Federated Anomaly Analytics for Local Model Poisoning Attack

Siping Shi, Chuang Hu, Dan Wang, *Senior Member, IEEE*, Yifei Zhu, *Member, IEEE*, and Zhu Han, *Fellow, IEEE*

***Abstract*—The local model poisoning attack is an attack to manipulate the shared local models during the process of distributed learning. Existing defense methods are passive in the sense that they try to mitigate the negative impact of the poisoned local models instead of eliminating them. In this paper, we leverage the new federated analytics paradigm, to develop a proactive defense method. More specifically, federated analytics is to collectively carry out analytics tasks without disclosing local data of the edge devices. We propose a Federated Anomaly Analytics enhanced Distributed Learning (FAA-DL) framework, where the clients and the server collaboratively analyze the anomalies. FAA-DL firstly detects all the uploaded local models and splits out the potential malicious ones. Then, it verifies each potential malicious local model with functional encryption. Finally, it removes the verified anomalies and aggregates the remaining to produce the global model. We analyze the FAA-DL framework and show that it is accurate, robust, and efficient. We evaluate FAA-DL by training classifiers on MNIST and Fashion-MNIST under various local model poisoning attacks. Our experiment results show FAA-DL improves the accuracy of the learned global model under strong attacks up to 6.90 times and outperforms the state-of-the-art defense methods with a robustness guarantee.**

***Index Terms*—Distributed learning, local model poisoning attacks, federated analytics, anomaly detection, functional encryption.**

I. Introduction

ARTIFICIAL intelligence (AI) driven by big data shows its strengths in almost every industry and most walks of life. The data existing in the form of isolated islands and the strengthening of data privacy impel data mining and analysis in a *federated* way [1]–[3], where a model is learned across multiple decentralized edge devices holding local data samples without exchanging data directly [4]–[7]. In the federated scenarios, *local model poisoning attack*, an attack to poison local models on edge devices, has a harmful effect on the whole learning process, such as broadly slowing down the convergence rate of the learning process in [8] and significantly degrading the prediction accuracy of the learned global model in [9].

Existing defense methods for the local model poisoning attacks are mainly *passive* [10], [11], employing robust statistical methods to mitigate the negative impact of the poisoned local models and treating the normal local models and the poisoned local models indiscriminately. For instance, using the trimmed average value of all the clients' local models to generate the global model [10]. Although passive defense methods are simple and highly decoupled with the underlying learning systems, they are accompanied by several drawbacks: First, as passive defense methods cannot eliminate all the poisoned local models, the training performance is affected to some degree, i.e., the accuracy of the learned global model is reduced; Second, the privacy of local data is not preserved as part of the local data are collected for further computation [12]. A few *proactive defense* methods are also explored via detecting and removing the poisoned local models from the whole set with a high-accuracy abnormal detection model [13]. However, proactively defending the local model poisoning attacks requires data analytics on distributed stored datasets rather than complicated detection models. *It inspires us to apply federated analytics instead of federated learning in defense of local model poisoning attacks.*

With the success of applying federated learning in training models, there is a growing interest in using federated technologies to answer more fundamental questions about decentralized data - like computing counts - that often do not involve machine learning at all. To achieve this, the Google AI team proposes a new paradigm, *federated analytics* (FA) [14], to collectively carry out analytics tasks without disclosing local data of the edge devices. FA is first explored by Gboard [15] engineers to measure the overall quality of next word prediction models against raw typing data held on users' phones: Participating phones locally computed a metric of how well the model's predictions matched the words that were actually typed, and then engineers obtained a population-level summary of model performance by averaging the metrics

Manuscript received March 1, 2021; revised September 9, 2021; accepted September 22, 2021. Date of publication October 11, 2021; date of current version January 17, 2022. The work of Chuang Hu was supported in part by The Hong Kong Polytechnic University (PolyU) under Grant 1-ZVUG. The work of Dan Wang was supported in part by the General Research Fund (GRF) under Grant 15210119, Grant 15209220, and Grant 15200321; in part by the Innovation and Technology Fund-Innovation and Technology Support Programme (ITF-ITSP) under Grant ITS/070/19FP; in part by the Collaborative Research Fund (CRF) under Grant C5026-18G and Grant C5018-20G; in part by PolyU under Grant 1-ZVPZ; and in part by Huawei Collaborative Project. The work of Yifei Zhu was supported by Shanghai Jiao Tong University (SJTU) Explore-X Grant. The work of Zhu Han was supported in part by the National Science Foundation (NSF) under Grant CNS-2128368 and Grant CNS-2107216 and in part by Toyota and Amazon. *(Corresponding author: Chuang Hu.)*

Siping Shi, Chuang Hu, and Dan Wang are with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong (e-mail: cssshi@comp.polyu.edu.hk; cschuang.hu@comp.polyu.edu.hk; csdwang@comp.polyu.edu.hk).

Yifei Zhu is with the University of Michigan-Shanghai Jiao Tong University Joint Institute, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: yifei.zhu@sjtu.edu.cn).

Zhu Han is with the Department of Electrical and Computer Engineering, University of Houston, Houston, TX 77004 USA, and also with the Department of Computer Science and Engineering, Kyung Hee University, Seoul 446-701, South Korea (e-mail: zhan2@uh.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JSAC.2021.3118347.

Digital Object Identifier 10.1109/JSAC.2021.3118347

uploaded by many phones. Obviously, FA is essential to ensure that the underlying data remains private and secure, unlike the traditional collaborative analytics approaches required to collect local data [16]. Even though FA re-uses the infrastructure of federated learning, it does not have the model training part and mainly focuses on data science needs. FA can be easily applied to various data analytics tasks, including but not limited to model evaluation [17], data query [18], and frequently used items discovering [19]. FA has huge value in practice and it can make contributions to develop better applications.

Intuitively, there are several reasons to leverage the new federated analytics paradigm to develop a proactive defense method. Firstly, compared to applying anomaly detection separately [13], federated analytics enables the central server to have a more accurate anomaly analytics result via peer-wise collaboration. Secondly, federated analytics is more practical and efficient in reality, while the learning approach requires massive data and costs much time for training. Thirdly, in federated analytics, since raw data will not be exposed, the users' privacy can be well protected. This is extremely important for devices working in privacy-sensitive scenarios, like healthcare and personal applications.

Federated analytics can be implemented in two ways: 1) sending the intermediate analytics result of local data to the server for aggregation [17], [18], and 2) sending the transformed local data to the server for analyzing [19]. We know that the intermediate analytics results calculated by malicious clients are not reliable, and thus the second way is more appropriate in our scenario. However, how to transform the local data without information leakage is challenging. Differential privacy is an attractive option as it is lightweight and easy to deploy, but the accuracy is decreased to ensure a high level of privacy [20]. Another choice is using cryptographic technology. Classical cryptographic methods are secure multi-party computation (SMC) and homomorphic encryption (HE). Both of them are not efficient and with high computational costs in practice [21]. Recently, the emerging functional encryption (FE) [22] demonstrates its efficiency and practicality in allowing the central server to learn specific functions of the encrypted data without exposing any plain text. Therefore, considering these advantages of FE, we choose the FE scheme to support federated analytics.

In this paper, we propose a *Federated Anomaly Analytics* (FAA-DL) framework to defend local model poisoning attacks in distributed machine learning. In this framework, we firstly apply a lightweight *anomaly detection* to split out potential malicious local models. Then, it employs the privacy-preserving *anomaly verification* via functional encryption to verify the anomalies with high accuracy. Finally, FAA-DL removes the verified anomalies accurately and aggregates the remaining normal local models to produce the global model. The contributions of this work are summarized as follows:

- We design a federated anomaly analytics enhanced distributed learning framework to proactively defend local model poisoning attacks with high accuracy, strong robustness, and low overheads.
- With the theoretical analysis, we prove that our framework is robust to strong attacks with high accuracy and low time complexity, and the convergence rate can reach to $\mathcal{O}\left(\frac{1}{T}\right)$.
- We evaluate our framework through training classifiers on MNIST and Fashion-MNIST. The extensive experiments demonstrate that our framework improves the accuracy of the learned global model under strong attacks for $1.44\times \sim 6.90\times$ and significantly better than other existing defense methods.

The rest of this paper is organized as follows. Section II discusses the related work. Section III defines the problem and defense goals. Section IV introduces the proposed framework FAA-DL in detail and Section V demonstrates theoretical analysis. Section VI describes the implementation of FAA-DL and the evaluation results of FAA-DL based on extensive experiments are presented in Section VII. Finally, a conclusion is drawn in Section VIII.

II. Related Work

A. Passive Defense vs Proactive Defense

The methods to defend against the local model poisoning attacks in the distributed learning system can be categorized into passive defense and proactive defense. Most of them are passive defense: Trimmed mean [10] uses trimmed average value to defend the attacks, GeoMed [11] computes the geometric median of all local models to update the global model, and Krum [8] selects the local model which has the minimum sum of distances to its closest neighbors to perform model aggregation. These methods do not differentiate the poisoned local models from the normal ones. Instead, they aim to tolerate the adversarial attacks and mitigate their negative impacts. They can easily fail when encountered fine crafted local model poisoning attacks as described in [23].

The proactive defense approach was first proposed in [13]. It tries to defend the attacks by detecting and removing the malicious local models with a pre-trained spectral anomaly detection model and it shows better performance than the passive ones mentioned above. However, it assumes the distribution of the local training dataset is known, which is challenging to fulfill in practice. Moreover, it does not consider the large variance of local model updates generated by non-IID local datasets, and the attacks which take advantage of this property make them ineffectively [24].

B. Federated Analytics

Federated analytics (FA) is recently proposed in [14]. It is to collectively carry out analytics tasks for certain data science needs without disclosing local data of the edge devices. Unlike federated learning which serves for neural network model training, FA mainly focuses on data science methods. It applies the infrastructure of federated learning but without the learning part. There are two approaches to achieve the goal: 1) sending the intermediate analytics result of local data to the server for aggregation [17], [18], and 2) sending the transformed local data to the server for analyzing [19].

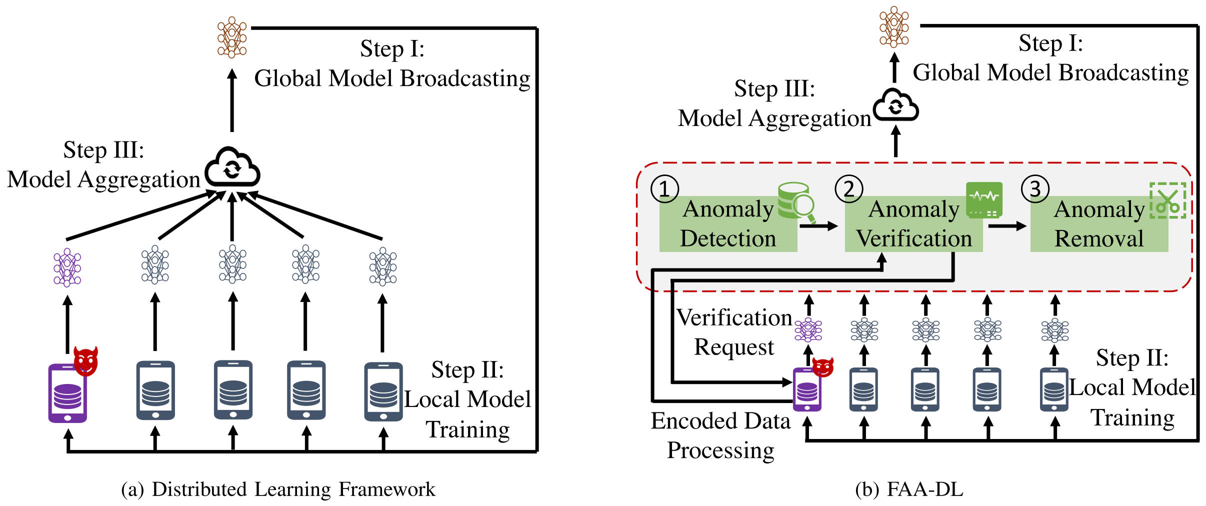


Fig. 1. Overview of framework. (a) is the framework of classical distributed learning, and (b) is our proposed framework with three additional modules: Anomaly Detection (Section IV-B), Anomaly Verification (Section IV-C) and Anomaly Removal (Section IV-D).

