Privacy Preservation for Federated Learning with Robust Aggregation in Edge Computing

Wentao Liu, Xiaolong Xu*, *Member, IEEE,* Dejuan Li, Lianyong Qi, Fei Dai, Wanchun Dou, and Qiang Ni, *Senior Member, IEEE*

***Abstract*—Benefiting from the powerful data analysis and prediction capabilities of artificial intelligence (AI), the data on the edge is often transferred to the cloud center for centralized training to obtain an accurate model. To resist the risk of privacy leakage due to frequent data transmission between the edge and the cloud, federated learning (FL) is engaged in the edge paradigm, uploading the model updated on the edge server (ES) to the central server for aggregation, instead of transferring data directly. However, the adversarial ES can infer the update of other ESs from the aggregated model and the update may still expose some characteristics of data of other ESs. Besides, there is a certain probability that the entire aggregation is disrupted by the adversarial ESs through uploading a malicious update. In this paper, a privacy-preserving FL scheme with robust aggregation in edge computing is proposed, named FL-RAEC. First, the hybrid privacy-preserving mechanism is constructed to preserve the integrity and privacy of the data uploaded by the ESs. For the robust model aggregation, a phased aggregation strategy is proposed. Specifically, anomaly detection based on autoencoder is performed while some ESs are selected for anonymous trust verification at the beginning. In the next stage, via multiple rounds of random verification, the trust score of each ES is assessed to identify the malicious participants. Eventually, FL-RAEC is evaluated in detail, depicting that FL-RAEC has strong robustness and high accuracy under different attacks.**

***Index Terms*—Federated learning, Privacy preservation, Security, Edge computing**

I. Introduction

The continuous evolution of the Internet of Things (IoT) brings about an explosive data increase over mobile networks, which causes the data generation rate to far exceed the network capacity, making it unwise to concentrate all data on the cloud platform for processing [1], [2]. Therefore, most data need to be analyzed and processed locally to relieve the pressure of network transmission and the burden on the cloud data center. The proposal of edge computing provides a practical and flexible solution to local data processing. Compared with the cloud platform, uploading data to edge servers (ESs) drastically reduces the transmission delay and alleviates the network load owing to its proximity to end-users. According to the latest Cisco global cloud index, 850ZB data are generated at the network edge while only 20.6ZB data center traffic occurs in 2021, which indicates that the majority of data are processed in the ESs [2].

For the analysis of a large amount of data and the acquisition of valuable information, artificial intelligence (AI) is applied in edge computing due to its extraordinary adeptness in predicting future events [3]. As the most representative AI technology, machine learning (ML) is adopted in multitudinous fields (e.g. object detection, speech recognition, and natural language processing) [4]. A key enabler in ML is to train prediction models with massive data. With the assistance of abundant data, a well-trained model conducts prediction operations based on the input data, which has a high matching rate with the real situation [5], [6]. Additionally, the accuracy of the ML model is highly related to the size of datasets and the granularity of each record. Diverse data significantly improve the prediction effect of the model. Obviously, introducing ML to edge computing alleviates large data traffic in the ES and enhances the quality of services (QoS) of mobile network operators.

The resource capability of the ES, however, is limited and fails to meet the needs of complex AI models [7]. Mobile crowdsourcing is adopted for model training under the cooperation of the ESs. In mobile crowdsourcing, the resources and datasets of multiple ESs, as well as the model trained in each ES, are shared for training a global model synergistically. As the collaborative training requires data sharing, security and privacy issues exist in the procedure of ML in edge computing [8]. According to general data protection regulation (GDPR), data sharing between entities is forbidden without user permission, lessening the feasibility of data aggregation practices [9], [10]. Consequently, it remains challenging to realize the synergistic training of ML models in edge computing and ensure the security of private user data.

