *Proc. Priv. Enhancing Technol.*, vol. 2019, no. 3, pp. 26–49, 2019, doi: 10.2478/popets-2019-0035.

[31] I. Damgård, M. Keller, E. Larraia, V. Pastro, P. Scholl, and N.P. Smart, "Practical covertly secure mpc for dishonest majority–or: breaking the spdz limits," in *Computer Security–ESORICS 2013: 18th European Symposium on Research in Computer Security, Egham, UK, September 9-13, 2013. Proceedings 18*. Springer, 2013, pp. 1–18, doi: 10.1007/978-3-642-40203-6_1.

[32] K. Bonawitz *et al.*, "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1175–1191, doi: 10.1145/3133956.3133982.

[33] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B.A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282. [Online]. Available: https://proceedings.mlr.press/v54/mcmahan17a.html

[34] T. Li, A.K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine learning and systems*, vol. 2, pp. 429–450, 2020. [Online]. Available: https://proceedings.mlsys.org/paper/2020/file/38af86134b65d0f10fe33d30dd76442e-Paper.pdf

[35] S. Kit Lo, Q. Lu, L. Zhu, H.-y. Paik, X. Xu, and C. Wang, "Architectural patterns for the design of federated learning systems," *arXiv preprint arXiv: 2101.02373*, 2021, doi: 10.48550/arXiv.2101.02373.