For the first approach, the intermediate analytics result can be a statistical result (i.e., counting, mean, median, and so on) or an inference result (i.e., Bayesian inference) based on the local data of clients. In this branch, Google initially applies FA to support the quality measurement of federated learning in Song Recognition [17]. Here each selected phone computes the recognition rate for the specific songs with their historical data and then sends it to the federated analytics server for aggregation. Similarly, Orchard [18] is proposed to answer queries about sensitive data held by millions of user devices through federated analytics.

For the second approach, the local data should be first transformed into the form without information leakage, and then the server computes over the transformed data to obtain the analytics result. For instance, heavy hitter [19] transforms the user-generated data streams with differential privacy and discovers the most frequently used items in them. As differential privacy sacrifices accuracy for privacy, the cryptographic method is another choice to transform raw data. We know the classical cryptographic methods are time-consuming, such as homomorphic encryption. Therefore, it is essential to choose an appropriate cryptographic method to efficiently implement FA.

C. Functional Encryption

Functional encryption (FE) is the generalization of public-key encryption in which possessing a secret key allows one to learn a function of what the ciphertext is encrypting. To be concrete, a functional encryption scheme for a given function consists of four algorithms: 1) setting public key and master secret key, 2) generating secret function key based on the master secret key and the specific function, 3) encrypting plain text with the public key, and 4) computing the function result with the secret function key and encrypted data. FE enables one to learn partial information of the private data without knowing what the data is. There are several classes of FE schemes in literature, including identity-based encryption (IBE) [25], attribute-based encryption (ABE) [26], and inner product functional encryption (IPFE) [27]. FE is first proposed by [28] and formalized by [29]. Recently, several works fed FE into the machine learning procedure for privacy-preserving machine learning problems, including encrypted inference [22] and encrypted training [30].

III. System Model and Problem Formulation

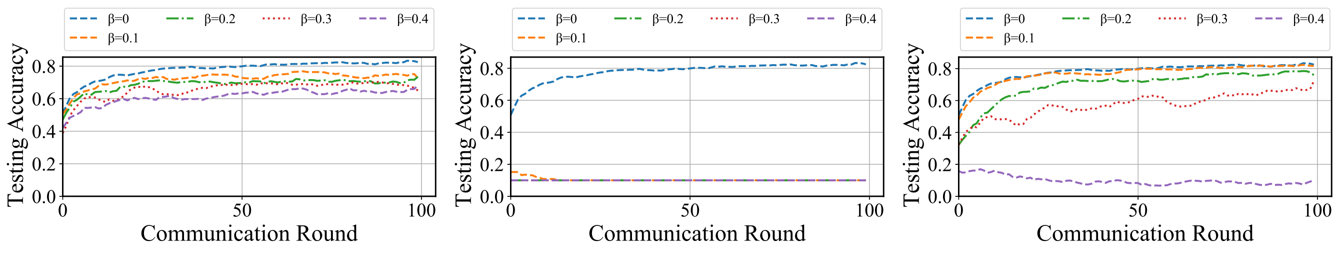
In this section, the scenario of distributed learning under local model poisoning attacks is described. We first present the distributed learning system model in Section III-A. Then we formulate the corresponding problem in Section III-B. Finally, the defense goals are set in Section III-C.

A. Distributed Learning System Model

We consider a distributed learning system with a server and n clients as shown in Figure 1a. Each client i has a local training dataset $D_i, i = 1, 2, \cdots, n$. In each iteration of model training, the server first broadcasts the current global model $\boldsymbol{w}$ to all clients, and then the clients perform local model training, finally, the server aggregates all the uploaded local models to update the global model for next iteration.

1) Threat Model: The type of attack we considered is the local model poisoning attack that the attacker directly manipulates the local model updates with a certain poisoning function, i.e., manipulates the local model updates by adding Gaussian noise as in [31], but not indirectly manipulates the local model updates with the generated training data as in the data poisoning attacks [32]. We also assume the proportion of the malicious clients is below 0.5 (otherwise, the system turns out of control) as in [23], [33].

2) Attacker's Capability: As in most existing work [23], [33], we assume the attacker can control the clients to be malicious but cannot control the server. Specifically, it can modify the local model updates sent from the controlled clients to the server to be the malicious ones. Following most of the existing local model poisoning attacks [23], [24], [31], [33],



(a) Gaussian attack (b) Sign-flipping attack (c) ALIE attack

Fig. 2. Model accuracy affected by the proportion of malicious devices and various attacks.

the malicious local model updates generated by the attacker is based on all the known local model updates but not obtained by training a malicious dataset. Besides, the attacker has partial knowledge about the clients that it can only know the local model updates but not the local data of all the participated clients. Without loss of generality, we assume the last m of the n participated clients are malicious and the left is all benign.

For each benign client, the local model is trained via fine-tuning the global model with its local dataset. Formally, the i-th client solves the optimization problem

$$\min_{\boldsymbol{w}_i} f_i(D_i, \boldsymbol{w}_i), \quad i = 1, \ldots, n-m, \tag{1}$$

where f_i and $\boldsymbol{w}_i$ is the objective function and the local model of client i, respectively. The local model update of client i is defined as

$$\boldsymbol{g}_i = \boldsymbol{w}_i - \boldsymbol{w}, \quad i = 1, \ldots, n-m. \tag{2}$$

For each malicious client, the local model update is replaced by the arbitrary malicious value $\hat{\boldsymbol{g}}_j$ which is generated by the adversary. Assume $\boldsymbol{g}_j$ the true local model update of client j. We have

$$\hat{\boldsymbol{g}}_j = \mathcal{P}\left(\boldsymbol{g}_j\right), \quad j = n-m+1, \ldots, n, \tag{3}$$

where $\mathcal{P}$ is the poisoning function and $\hat{\boldsymbol{g}}_j \neq \boldsymbol{g}_j$.

Once the computation of local model training is finished, each client sends the local model update to the server, and the server aggregates them to produce the global model update according to a specific aggregation rule. We choose the popular FedAvg [34] as the aggregation rule in the distributed learning system. Assume $\boldsymbol{g}$ is the global model update, and the dataset of each client is balanced. We have

$$\boldsymbol{g} = \frac{1}{n}\left(\sum_{i=1}^{n-m} \boldsymbol{g}_i + \sum_{j=n-m+1}^{n} \hat{\boldsymbol{g}}_j\right). \tag{4}$$

As $\hat{\boldsymbol{g}}_j \neq \boldsymbol{g}_j$, the global model update deviates from the value it should be, and thus the performance of the global model is hurt. Let $\mathcal{T}$ be the negative impact caused by the local model poisoning attack on the global model, and we use the deviation degree of the global model update to measure it. Thus we have

$$\mathcal{T} = m \cdot \mathbb{E}\left[\sum_{j=n-m+1}^{n} \left\|\hat{\boldsymbol{g}}_j - \boldsymbol{g}_j\right\|\right]. \tag{5}$$

Taken (3) into (5), we get

$$\mathcal{T} = m \cdot \mathbb{E}\left[\sum_{j=n-m+1}^{n} \left\|\mathcal{P}\left(\boldsymbol{g}_j\right) - \boldsymbol{g}_j\right\|\right]. \tag{6}$$

From (6), we observed that the negative impact of the local model poisoning attack is determined by two factors: (i) the poisoning function $\mathcal{P}$ determined by the adversary, and (ii) the number m of malicious local model updates in a distributed learning system. We further confirm these observations with simulation experiments shown in Figure 2. For each local model poisoning attack: the more clients become malicious, the more significant the performance degradation it will cause.

B. Problem Formulation

In a distributed learning system, the server has no knowledge about the malicious clients. Let $\boldsymbol{C}$ and $\boldsymbol{C}_m$ be the set of all clients and the set of malicious clients, respectively. We have $\boldsymbol{C}_m \subset \boldsymbol{C}$. Since the presence of local model poisoning attacks hurts the performance of the learned global model, our objective is to minimize the negative impact caused by the local model poisoning attack, and we have the optimization problem

$$\min_{\boldsymbol{w}} F(\boldsymbol{w}) = \frac{1}{|\boldsymbol{C}| - |\boldsymbol{C}_m|} \sum_{i \notin \boldsymbol{C}_m} f_i\left(D_i, \boldsymbol{w}\right), \tag{7}$$

where $|\boldsymbol{C}|$ and $|\boldsymbol{C}_m|$ are the cardinality of $\boldsymbol{C}$ and $\boldsymbol{C}_m$, respectively, and $F(\cdot)$ is the objective function of the global model.

To solve (7), we need to determine $\boldsymbol{C}_m$ according to the given information. The server has full access to the local model updates set $\boldsymbol{G}$ and the global model $\boldsymbol{w}_k$ in each iteration k, but have no knowledge of the raw local training data D_i on the client i. However, the server can access encrypted local training data $Enc(D_i)$, and only the client itself can decrypt $Enc(D_i)$ to D_i. Let T be the workload of the server, we have the constraint

$$T \leq T_{max}, \tag{8}$$

where T_{max} is the maximum workload of the server. Therefore, our problem can be formulated as

Problem: Given $\boldsymbol{G}$, $\boldsymbol{w}_k$ and $Enc(D_i)$, determine $\boldsymbol{C}_m$, to minimize $F(\boldsymbol{w})$, with constraint (8).

C. Defense Goals

Our goals for defending the local model poisoning attacks in distributed learning systems can be decomposed into three aspects as following.

1) Fidelity: The designed method should guarantee the fidelity of distributed learning. Specifically, we take FedAvg [34] under no attacks as the baseline. The learned global model should be as accurate as the baseline learned when there is no attack.

2) Robustness: The designed method should have the ability to defend against strong attacks. This allows the distributed learning system to learn a global model as accurate as the baseline learned even under strong attacks, i.e., with a large proportion of malicious clients.

3) Efficiency: Much extra computation and communication overheads should not occur in the distributed learning system, especially in resource-constrained edge devices. Thus the designed method can learn a global model as efficiently as the baseline.

IV. A Federated Anomaly Analytics Enhanced Distributed Learning Framework

A. Overview

Our proposed Federated Anomaly Analytics enhanced Distributed Learning Framework (FAA-DL) proactively defends the local model poisoning attack with the assistance of federated anomaly analytics. To achieve the accuracy and robustness goals mentioned in Section III-C, we execute not only anomaly detection but also anomaly verification. Moreover, we apply Functional Encryption and All-or-Nothing Transform to ensure the efficiency goal.