Federated learning (FL) is efficient to settle the privacy leakage problem for distributed datasets in edge computing

Wentao Liu is with the School of Computer Science, Nanjing University of Information Science and Technology, Nanjing, China
Email: liuwentao728@gmail.com

Xiaolong Xu (Corresponding Author) is with the School of Software, Nanjing University of Information Science and Technology, Nanjing, China, Jiangsu Collaborative Innovation Center of Atmospheric Environment and Equipment Technology (CICAEET), Nanjing University of Information Science and Technology, Nanjing, China, and State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
Email: xlxu@ieee.org

Dejuan Li is with Shandong Provincial University Laboratory for Protected Horticulture, Weifang University of Science and Technology, Weifang, China
E-mail: lidejuan2016@wfust.edu.cn

Lianyong Qi is with the School of Computer Science, Qufu Normal University, Qufu, China.
E-mail: lianyongqi@gmail.com

Fei Dai is with the College of Big Data and Intelligent Engineering, Southwest Forestry University, Kunming, China.
E-mail: daifei@swfu.edu.cn

Wanchun Dou is with State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China.
E-mail: douwc@nju.edu.cn

Qiang Ni is with the School of Computing and Communications, Lancaster University, Lancaster LA1 4WA, U.K.
E-mail: q.ni@lancaster.ac.uk

[11]. Compared with conventional ML methods, when the federated system is established, the data of each ES are preserved locally instead of being transmitted to a centralized node [12]. With FL, each ES exploits its own data to train a model, which is optimized only based on the dataset of the creator ES. For the sake of aggregating training results, the federated system establishes a virtual global model through parameter exchange under the encryption mechanism [13]. The virtual global model is distributed to the ESs for local model modification, representing the completion of synergistic training of AI models. Moreover, the inference effect of the aggregated global model is nearly equivalent to the trained model with conventional centralized ML methods. Benefiting from the decentralized feature, the security and privacy preservation for distributed datasets is improved while maintaining the inference accuracy for AI applications.

Although FL employs local data processing, recent researches reveal that security and privacy problems still take place in FL, which are mainly occurred on the aggregator-side and have close relationships to the model aggregation step [14], [15]. When thousands of ESs participate in the process of FL, malicious ESs can easily sneak into the training without prior verification. Provided that the malicious ESs intentionally change the model update(such as gradients) to be uploaded, the inference performance of the aggregated global model may sharply decline [16]. Currently, most of the existing security solutions focus on the aggregator-side anomaly detection which sometimes is bypassed by malicious ESs with confrontational means [17]. In addition, resource-constrained ESs are vulnerable to attacks. If the ESs participating in FL are invaded, they will deliberately steal model updates uploaded by other ESs, causing privacy leakage in FL. Finally, the compromised ESs restore the model from the stolen update to infer the characteristics of the local datasets [18].

To contrapose the security and privacy issues of applying FL in edge computing, we propose a privacy-preserving FL scheme with robust aggregation in edge computing, named FL-RAEC. In the upload stage, a hybrid privacy-preserving mechanism is constructed to guarantee the integrity and privacy of data during data uploading. In the model aggregation stage, a phased aggregation strategy with anomaly detection and anonymous trust verification is proposed, where the anomaly score of model weight is estimated by the result of autoencoder-based anomaly detection before each aggregation. During the training process, some parts of the ESs execute trust verification at the same time. After multiple rounds of anonymous trust verification, the malicious ESs will be identified through a trust score. Finally, we evaluate the performance of FL-RAEC in the simulation experiment.

The main contributions of this paper are as follows.

- An improved FL scheme is proposed to guarantee privacy and secure distributed training in edge computing which consists of a hybrid privacy-preserving mechanism and a phased aggregation strategy.
- The hybrid privacy-preserving mechanism is constructed with asymmetric encryption and local differential privacy (LDP) for data uploading. The mechanism avoids parameter tampering and privacy leakage of model updates during the data uploading, preserving the integrity and privacy of uploaded data.
- In the phased aggregation strategy, the autoencoder-based anomaly detection is responsible for reducing the impact of the malicious model update, while the trust estimation of each ES is launched to identify malicious ESs after multiple rounds of anonymous random verification.
- The performance of FL-RAEC is evaluated via various experiments, which confirms that FL-RAEC guarantees robust aggregation under different attacks while protecting the privacy of ESs.