We present the overall architecture of FAA-DL in Figure 1b. The system consists of a set of distributed edge devices and a centralized server. In each iteration, the server will broadcast a global model to the edge device. The edge device updates the global model based on the local data and forms the local model, which is then sent to the server for anomaly analytics and model aggregation. The anomaly analytics logic is executed before aggregating the local models into a global model.

Specifically, FAA-DL enhances the original FL pipeline with three extra modules: 1) The **Anomaly Detection Module** runs the anomaly detection algorithm (Section IV-B). It takes the local model updates as inputs, computes the optimal anomaly decision boundary to identify whether the local model updates are malicious or not. The normal local model updates are sent to the *aggregator* for further model aggregation; while the malicious local model updates are sent to the Anomaly Verification Module for verification. 2) The **Anomaly Verification Module** runs the anomaly verification algorithm. It takes the local model that is identified as malicious by anomaly detection and the encrypted local training data as inputs, computes the true local model updates through Functional Encryption (Section IV-C). If the local model update is verified as malicious, it sends the local model update to the Anomaly Removal; otherwise, to the aggregator for further model aggregation. 3) The **Anomaly Removal Module** removes all true anomalies from the local model updates set (Section IV-D).

B. Lightweight and Unsupervised Anomaly Detection

Since the dominant majority of the local model updates are benign, we can use an anomaly detection algorithm to filter out malicious ones.

Significant work has been expended to improve the accuracy of anomaly detection [35]–[37]. Across these work, a dominant strategy is to improve detection accuracy by using advanced supervised/semi-supervised learning-based methods [38]–[40]. However, employing such learning-based methods is challenging due to their time-consuming characteristics and lacking labeled data in practice.

To this end, we develop a lightweight and unsupervised anomaly detection method based on Support Vector Machine (SVM) [41]. The main idea is to estimate a non-linear decision boundary to find anomalies, using appropriate kernel functions and soft margins. Through dimensionality reduction techniques, we can reduce the overhead induced by the computation on high-dimensional local model updates. The optimization objective of anomaly detection is to separate all data points from the origin by a maximum margin with respect to some constraint relaxation, as shown in following

$$\min_{w\in F,\ \boldsymbol{\xi}\in\mathbb{R}^{\ell},\ \rho\in\mathbb{R}} \quad \frac{1}{2}\|w\|^2+\frac{1}{\nu\ell}\sum_i \xi_i-\rho \tag{9}$$

$$\text{subject to} \quad w\cdot\Phi\left(\mathbf{x}_i\right)\geq\rho-\xi_i \tag{10}$$

$$\text{and } \xi_i\geq 0, \tag{11}$$

where w and ρ create a hyper-plane that maximizes the distance from the origin in space F. The variable ν provides the upper bound on the fraction of anomalies and $\boldsymbol{\xi}$ is a slack variable with ℓ-dimensional degree. Feature map function Φ maps the original feature into a high-dimensional kernel space F by using some simple kernel function

$$\kappa\left(\mathbf{x},\mathbf{y}\right)=\left\langle\Phi\left(\mathbf{x}\right),\Phi\left(\mathbf{y}\right)\right\rangle. \tag{12}$$

Usually we use Gaussian kernel function [41], [42]

$$\kappa\left(\mathbf{x},\mathbf{y}\right)=e^{-\|\mathbf{x}-\mathbf{y}\|^2/c}. \tag{13}$$

Then, the non-linear decision boundary $f\left(\cdot\right)$ can be written as:

$$\begin{aligned} f(\mathbf{x}) &= \operatorname{sgn}\left(\left(w\cdot\Phi\left(\mathbf{x}_i\right)\right)-\rho\right) \\ &= \operatorname{sgn}\left(\sum_{i=1}^{n}\alpha_i\kappa\left(\mathbf{x},\mathbf{x}_i\right)-\rho\right), \end{aligned} \tag{14}$$

which returns -1 for abnormal data and $+1$ elsewhere. Then the coefficient $\boldsymbol{\alpha}$ is found as the solution of the dual problem:

$$\min_{\boldsymbol{\alpha}} \quad \frac{1}{2}\sum_{ij}\alpha_i\alpha_j\kappa\left(\mathbf{x}_i,\mathbf{x}_j\right) \tag{15}$$

$$\text{subject to} \quad 0\leq\alpha_i\leq\frac{1}{\nu\ell} \tag{16}$$

$$\text{and } \sum_i\alpha_i=1. \tag{17}$$

Suppose in each iteration, there are k devices participating in the training. The local model update sent by each device is $\boldsymbol{g}_i$, $i = 1, \ldots, k$. We use $\boldsymbol{G} = \{\boldsymbol{g}_1, \ldots, \boldsymbol{g}_k\}$ to denote the set of all local model updates received in server. After performing anomaly detection with the One-Class SVM algorithm on all elements of set $\boldsymbol{G}$, the anomalies are split into the set $\boldsymbol{G}'_m$, where $\boldsymbol{G}'_m \subset \boldsymbol{G}$.

FAA-FL allows greater compatibility with various anomaly detection algorithms, and one can also choose other suitable ones to implement.

C. Privacy-Preserving Anomaly Verification

The variance of local model updates is large in the distributed learning scenario (i.e., local model updates in federated learning), because the training data in each device is non-IID. Some benign local updates may thus be misclassified into the set of potential malicious local model updates $\boldsymbol{G}'_m$. Note that benign devices calculate the model updates based on local training data, while malicious devices do not. True malicious local model updates can be easily identified. By assuming $\boldsymbol{g}_i \in \boldsymbol{G}'_m$, D_i is the local training data set of device i, $\boldsymbol{g}'_i$ is calculated on the server based on D_i. If $\boldsymbol{g}'_i \neq \boldsymbol{g}_i$, $\boldsymbol{g}_i$ is true malicious; otherwise benign. The main challenge of this approach is that we cannot directly compute with the local training dataset due to privacy concerns. One approach is to calculate with encrypted data, which is related to the privacy-preserving machine learning domain. Secure multi-party computation (SMC) [43] and homomorphic encryption (HE) [44] are two widely adopted approaches in literature. However, neither of these cryptographic approaches provides an efficient solution in practical.

Recently, the *Functional Encryption* scheme showed its efficiency and practicality in privacy-preserving machine learning [22], [30], [45]. From this aspect, we choose the emerging Functional Encryption scheme to overcome the challenge.

Functional Encryption (FE): Enables one to learn a function of what the ciphertext is encrypting, without revealing the plaintext. To be concrete, $Enc\,(x)$ is the ciphertext of plaintext x encrypted by the master public key mpk, and sk_f is a secret key generated with the function $f\,(\cdot)$ and master secret key msk. The value $f\,(x)$ can be directly calculated with the secret key sk_f:

$$f(x) \leftarrow Dec\,(mpk, sk_f, Enc\,(x))\,, \tag{18}$$

where $Dec\,(\cdot,\cdot)$ is a decryption function. Note that the output calculated by secret key sk_f and ciphertext $Enc\,(x)$ is $g^{f(x)}$, where g is the generator in mpk for encryption. Therefore, to obtain the final decryption result $f(x)$, mpk is also required to derive the discrete logarithm in basis g.

There are several types of FE mechanisms in literature, such as identity-based encryption scheme [25], attribute-based encryption scheme [26] and inner product functional encryption (IPFE) scheme [27], etc. We choose IPFE because the computation of model update is similar to calculating the inner product between two vectors. Below is the detailed analysis.

Suppose a neural network model has l layers ($l \geq 3$), and $\mathbf{A}^j$ is the output of the j-th layer ($j \in [1, l]$). In forward

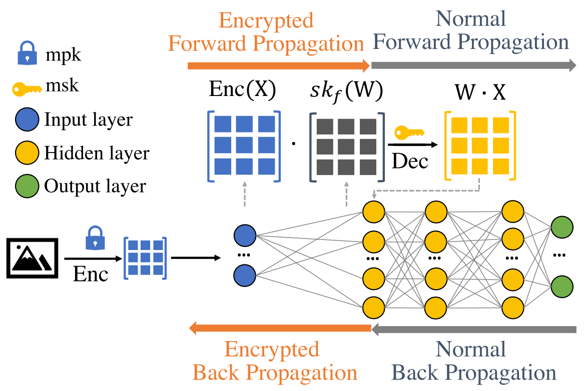


Fig. 3. Training on encrypted data with IPFE.

propagation, we have:

$$\mathbf{A}^j = \theta^j\left(\mathbf{Z}^j\right), \tag{19}$$

$$\mathbf{Z}^j = \mathbf{W}^j \cdot \mathbf{A}^{j-1} + \mathbf{b}^j, \tag{20}$$

where $\theta^j\,(\cdot)$, $\mathbf{W}^j$ and $\mathbf{b}^j$ is the activation function, weights and bias of the j-th layer, respectively. Let E be the loss function, and $\nabla\mathbf{W}^j$ is the update of $\mathbf{W}^j$. In backward propagation, we have:

$$\begin{aligned}\nabla\mathbf{W}^j &= \frac{\partial E}{\partial \mathbf{W}^j} = \frac{\partial E}{\partial \mathbf{A}^j}\cdot\frac{\partial \mathbf{A}^j}{\partial \theta^j}\cdot\frac{\partial \theta^j}{\partial \mathbf{Z}^j}\cdot\frac{\partial \mathbf{Z}^j}{\partial \mathbf{W}^j}\\ &= \frac{\partial E}{\partial \mathbf{A}^j}\cdot\frac{\partial \mathbf{A}^j}{\partial \theta^j}\cdot\frac{\partial \theta^j}{\partial \mathbf{Z}^j}\cdot\mathbf{A}^{j-1}.\end{aligned} \tag{21}$$

Let $\mathbf{X}$ be the input of the neural network model, and we have $A^0 = \mathbf{X}$. From (19), (20) and (21), we can find only the computation of $\mathbf{A}^1$ and $\nabla\mathbf{W}^1$ is based on $\mathbf{X}$. Both computations are dot product of two matrices, which can be transformed into the inner product of multiple vectors. Therefore, the IPFE scheme is suitable for our scenario. Figure 3 shows the process of training on encrypted data with IPFE. Note that we do not encrypt the label of each data in the same way as it may not be privacy-sensitive as the data itself. The privacy of labels can be preserved in a simple way like one-hot encoding and randomly mapping as in [30].

For IPFE, the most time-consuming operation is the computation of discrete logarithm. We can solve it by pre-computing as processed in [22]. We also apply *All or Nothing Transform (AONT)* [46] to the input data to further improve the computation and communication efficiency of IPFE.

AONT: Converts data into an encoded format and one cannot invert the encoded format back unless all of the encoded output is known. Linear AONT [47] is a linear transform that can maintain the property of AONT while being able to further reduce the computational complexity.

For $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)^T$, using linear AONT matrix $\mathbf{M}$, we can get the transformed data $\mathbf{U} = \mathbf{M}\cdot\mathbf{X}$.Then partially encrypting $\mathbf{U}$, we have the ciphertext $\mathbf{CT} = (\mathbf{u}_1, \ldots, \mathbf{u}_{n-1}, Enc\,(\mathbf{u}_n))^T$. One can learn nothing of $\mathbf{X}$ unless they know all the information of $\mathbf{U}$. Compared to the former approach, which needs n times encryption and decryption computations, respectively, only one encryption and one decryption computation are performed after AONT.

The procedure of anomaly verification is shown in Algorithm 1. Briefly, it includes three steps: 1) For each

Algorithm 1 AnomalyVerification($\boldsymbol{g}_i, \boldsymbol{w}, \alpha, D_i$)

```
Output: Verification result: $VR$;
1  Trusted third party:
2  $(mpk, msk) \leftarrow \texttt{Setup}(1^\lambda, 1^\eta)$,
3  $sk_f \leftarrow \texttt{KeyDer}(msk, f)$
4  Server side:
5  Request client $C_i$ to send encrypted training data for verification.
6  Client $C_i$:
7  $\mathbf{U}_i \leftarrow AONT(D_i)$,
8  $\mathbf{CT}_i \leftarrow (\mathbf{u}_1, \ldots, \mathbf{u}_{n-1}, \texttt{Encrypt}(mpk, \mathbf{u}_n))^T$,
9  Send $\mathbf{CT}_i$ to server.
10 Server side:
11 Once received $\mathbf{CT}_i$,
12 $\boldsymbol{g}'_i \leftarrow \texttt{Decrypt}(mpk, \mathbf{CT}_i, sk_f, \boldsymbol{w}, \alpha)$,
13 if $\boldsymbol{g}'_i == \boldsymbol{g}_i$ then
14     $VR \leftarrow False$;
15 else
16     $VR \leftarrow True$.
17 return $VR$
```

$\boldsymbol{g}_i \in \boldsymbol{G}'_m$, the server requests the suspected device i to send back the encrypted training data; 2) Each requested device transforms D_i to the partially encrypted data set CT_i and send CT_i back to the server; 3) Once receiving the encrypted data CT_i, the server can get the model update $\boldsymbol{g}'_i$ with the FE method, and compares with $\boldsymbol{g}_i$. If $\boldsymbol{g}'_i$ is the not the same as $\boldsymbol{g}_i$, then $\boldsymbol{g}_i$ is truly malicious, otherwise fake malicious. After all the data in $\boldsymbol{G}'_m$ are verified, we remove all fake malicious local model updates from $\boldsymbol{G}'_m$. Then we get the set $\boldsymbol{G}_m$, in which all the local model updates are true malicious.

Remark: The malicious devices may intend to escape the verification with well-designed data, but it is impossible. The main reason is that it is very difficult to recover the training data only with the gradient. Most of the work in literature tries to recover the input training data by solving the optimization problem that enabling the estimated training data as much similar as the real one [48]–[50]. Therefore, the gradient calculated based on the estimated training data is not the same as the original gradient, and the malicious devices fail to escape the anomaly verification.

D. Anomaly Removal

We know that malicious local model updates can hurt global model accuracy and convergence. We should remove them before aggregation. We can easily get the benign model updates set $\boldsymbol{G}_b$ by eliminating all elements of $\boldsymbol{G}_m$ from $\boldsymbol{G}$ and then perform the commonly used aggregation rule FedAvg to aggregate all local model updates in $\boldsymbol{G}_{\boldsymbol{b}}$.

Algorithm 2 shows our complete FAA-DL framework. In each iteration, FAA-DL performs three steps. In Step I, the server broadcasts the latest global model to participated clients. In Step II, each client computes local model update based on the global model and their own data. Then send it back to the server. In Step III, FAA-DL first performs anomaly detection on the set of received local model updates and splits the anomalies out, and then executes anomaly verification to further identify true anomalies. Finally, FAA-DL removes all true anomalies from the local model updates set and applies FedAvg algorithm on the remaining set.

Algorithm 2 FAA-DL

```
Input: Number of participated clients: $n$; Local training data of client $i$: $D_i$; Number of global iterations: $R$ Number of selected clients: $k$; Set of local model update: $\boldsymbol{G} = \{\boldsymbol{g}_1, \ldots, \boldsymbol{g}_k\}$; Learning rate: $\alpha$; Proportion of malicious client: $\beta$.
Output: Global model: $\boldsymbol{w}$;
1  $\boldsymbol{w} \leftarrow$ random initialization.
2  for $r = 1, 2, \ldots, R$ do
3      // Step I: Global model broadcasting
4      The server randomly selects $k$ clients from $n$ clients and sends them $\boldsymbol{w}$.
5      // Step II: Local model training
6      Client side:
7      for $i = 1, 2, \ldots, k$ do
8          $\boldsymbol{g}_i = \texttt{ModelUpdate}(\boldsymbol{w}, D_i)$,
9          Send $\boldsymbol{g}_i$ to server.
10     // Step III: Global model aggregation
11     Server side:
12     $\boldsymbol{G}'_m \leftarrow \texttt{AnomalyDetection}(\boldsymbol{G}_m, \beta)$,
13     for $\boldsymbol{g}_i \in \boldsymbol{G}'_m$ do
14         $VR_i \leftarrow \texttt{AnomalyVerification}(\boldsymbol{g}_i, \boldsymbol{w}, \alpha, D_i)$
           if $VR_i == True$ then
15             $\boldsymbol{G}_m.\texttt{add}(\boldsymbol{g}_i)$
16     $\boldsymbol{G}_b \leftarrow \texttt{AnomalyRemoval}(\boldsymbol{G}_m, \boldsymbol{G})$,
17     $\boldsymbol{g} \leftarrow \texttt{FedAvg}(\boldsymbol{G}_b)$,
18     $\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha \cdot \boldsymbol{g}$
```

V. Analysis

In this section, we provide theoretical analysis to show that FAA-DL can achieve the goals proposed in Section III-C. First, we prove the fidelity of FAA-DL. Then, we give the robustness proof and the convergence rate of FAA-DL. Finally, we analyze the time complexity of FAA-DL to show its efficiency.

Our problem is formulated based on federated learning system which is a promising distributed learning system. Suppose there are K devices participating in the global model training. Let F_i be the loss function of device i, and F is the global loss function. The optimization problem of federated learning is $\min_{\mathbf{w}}\{F(\mathbf{w}) \triangleq \frac{1}{K}\sum_{i=1}^{K} F_i(\mathbf{w})\}$, where $\mathbf{w}$ is the model parameter.

We make the following assumptions: Assumptions 1 and 2 are standard. Typical examples are the ℓ_2-norm regularized linear regression, logistic regression and softmax classifier.

Assumption 1: F_i is L-smooth: for all $\mathbf{w}_1$ and $\mathbf{w}_2$, $F_i(\mathbf{w}_1) \leq F_i(\mathbf{w}_2) + (\mathbf{w}_1 - \mathbf{w}_2)^T \nabla F_i(\mathbf{w}_2) + \frac{L}{2}\left\|\mathbf{w}_1 - \mathbf{w}_2\right\|_2^2$.

Assumption 2: F_i is μ-strongly convex: for all $\mathbf{w}_1$ and $\mathbf{w}_2$, $F(\mathbf{w}_1) \geq F_i(\mathbf{w}_2) + (\mathbf{w}_1 - \mathbf{w}_2)^T \nabla F_i(\mathbf{w}_2) + \frac{\mu}{2}\left\|\mathbf{w}_1 - \mathbf{w}_2\right\|_2^2$.

As non-IID exists in federated learning, we make the following assumptions as in [51], [52].

Assumption 3: Let ξ^i be sampled from the i-th device's local data uniformly at random. The variance of stochastic gradients in each device is bounded: $\mathbb{E}\left\|\nabla F_i(\mathbf{w}^i,\xi^i)-\nabla F_i(\mathbf{w}^i)\right\|^2 \le \sigma_i^2$ for $i=1,\dots,K$.

Assumption 4: The expected squared norm of stochastic gradients is uniformly bounded, i.e., $\mathbb{E}\left\|\nabla F_i(\mathbf{w}^i,\xi^i)\right\|^2 \le G^2$ for all $i=1,\dots,K$.

A. Fidelity Analysis

Our goal of fidelity is that the method should guarantee the learned global model to be as accurate as the global model learned by the baseline (FedAvg) when there is no attack. We have the following Lemma 1.

Lemma 1: Let $\mathbf{w}_F$ be the global model learnt by FAA-DL and $\mathbf{w}_A$ be the global model learnt by FedAvg. When there is no attack, the accuracy of $\mathbf{w}_F$ is equal to the accuracy of $\mathbf{w}_A$.

Proof: It is trivial to prove Lemma 1. Specifically, in each iteration of FAA-DL, all the local model updates are detected by the lightweight anomaly detection algorithm first, and then anomaly verification is performed on the detected potential malicious local model updates set. As there is no attack, the proportion of malicious local model updates is 0. No matter how many local model updates are detected to be potentially malicious through the anomaly detection algorithm, the set of verified malicious local model updates is empty. Finally, all local model updates are fed into the aggregator to perform averaged aggregation. FAA-DL is then reduced to FedAvg, and its accuracy is the same as FedAvg under no attack. Therefore, the fidelity of FAA-DL is guaranteed. □

B. Robustness Analysis

Robustness means no matter how strong the local model poisoning attack is, the accuracy of the learned global model should approximate to the value of baseline. The baseline is defined in Section III-C, which is FedAvg under no attacks. Before the robustness analysis, we first introduce a proposition.

Proposition 1: Let the last M of K local model updates be the malicious local model updates removed through FAA-DL, and thus the first $K-M$ local model updates are fed into aggregation. We have the expected squared norm of the aggregated local model updates to be bounded, i.e., $\mathbb{E}\left\|\sum_{i=1}^{K-M}\alpha_i\nabla F_i(\mathbf{w}^i,\xi^i)\right\|^2 \le G^2$ for all $i=1,\dots,K-M$, where α_i is the aggregation weight and $\sum_{i=1}^{K-M}\alpha_i=1$.

Proof: From Assumption 4, we know the expected squared norm of gradients is bounded by G^2 for each device. Thus, we can derive the upper bound of the expected squared norm of the aggregated local model updates as below:

$$\begin{aligned}\mathbb{E}\left\|\sum_{i=1}^{K-M}\alpha_i\nabla F_i(\mathbf{w}^i,\xi^i)\right\|^2 &\le \sum_{i=1}^{K-M}\alpha_i\mathbb{E}\left\|\nabla F_i(\mathbf{w}^i,\xi^i)\right\|^2\\ &\le \sum_{i=1}^{K-M}\alpha_i G^2\\ &= G^2. \end{aligned} \tag{22}$$

The first inequality is derived according to the Jensen's inequality. Specifically, since the squared norm function is convex and we have $\sum_{i=1}^{K-M}\alpha_i=1$, based on the Jensen's inequality, we can obtain $\left\|\sum_{i=1}^{K-M}\alpha_i\nabla F_i(\mathbf{w}^i,\xi^i)\right\|^2 \le \sum_{i=1}^{K-M}\alpha_i\left\|\nabla F_i(\mathbf{w}^i,\xi^i)\right\|^2$. Take expectation on both side of the inequality, we can get the first inequality of (22). And the second inequality is derived according to Assumption 4. Finally, we get Proposition 1. □

We note that there may exist false benign local updates in the aggregation process due to the limitation of anomaly detection methods. As the false benign local updates follow the distribution of the true benign local updates, Assumption 4 still holds for the false benign local updates, and thus the upper bound analyzed before will not be influenced by the false benign local updates.

Based on Proposition 1, we can derive Theorem 1 as follows.

Theorem 1: Let Assumptions 1 to 4 hold and R is the number of local iterations. Let $\kappa=\frac{L}{\mu}$, $\gamma=\max\{8\kappa,R\}$ and $\Delta_1=\mathbb{E}\left\|\mathbf{w}_1-\mathbf{w}^*\right\|^2$, the distance of loss function value between global model learned in FAA-DL and optimal model learned in FedAvg is upper bounded. We have

$$\mathbb{E}\left[F(\mathbf{w}_T)\right]-F^* \le \frac{\kappa}{\gamma+T-1}\left(\frac{\beta}{1-\beta}\cdot A+\frac{\mu\gamma}{2}\Delta_1+\frac{2B}{\mu}\right), \tag{23}$$

where

$$A=\frac{8R^2G^2}{\mu(K-1)}, \tag{24}$$

$$B=\sum_{i=1}^{K}\frac{\sigma_i^2}{K^2}+6L\Gamma+8(R-1)^2G^2. \tag{25}$$

Proof: Let β be the proportion of malicious local model updates removed through FAA-DL, and $(1-\beta)K$ local model updates are aggregated by FedAvg. Thus the convergence of FAA-DL is guaranteed by FedAvg. Combined with Proposition 1, Equation (26) is derived from Theorem 3 in [53] with replacing N and K of Theorem 3 in [53] to K and $(1-\beta)K$, respectively. Thus, we have

$$\begin{aligned}\mathbb{E}\left[F(\mathbf{w}_T)\right]-F^* &\le \frac{\kappa}{\gamma+T-1}\left(\frac{2}{\mu}\cdot\frac{K-(1-\beta)K}{K-1}\right.\\ &\qquad\left.\cdot\frac{4R^2G^2}{(1-\beta)K}+\frac{\mu\gamma}{2}\Delta_1+\frac{2B}{\mu}\right)\\ &\le \frac{\kappa}{\gamma+T-1}\left(\frac{\beta}{1-\beta}\cdot\frac{8R^2G^2}{\mu(K-1)}\right.\\ &\qquad\left.+\frac{\mu\gamma}{2}\Delta_1+\frac{2B}{\mu}\right)\\ &\le \frac{\kappa}{\gamma+T-1}\left(\frac{\beta}{1-\beta}\cdot A+\frac{\mu\gamma}{2}\Delta_1\right.\\ &\qquad\left.+\frac{2B}{\mu}\right),\end{aligned} \tag{26}$$

TABLE I
COMPUTATIONAL COMPLEXITY OF FAA-DL IN TERMS OF MATHEMATICAL OPERATIONS FOR EACH STAGE IN EACH ITERATION

Mathematical Operations / Stage	Server			Device		
	Addition	Multiplication	Exponentiation	Addition	Multiplication	Exponentiation
Anomaly Detection	$3M \cdot N$	$(2M+2) \cdot N$	N	–	–	–
Anomaly Verification	–	$K \cdot (W+1)$	–	$B \cdot H \cdot (H-1)$	$B \cdot \left(H^3+H+1\right)$	$B \cdot (3H+2)$
Anomaly Removal	–	–	–	–	–	–

where

$$A = \frac{8R^2G^2}{\mu(K-1)}, \tag{27}$$

$$B = \sum_{i=1}^{K} \frac{\sigma_i^2}{K^2} + 6L\Gamma + 8(R-1)^2G^2. \tag{28}$$

□

From (26), we can find that FAA-DL will asymptotically converge to the global optimum with the rate $\mathcal{O}(\frac{1}{T})$, where T is the training iterations.

From Theorem 1, we can find that the upper bound will always approximate to 0 even the value of β is largely increased ($0 \leq \beta < 0.5$), with $T \rightarrow \infty$. It means FAA-DL is robust enough to the strong attacks with a convergence guarantee.

C. Efficiency Analysis

The objective of efficiency is to avoid extra computations on resource-limited edge devices as much as possible. To measure the efficiency of FAA-DL, we consider analyzing its time complexity and computational complexity.

1) Time Complexity: In each iteration of FAA-DL, only a small amount of potential malicious devices need to perform extra computation. The computation contains a linear transformation and partial encryption. Both operations have a time complexity of $\mathcal{O}(1)$, and thus the extra computation is negligible. For the server, extra computation comes from anomaly detection and verification executed before aggregation. Specifically, the anomaly detection step is executed with the One-Class SVM algorithm, and its computation complexity is well known as $\mathcal{O}\left(n^3\right)$ [54]. Another operation performed by the server is one iteration training, which has the time complexity $\mathcal{O}(1)$. Therefore, the time complexity of overall computations that occurred on the server is $\mathcal{O}\left(n^3\right)$. Considered the strong computation capacity of the server, the extra computation is relatively small. This analysis result is also verified by our experimental result in Section VII-B.

2) Computational Complexity: We consider the number of mathematical operations required at each stage to show the computational complexity of FAA-DL. For each iteration, there are three types of mathematical operations: addition, multiplication, and exponentiation. The total number of each mathematical operation required for each stage is summarized in Table I. In the anomaly detection stage, extra operations only occur in the server. With formula (14), each uploaded local model update is classified as potentially malicious or normal, and the total number of addition, multiplication, and exponentiation for N participated devices are $3M \cdot N$, $(2M+2) \cdot N$ and N, respectively, where M is the number of parameters in the training model. In the anomaly verification stage, both devices and the server have additional operations. For each potential malicious device, local training data is transformed with AONT and then encrypted by functional encryption. According to the functions of the two operations, we can get the related number of addition, multiplication,and exponentiation are $B \cdot H \cdot (H-1)$, $B \cdot \left(H^3+H+1\right)$ and $B \cdot (3H+2)$, respectively, where B is the amount of local training data which has size $H \times H$. For the server, each uploaded encrypted local data is required to perform one training iteration to obtain the gradient. Besides the W multiplication executed in the training process, one extra multiplication is performed for decryption. Therefore, the total number of multiplication performed on the server is $K \cdot (W+1)$ where K is the number of potential malicious devices. In the anomaly removal stage, the server is only required to perform the simple removal operation which consumes negligible computational resources and we neglect the mathematical operations involved. From Table I, we can find that the additional computation for the device mainly relies on the amount of training data B and the data size H. And the additional computation for the server depends on the size of training model parameters M, the number of operations in the training model W, and the number of potentially malicious devices K. Our experimental result in Section VII-B can also confirm this analysis result.

VI. IMPLEMENTATION

We implement a prototype of FAA-DL with PyTorch, including the anomaly detection module, the anomaly verification module, and the anomaly removal module.

A. FAA-DL Client and Server

We rent an Amazon cloud as the server and implement the FAA-DL clients on Raspberry Pi 4 model B. There are 7 Raspberry Pi in the implemented system and each Raspberry Pi has 4GB of memory. The cloud server is equipped with an Intel Xeon Gold 5218 CPU with 256GB memory, a GeForce RTX 2080 Ti GPU with 11GB memory, and a 1T SSD.

B. Anomaly Detection Module

The server performs the anomaly detection algorithm on all local model updates received from the clients. The anomaly detection module is implemented based on the One-Class SVM algorithm in PyOD [55], a popular open-source toolbox for performing scalable outlier detection. The kernel of One-Class SVM is implemented by the radial basis function, and the upper bound on the fraction of anomalies is set to 0.5.

C. Anomaly Verification Module

The clients and the server collaboratively execute the anomaly verification with the help of the functional encryption scheme. The functional encryption scheme in our framework is based on [27], which is a simple and practical inner product functional encryption method. The underlying crypto system is implemented using the Charm library [56], which is a Python-based crypto toolkit and its experimental calculations rely on the GMP library. We rent another same Amazon cloud to be the Trusted Third Party (TTP) for managing the keys for encryption and decryption. Before encrypting the training data of the potential malicious client, the data is transformed with a linear AONT matrix. The linear transform matrix applied in our framework is

$$\mathbf{M} = \begin{pmatrix} 1 & 0 & \cdots & 0 & 1 \\ 0 & 1 & \cdots & 0 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 1 \\ 1 & 1 & \cdots & 1 & \lambda \end{pmatrix}, \tag{29}$$

where $\lambda = 2$ and the dimension of $\mathbf{M}$ is equal to the width of the input image (i.e., the width of input image in our implementation is 28).

D. Anomaly Removal Module

The anomaly removal module is implemented on the server. After all the anomalies are verified, the remover drops them out of the whole local model updates set. Then the remaining normal local model update is sent to the aggregator for producing the new global model.

VII. Evaluation

In this section, we evaluate FAA-DL with image classification tasks on two datasets, MNIST and Fashion-MNIST. We demonstrate the better performance of our framework by comparing it with several baseline defense approaches as well as an ideal baseline without attack.

A. Evaluation Setup

1) Training Dataset: We employ two widely used public dataset MNIST [57] and Fashion-MNIST [58], both of which contain $60,000$ training examples and $10,000$ testing examples. The MNIST dataset is comprised of 10-class handwriting digits, and Fashion-MNIST consists of fashion products from 10 categories. To simulate the non-IID setting, we follow [34] to sort data examples based on the digit labels, and each device is equally assigned data examples with only two digits.

2) Classification Models: The model applied on MNIST is a CNN with two 5×5 convolution layers (the first with 32 channels, the second with 64, each followed with 2×2 max pooling), a fully connected layer with 320 units and ReLu activation, and a final softmax output layer. Fashion-MNIST follows the same structure as MNIST, besides adding batch normalization after convolution layers.

3) Attack Models: For local model poisoning attack, there exist several attack algorithms in the literature, we choose three popular algorithms in our paper to defend, and the detailed introduction is as follows:

- *Gaussian Noise Attack:* Gaussian noise attack [31] is an untargeted attack. The adversary manipulates the local model weights by adding Gaussian noise. It is a simple and effective attack to hurt model performance. The noise follows the distribution $\mathcal{N}(0, \sigma^2)$.
- *Sign-Flipping Attack:* Sign-flipping attack [33] flips the sign of model weights. To be specific, the true model weight of a malicious device i is $\boldsymbol{w}_i$, and after flipping the sign by a negative constant ϵ, malicious weight $\hat{\boldsymbol{w}}_i = \epsilon \boldsymbol{w}_i$ will be sent to the server.
- *ALIE Attack:* ALIE [24] attack commands malicious devices to send $\hat{\boldsymbol{\mu}} + z \cdot \hat{\boldsymbol{\sigma}}$ to server, where $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\sigma}}$ are the estimated coordinate-wise mean and standard deviation of all the malicious devices' local model updates and z is a hyper-parameter which was tuned in [24].

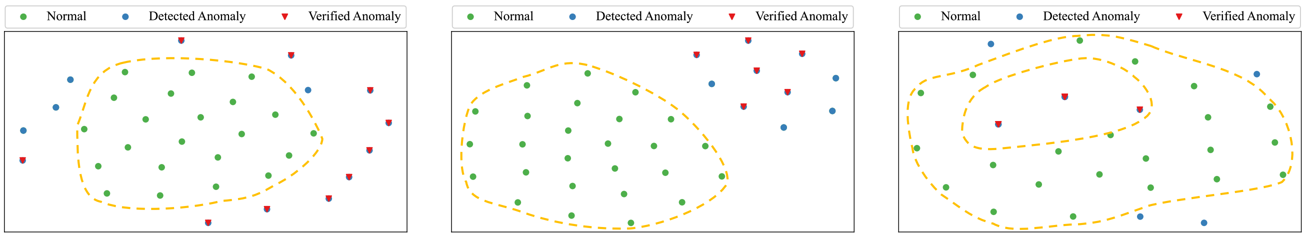
4) Defense Benchmark: To evaluate the performance of our framework, we compare to four popular passive defense approaches: FedAVG (AVG) [53], Trimmed Mean (TM) [10], GeoMed (GM) [11] and Krum (KR) [8]. To analyze the contribution of each module in FAA-DL, we defend the attacks with anomaly detection module (OC) and anomaly verification module (FE) separately. We also treat OC as the proactive defense approach. FedAvg under no attacks is compared as the ideal baseline.

5) Experiment Environment: Our framework is implemented with PyTorch and tested on a laptop equipped with an Intel Core i7 processor, 16GB memory, running with macOS. For the federated learning process, 20 clients are randomly chosen from 100 clients in each communication round, and each client executes one epoch with batch size 50.

B. Results

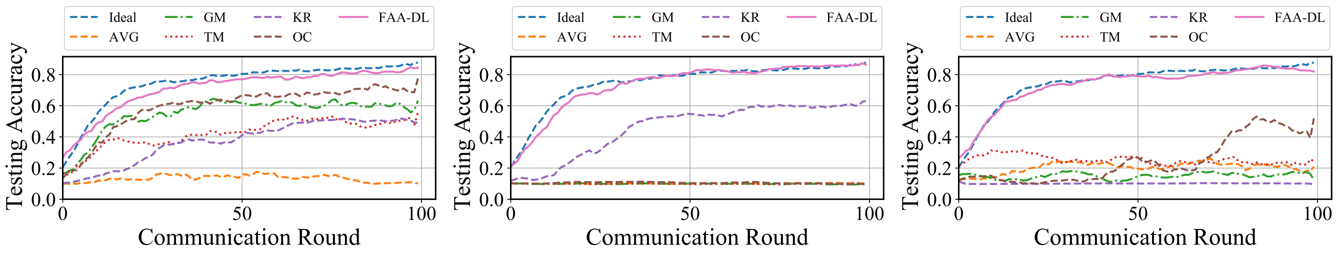
We evaluate the FAA-DL from two aspects, the first aspect is macro analysis including effectiveness analysis, accuracy analysis, robustness analysis, and efficiency analysis, and the second aspect is micro analysis including analyzing the impact of anomaly detection module and anomaly verification module, respectively. Experimental results show that the goals proposed in Section III-C are all accomplished.

1) Effectiveness Analysis: FAA-DL aims to remove all the malicious local model updates from aggregation to proactively defend the local model poisoning attack. To show the effectiveness of FAA-DL, we illustrate the process result of FAA-DL when defending different attacks in Fig. 4. Particularly, in the anomaly detection step, to enable the true malicious devices to be identified as much as possible, we set the proportion of estimated malicious devices in the anomaly detection algorithm to be larger than the real one (i.e., we can set the proportion of estimated malicious devices in OCSVM to be 0.5 as the threat model of our paper assumes the proportion of malicious devices is less than 0.5). We classify the local model updates according to the decision boundary (the yellow dashed curve in Fig. 4) obtained through the anomaly detection algorithm



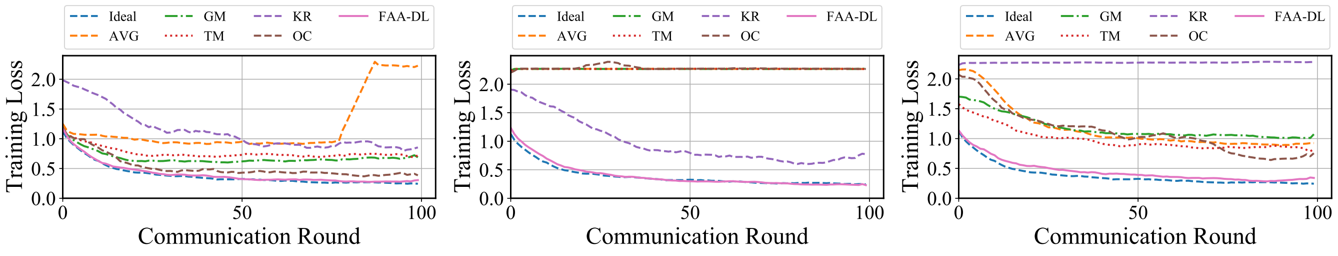
(a) Gaussian attack (b) Sign-flipping attack (c) ALIE attack

Fig. 4. Illustration of the process when defending different attacks with FAA-DL.



(a) Gaussian attack (b) Sign-flipping attack (c) ALIE attack

Fig. 5. The accuracy of defense to different attacks with different methods.



(a) Gaussian attack (b) Sign-flipping attack (c) ALIE attack

Fig. 6. The loss of defense to different attacks with different methods.

(OCSVM). If the local model update is within the boundary, then it is classified to be normal, otherwise, it is classified to be abnormal. Upon almost all the potential malicious local model updates are filtered out, anomaly verification is performed to identify the truly malicious local model updates. For example, in Fig. 4a, the normal local model updates (green point) are identified in the anomaly detection step, then each potential malicious local model update (blue point) is checked and finally, the truly malicious one can be verified.

2) Accuracy Analysis: The accuracy result of FAA-DL under different attacks on MNIST is shown in Fig. 5. There are three key takeaways from these results. First, our proposed FAA-DL achieves the best performance in all settings. The accuracy of FAA-DL can reach up to 89.95%, which outperform TM, OC, AVG, KR, GM with an accuracy improvement of $1.0\times, 0.36\times, 1.55\times, 3.81\times, 2.03\times$. Second, the performance gap of FAA-DL is within 0.92%–2.48% of the ideal baseline across all datasets. This reveals little room exists for FAA-DL to improve in these scenarios. Third, the performance of proactive defense methods outperforms that of passive defense methods. More specifically, for passive methods in defense benchmark, the best accuracy they can reach is around 60%, and our framework can reach 90%. While for proactive defense method OC, which only performs anomaly detection, and it fails when the attack is fine crafted such as the ALIE attack showed in Figure 5c, where our method is still effective.

We also compare the training loss of each defense method in Figure 6, the convergence rate of FAA-DL is the same as FedAvg under no attack, regardless of the type of attack. The top-1 accuracy of each method to the three attacks on MNIST is listed in Table II. From which we can see that when there is no attack, the accuracy of FAA-DL is similar to FedAvg. When an attack exists (we set the proportion of malicious devices to 0.4), our method still performs as well as the ideal baseline. The same experiment is performed on Fashion-MNIST, and the result is showed in Table III. Obviously, FAA-DL shows better performance than other defense methods in defense benchmarks, no matter how strong the attack is.

3) Robustness Analysis: We also evaluate the robustness by comparing the accuracy of the learned global model when the number of malicious devices increased. The experimental result is showed in Figure 7, which is performed on MNIST. It shows that our framework can guarantee the accuracy of the

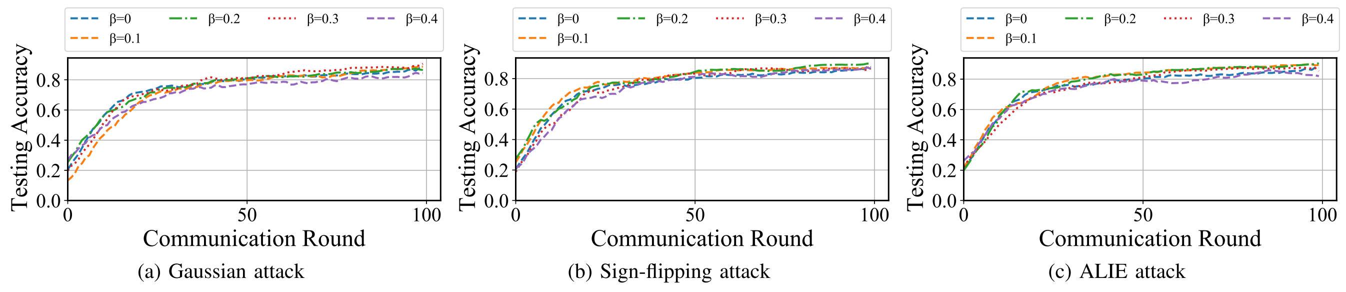


Fig. 7. Robustness of our framework to different attacks with different fraction of malicious devices (from 0.1 to 0.4).

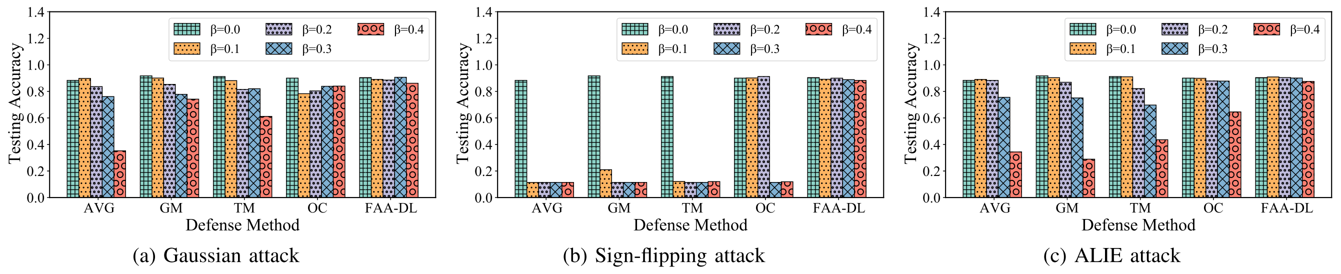


Fig. 8. Top-1 accuracy of different defense methods to different attacks with various fraction of malicious devices(from 0.1 to 0.4).

TABLE II
TOP-1 ACCURACY OF DIFFERENT DEFENSE METHODS TO VARIOUS ATTACKS (DATASET: MNIST)

Attack / defense	No attack	Gaussian noise	Sign flipping	ALIE
AVG	88.31%	35.23%	11.35%	34.34%
GM	91.71%	74.15%	11.35%	28.92%
TM	91.20%	61.16%	11.35%	43.58%
KR	53.92%	54.15%	58.15%	18.2%
FAA-DL	90.39%	86.12%	89.95%	87.5%

TABLE III
TOP-1 ACCURACY OF DIFFERENT DEFENSE METHODS TO VARIOUS ATTACKS (DATASET: FASHION-MNIST)

Attack / Defense	No attack	Gaussian noise	Sign flipping	ALIE
AVG	83.90%	75.87%	10.00%	79.69%
GM	84.52%	75.86%	10.00%	78.89%
TM	84.66%	74.4%	10.00%	77.19%
KR	56.61%	52.25%	52.79%	28.68%
FAA-DL	83.63%	77.84%	84.95%	80.98%

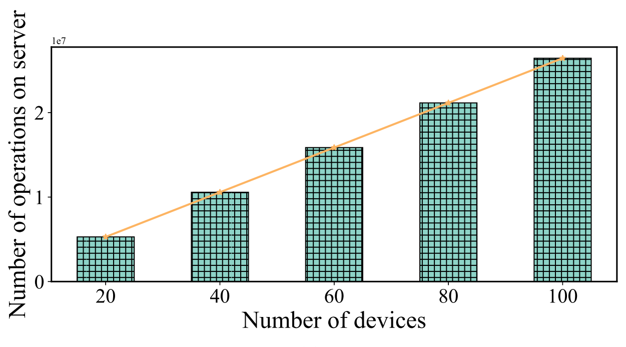
learned global model to be as accurate as the ideal baseline no matter how strong the attack is. The robustness evaluation results of passive and proactive defense methods are shown in Figure 8. For passive defense methods, such as Trimmed mean, when the proportion of Gaussian noise attacked devices increased from 0.1 to 0.4, the accuracy decreased from nearly 88% to less than 60%, while our method remains nearly the same accuracy as the ideal baseline. A similar result happens on the sign-flipping attack and the ALIE attack. When the strong attack occurs (i.e., the fraction of malicious devices is 0.4), FAA-DL improves the testing accuracy of the learned model by $1.44\times \sim 6.90\times$ compared to the FedAvg method with no defense, and $0.16\times \sim 1.00\times$ compared to the passive defense methods. It is obvious that the proactive method is more robust than the passive method.

Among the proactive methods, FAA-DL also performs the best, as shown in Figure 8. Specifically, FAA-DL outperforms OC with an improvement of 36%, respectively. For Figure 8, we can observe that the accuracy of FAA-DL can reach up to 89.95% under all kinds of attacks. This validates that FAA-DL can preserve accuracy even under strong poisoning attacks.

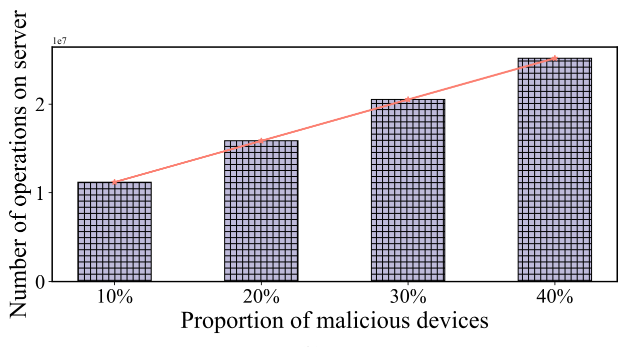
4) Efficiency Analysis: To measure the efficiency, we consider the time cost of each iteration in the training process. We test all passive and proactive defense methods with different attacks and the proportion of malicious devices is 0.2. Tables IV and V are the detailed result of MNIST and Fashion-MNIST, respectively, and the unit is seconds. From these results, we can find that the time cost of our method is approximate to the ideal baseline, which confirms our analysis in Section V-C.

Another observation is that the passive defense method Trimmed Mean is the most time-consuming among all tested defense methods for MNIST. The main reason is that it needs to handle each dimension of model parameters separately, while the number of dimensions is large in the CNN model. Moreover, as the time complexity of Krum is $\mathcal{O}\left(n^2 \cdot d\right)$ [8], it becomes the most time consuming when the dimension d of the CNN model increased dramatically, such as the model employed in Fashion-MNIST shown in Table V.

We also measure how the computational complexity varies in different situations to show the efficiency of FAA-DL. According to the analysis in Section V-C, the number of mathematical operations in the device is not affected by either the number of total participated devices or the number of malicious devices, and it mainly depends on the amount of local training data and the size of each data. Moreover,



(a)



(b)

Fig. 9. Computational complexity of FAA-DL effected by different factors. (a) is in terms of the number of participated devices, and (b) is in terms of the proportion of malicious devices.

TABLE IV
TIME COST IN EACH ITERATION OF DIFFERENT DEFENSE METHODS TO VARIOUS ATTACKS (DATASET: MNIST, UNIT: SECOND)

Defense / Attack	AVG	GM	TM	KR	OC	FAA-DL	FE
Gaussian noise	4.07	4.26	5.78	4.87	4.12	5.47	6.79
Sign flipping	4.26	4.22	5.75	4.84	4.13	5.42	6.97
ALIE	4.19	4.39	5.86	4.88	4.18	5.53	6.80

TABLE V
TIME COST IN EACH ITERATION OF DIFFERENT DEFENSE METHODS TO VARIOUS ATTACKS (DATASET: FASHION-MNIST, UNIT: SECOND)

Defense / Attack	AVG	GM	TM	KR	OC	FAA-DL	FE
Gaussian noise	4.09	4.44	6.77	7.54	4.08	5.25	6.37
Sign flipping	4.06	4.23	6.72	7.58	4.06	5.24	6.41
ALIE	4.08	4.30	6.74	7.53	4.12	5.29	6.62

the total number of mathematical operations in the server can be expressed as $N\cdot(5M+3)+K\cdot(W+1)$. Thus, the computational complexity of the server will be increase linearly when either the number of participated devices N or the proportion of malicious devices K increased, as shown in Fig. 9. We consider the server is equipped with a GeForce RTX 2080 Ti GPU which can perform 14.2 TFLOPs (10^{12} float point operation) per second, the time cost due to FAA-DL is then less than 10 milliseconds. Similarly, the local device with a Raspberry Pi 4B can execute 13.5 GFLOPs (10^{9} float point operations) per second, and the additional time cost is less than 1 millisecond. Therefore, the extra computation time caused by FAA-DL is negligible.

5) Impact of Anomaly Detection Module: FAA-DL designs a lightweight and unsupervised anomaly detection module to separate the potential malicious local model updates from the whole set. Its objective is to guarantee the efficiency of FAA-DL without extra overheads. The impact of this module is reflected in Table IV and V. Comparing to the baseline defense approach FedAvg, we can easily find that the total time cost increased about 64% if we defend the attacks by directly verifying all uploaded local model updates (FE), and the extra time cost will be decreased to around 27% after employing the anomaly detection module before verification (FAA-DL). Meanwhile, we can also observe that the time cost of the anomaly detection module (OC) approximates to FedAvg, which proved the designed anomaly detection module is lightweight.

6) Impact of Anomaly Verification Module: The aim of designing the anomaly verification module after anomaly detection is to enable the accuracy of the learned global model to be as accurate as of the ideal defense approach. We prove the effectiveness of the anomaly verification module by comparing the top-1 accuracy of the learned global model with the proactive defense approaches. As shown in Figure 8, OC and FAA-DL represent the proactive defense without and with anomaly verification module, respectively. The accuracy of the learned global model defended with OC approximates to the ideal defense when the attack is weak (i.e., the proportion of malicious clients is no more than 0.2), but decreased dramatically when the attack getting strong (i.e., sign-flipping attack with the proportion of malicious clients more than 0.2). While for FAA-DL, no matter how strong the local model poisoning attack is, the top-1 accuracy of the learned global model still approximates to the ideal defense. Therefore, it is essential to perform anomaly verification after the lightweight anomaly detection.

In conclusion, the numerous experimental results show that the goal of accuracy, robustness, and efficiency proposed in Section III-C is fully achieved. Our proactive federated anomaly analytics method demonstrates superior performance in defending local model poisoning attacks.

VIII. CONCLUSION

In this paper, we propose a federated anomaly analytics enhanced distributed learning framework to defend local model poisoning attacks proactively. We firstly split out the potential malicious local models through a lightweight and unsupervised anomaly detection algorithm. Then we verify anomalies via functional encryption method with privacy and high accuracy guarantee. We also apply All-or-Nothing transform to improve the efficiency of anomaly verification. Finally, the verified anomalies are removed from aggregation. The performance of FAA-DL is proved with theoretical analysis and experimental evaluation. The experimental results show that FAA-DL improves the testing accuracy of the learned global model under strong attacks for $1.44\times \sim 6.90\times$ compared to the FedAvg method with no defense, and $0.16\times \sim 1.00\times$ compared to the passive defense methods. Therefore, FAA-DL as a promising defense and a strong complement to existing defenses is effective in guaranteeing the accuracy and robustness of distributed learning systems under local model poisoning attacks.

REFERENCES

[1] L. U. Khan, M. Alsenwi, I. Yaqoob, M. Imran, Z. Han, and C. S. Hong, "Resource optimized federated learning-enabled cognitive Internet of Things for smart industries," *IEEE Access*, vol. 8, pp. 168854–168864, 2020.

[2] L. U. Khan *et al.*, "Federated learning for edge networks: Resource optimization and incentive mechanism," *IEEE Commun. Mag.*, vol. 58, no. 10, pp. 88–93, Oct. 2020.

[3] D. Chen *et al.*, "Matching-theory-based low-latency scheme for multitask federated learning in MEC networks," *IEEE Internet Things J.*, vol. 8, no. 14, pp. 11415–11426, Jul. 2021.

[4] P. Kairouz *et al.*, "Advances and open problems in federated learning," 2019, *arXiv:1912.04977*. [Online]. Available: http://arxiv.org/abs/1912.04977

[5] S. R. Pandey, N. H. Tran, M. Bennis, Y. K. Tun, A. Manzoor, and C. S. Hong, "A crowdsourcing framework for on-device federated learning," *IEEE Trans. Wireless Commun.*, vol. 19, no. 5, pp. 3241–3256, May 2020.

[6] M. Chen, H. V. Poor, W. Saad, and S. Cui, "Convergence time minimization of federated learning over wireless networks," in *Proc. IEEE ICC*, Dublin, Ireland, Jun. 2020, pp. 1–6.

[7] G. Zhu, Y. Du, D. Gunduz, and K. Huang, "One-bit over-the-air aggregation for communication-efficient federated edge learning: Design and convergence analysis," *IEEE Trans. Wireless Commun.*, vol. 20, no. 3, pp. 2120–2135, Mar. 2021.

[8] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Proc. NeurPIS*, Long Beach, CA, USA, Dec. 2017, pp. 118–128.

[9] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proc. AISTATS*, Aug. 2020, pp. 2938–2948.

[10] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *Proc. ICML*, Stockholm, Sweden, Jul. 2018, pp. 5650–5659.

[11] Y. Chen, L. Su, and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 1, no. 2, pp. 1–25, 2017.

[12] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "Fltrust: Byzantine-robust federated learning via trust bootstrapping," in *Proc. NDSS*, Feb. 2021, pp. 1–18.

[13] S. Li, Y. Cheng, W. Wang, Y. Liu, and T. Chen, "Learning to detect malicious clients for robust federated learning," 2020, *arXiv:2002.00211*. [Online]. Available: https://arxiv.org/abs/2002.00211

[14] D. Ramage. (May 2020). *Federated Analytics: Collaborative Data Science Without Data Collection*. [Online]. Available: https://ai.googleblog.com/2020/05/federated-analytics-collaborative-data.html

[15] A. Hard *et al.*, "Federated learning for mobile keyboard prediction," 2018, *arXiv:1811.03604*. [Online]. Available: http://arxiv.org/abs/1811.03604

[16] H. Jin, L. Jia, and Z. Zhou, "Boosting edge intelligence with collaborative cross-edge analytics," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2444–2458, Feb. 2021.

[17] K. Bonawitz *et al.*, "Towards federated learning at scale: System design," in *Proc. MLSys*, Stanford, CA, USA, Mar. 2019, pp. 1–5.

[18] E. Roth, H. Zhang, A. Haeberlen, and B. C. Pierce, "Orchard: Differentially private analytics at scale," in *Proc. OSDI*, Nov. 2020, pp. 1065–1081.

[19] W. Zhu, P. Kairouz, B. McMahan, H. Sun, and W. Li, "Federated heavy hitters discovery with differential privacy," in *Proc. AISTATS*, Aug. 2020, pp. 3837–3847.

[20] Z. Ji, Z. C. Lipton, and C. Elkan, "Differential privacy and machine learning: A survey and review," 2014, *arXiv:1412.7584*. [Online]. Available: http://arxiv.org/abs/1412.7584

[21] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A survey on homomorphic encryption schemes: Theory and implementation," *ACM Comput. Surv.*, vol. 51, no. 4, pp. 1–35, 2018.

[22] T. Ryffel, D. Pointcheval, F. Bach, E. Dufour-Sans, and R. Gay, "Partially encrypted deep learning using functional encryption," in *Proc. NeurIPS*, Vancouver, BC, Canada, Dec. 2019, pp. 1–21.

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

[24] G. Baruch, M. Baruch, and Y. Goldberg, "A little is enough: Circumventing defenses for distributed learning," in *Proc. NeurIPS*, Vancouver, BC, Canada, Dec. 2019, pp. 1–10.

[25] B. Waters, "Efficient identity-based encryption without random oracles," in *Proc. EUROCRYPT*, Aarhus, Denmark, May 2005, pp. 114–127.

[26] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. SP*, Berkeley, CA, USA, May 2007, pp. 321–334.

[27] M. Abdalla, F. Bourse, A. De Caro, and D. Pointcheval, "Simple functional encryption schemes for inner products," in *Proc. IACR*, Gaithersburg, MD, USA, Mar. 2015, pp. 733–751.

[28] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proc. EUROCRYPT*, Aarhus, Denmark, May 2005, pp. 457–473.

[29] D. Boneh, A. Sahai, and B. Waters, "Functional encryption: Definitions and challenges," in *Proc. TCC*, Providence, RI, USA, Mar. 2011, pp. 253–273.

[30] R. Xu, J. B. Joshi, and C. Li, "CryptoNN: Training neural networks over encrypted data," in *Proc. ICDCS*, Dallas, TX, USA, Jul. 2019, pp. 1199–1209.

[31] L. Li, W. Xu, T. Chen, G. B. Giannakis, and Q. Ling, "RSA: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets," in *Proc. AAAI*, Honolulu, HI, USA, Jan. 2019, pp. 1544–1551.

[32] A. Shafahi *et al.*, "Poison frogs targeted clean-label poisoning attacks on neural networks," in *Proc. NeurIPS*, Montréal, QC, Canada, Dec. 2018, pp. 1–8.

[33] Z. Wu, Q. Ling, T. Chen, and G. B. Giannakis, "Federated variance-reduced stochastic gradient descent with robustness to byzantine attacks," *IEEE Trans. Signal Process.*, vol. 68, pp. 4583–4596, 2020.

[34] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. AISTATS*, Fort Lauderdale, FL, USA, Apr. 2017, pp. 1273–1282.

[35] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," *ACM SIGMOD Rec.*, vol. 29, no. 2, pp. 93–104, Jun. 2000.

[36] H. Ren *et al.*, "Time-series anomaly detection service at Microsoft," in *Proc. ACM SIGKDD*, Anchorage, AK, USA, Aug. 2019, pp. 3009–3017.

[37] F. T. Liu, K. M. Ting, and Z. Zhou, "Isolation forest," in *Proc. IEEE ICDM*, Pisa, Italy, Dec. 2008, pp. 413–422.

[38] W. Chu and K. M. Kitani, "Neural batch sampling with reinforcement learning for semi-supervised anomaly detection," in *Proc. ECCV*, Glasgow, U.K., Aug. 2020, pp. 751–766.

[39] L. Ruff *et al.*, "Deep semi-supervised anomaly detection," in *Proc. ICLR*, Addis Ababa, Ethiopia, Apr. 2020, pp. 1–23.

[40] V. Vercruyssen, W. Meert, G. Verbruggen, K. Maes, R. Baumer, and J. Davis, "Semi-supervised anomaly detection with an application to water analytics," in *Proc. ICDM*, Singapore, Nov. 2018, pp. 527–536.

[41] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt, "Support vector method for novelty detection," in *Proc. NeurIPS*, Denver, CO, USA, Nov. 2000, pp. 582–588.

[42] B. Du, C. Liu, W. Zhou, Z. Hou, and H. Xiong, "Catch me if you can: Detecting pickpocket suspects from large-scale transit records," in *Proc. ACM SIGKDD*, San Francisco, CA, USA, Aug. 2016, pp. 87–96.

[43] A.-T. Tran, T.-D. Luong, J. Karnjana, and V.-N. Huynh, "An efficient approach for privacy preserving decentralized deep learning models based on secure multi-party computation," *Neurocomputing*, vol. 422, pp. 245–262, Jan. 2021.

[44] Y. Liu, Y. Kang, C. Xing, T. Chen, and Q. Yang, "A secure federated transfer learning framework," *IEEE Intell. Syst.*, vol. 35, no. 4, pp. 70–82, Jul. 2020.

[45] D. Wu, M. Pan, Z. Xu, Y. Zhang, and Z. Han, "Towards efficient secure aggregation for model update in federated learning," in *Proc. IEEE Globecom*, Taipei, China, Dec. 2020, pp. 1–6.

[46] R. L. Rivest, "All-or-nothing encryption and the package transform," in *Proc. FSE*, Haifa, Israel, Jan. 1997, pp. 210–218.

[47] D. R. Stinson, "Something about all or nothing (transforms)," *Des., Codes Cryptogr.*, vol. 22, no. 2, pp. 133–138, Mar. 2001.

[48] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning: Revisited and enhanced," in *Proc. ATIS*, Auckland, New Zealand, Jul. 2017, pp. 100–110.

[49] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Proc. NeurIPS*, Vancouver, BC, Canada, Canada, Dec. 2019, pp. 17–31.

[50] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, "Beyond inferring class representatives: User-level privacy leakage from federated learning," in *Proc. IEEE Conf. Comput. Commun.*, Paris, France, Apr. 2019, pp. 2512–2520.

[51] H. Yu, S. Yang, and S. Zhu, "Parallel restarted SGD with faster convergence and less communication: Demystifying why model averaging works for deep learning," in *Proc. AAAI*, Honolulu, HI, USA, Jan. 2019, pp. 5693–5700.

[52] S. U. Stich, "Local SGD converges fast and communicates little," in *Proc. ICLR*, New Orleans, LA, USA, May 2019, pp. 1–19.

[53] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of FedAvg on non-IID data," in *Proc. ICLR*, Addis Ababa, Ethiopia, Apr. 2020, pp. 1–26.

[54] Z. Wang, Z. Zhao, S. Weng, and C. Zhang, "Solving one-class problem with outlier examples by SVM," *Neurocomputing*, vol. 149, pp. 100–105, Feb. 2015.

[55] Y. Zhao, Z. Nasrullah, and Z. Li, "PyOD: A Python toolbox for scalable outlier detection," *J. Mach. Learn. Res.*, vol. 20, no. 96, pp. 1–7, Jan. 2019.

[56] J. A. Akinyele *et al.*, "Charm: A framework for rapidly prototyping cryptosystems," *J. Cryptograph. Eng.*, vol. 3, no. 2, pp. 111–128, 2013.

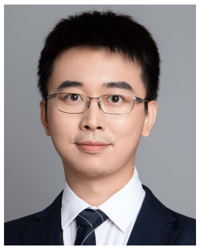
[57] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[58] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv:1708.07747*. [Online]. Available: http://arxiv.org/abs/1708.07747

Siping Shi received the B.S. degree in computer science from Sichuan University in 2014 and the M.S. degree in computer applied technology from the University of Chinese Academy of Sciences in 2017. She is currently pursuing the Ph.D. degree with The Hong Kong Polytechnic University. Her research interests include edge computing, federated learning, and analytics.

Chuang Hu received the B.Sc. and M.Sc. degrees from Wuhan University in 2013 in 2016, respectively, and the Ph.D. degree from The Hong Kong Polytechnic University in 2019. He is currently a Research Assistant Professor with the Department of Computing, The Hong Kong Polytechnic University. His research interests include edge learning, computing networking, and recently federated analytics.

Dan Wang (Senior Member, IEEE) received the B.Sc. degree from Peking University, the M.Sc. degree from Case Western Reserve University, and the Ph.D. degree from Simon Fraser University, all in computer science. His research has been adopted by industry, e.g., Henderson, Huawei, and IBM. His research falls in general computer networking and systems, where he has published in ACM SIGCOMM, ACM SIGMETRICS, and IEEE INFOCOM, and many others. His recent research interest includes smart energy systems. He is a Steering Committee Member of ACM e-Energy. He won the Best Paper Awards of ACM e-Energy 2018 and ACM Buildsys 2018. He won the Global Innovation Award, TechConnect, in 2017. He served as the TPC Co-Chair for IEEE/ACM IWQoS 2020. He has served as the TPC Co-Chair for the ACM e-Energy 2020. He is the Steering Committee Chair of IEEE/ACM IWQoS. He will serve as the General Co-Chair for the ACM e-Energy 2022. He serves as a Founding Area Editor for *ACM SIGEnergy Energy Informatics Review*.



Yifei Zhu (Member, IEEE) received the B.E. degree from Xi'an Jiaotong University, Xian, China, in 2012, the M.Phil. degree from The Hong Kong University of Science and Technology, Hong Kong, China, in 2015, and the Ph.D. degree in computer science from Simon Fraser University, BC, Canada, in 2020. He is currently an Assistant Professor at Shanghai Jiao Tong University. His research interests include edge computing, multimedia networking, distributed machine learning systems, and network economics.

Zhu Han (Fellow, IEEE) received the B.S. degree in electronic engineering from Tsinghua University in 1997 and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Maryland, College Park, MD, USA, in 1999 and 2003, respectively.

From 2000 to 2002, he was a Research and Development Engineer at JDSU, Germantown, MD, USA. From 2003 to 2006, he was a Research Associate at the University of Maryland. From 2006 to 2008, he was an Assistant Professor at Boise State University, ID, USA. He is currently a John and Rebecca Moores Professor with the Electrical and Computer Engineering Department as well as the Computer Science Department, University of Houston, TX, USA. His research interests include wireless resource allocation and management, wireless communications and networking, game theory, big data analysis, security, and smart grid.

Dr. Han was an IEEE Communications Society Distinguished Lecturer from 2015 to 2018. He has been an AAAS Fellow since 2019 and an ACM Distinguished Member since 2019. He received the NSF Career Award in 2010, Fred W. Ellersick Prize of the IEEE Communication Society in 2011, the Best Paper Award from the *EURASIP Journal on Advances in Signal Processing* in 2015, the IEEE Leonard G. Abraham Prize in the field of Communications Systems (best paper award in IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS) in 2016, and several best paper awards in IEEE conferences. He is also the Winner of 2021 IEEE Kiyo Tomiyasu Award for outstanding early to mid-career contributions to technologies holding the promise of innovative applications, with the following citation: "For contributions to game theory and distributed management of autonomous communication networks." He has been 1% Highly Cited Researcher since 2017 according to Web of Science.