The framework of this paper is arranged as follows. Section II poses an overview of the research status of privacy preservation and security in FL. Section III illustrates the system framework and problem formulation. Section IV describes a hybrid privacy-preserving mechanism with LDP. Section V proposes a phased model aggregation strategy with anomaly detection and trust verification. Section VI presents the experiment results and evaluation. Finally, Section VII discusses the conclusion and future work of this paper.

II. Related Work

Since ESs are with limited hardware resources and only collect small datasets, model training on a single ES is not efficient. Therefore, an ES usually needs to upload the collected data to cloud centers for centralized or collaborative training with other ESs. In this process, data exchange and transmission are required, which leads to privacy leakage of users and threats to data security. In such a case, federated learning (FL) emerges as a new machine learning paradigm. In FL, different machines work together to establish a virtual model with the data not uploaded, thereby user's privacy is well protected. Additionally, FL is tightly bounded with edge computing, which significantly reduces the computation load during the AI training process.

A. Federated Learning

As the increasing redundant data is produced by privacy protection in FL, the issue of resource shortage in the central server cannot be solved by improving algorithms alone. To break through the bottleneck, the edge-computing-assisted FL model has been widely studied in fields including distributed learning, computation resource allocation and wireless channel partitioning. For distributed learning, Cui et al. [19] developed a compressed algorithm of FL for content caching called CREAT, in which the edge nodes used the local data for model training. CREAT effectively dispersed the large datasets, thereby relaxing the computing burden of the central server. Luo et al. [20] focused on computation resource allocation and the edge device association in FL. By optimizing the convex resource allocation sub-problem, an edge association strategy was achieved through iterative global cost. Yu et al. [21] designed a two-timescale deep reinforcement learning method composed of a fast-timescale and a low-timescale process to allocate the computing resources effectively for the FL on edge servers. To improve the utilization of wireless channels, Amiri et al. [22] leveraged a digital stochastic gradient descent

scheme to divide the shared channels of the distributed edge servers for FL. According to the errors generated by the historical iteration, the gradient descent function is established to allocate the channel reasonably. Zhu et al. [23] combined federated edge learning with broadband analog aggregation technology to reduce communication latency during the data offloading process in FL.

B. Privacy and Security Issues in FL

As one of the most dominant attributes of FL, privacy protection has been deeply investigated in recent years. The most basic privacy protection is to upload the parameters after the operation and keep the original data locally. Wang et al. [24] adopted an algorithm of data augmentation based on semi-supervised learning and label guessing where the edge device sent the processed data to the cloud for further processing. To avoid the disclosure of the users' personal information through the analysis of client-uploaded parameters, Wei et al. [25] proposed a noising-before-model-aggregation FL scheme where artificial noise is added to the parameters at the clients' side. The privacy levels and convergence performance were jointly optimized based on the trade-off levels. In [26], Hao et al. improved the data noise technology by proposing a privacy-enhanced FL based on differential privacy (DP). By leveraging the distributed Gaussian mechanism, they achieved an example-level DP. Nevertheless, the process of encryption and decryption in privacy protection algorithms tends to produce additional datasets. Consequently, the increased computing burdens lead to inefficiency in FL. To tackle this issue, Xu et al. [27] proposed a privacy-preserving federated deep learning framework (PPFDL) with irregular users. With the high integration of additive homomorphism and Yao's garbled circuits, the quality of data sources was improved on the premise of user privacy, thus enhancing the efficiency of FL.

Guaranteeing the training process by avoiding poisoning attacks is another important goal of FL, and thus has received wide attention. Depending on the timing of the poisoning attacks, poisoning attacks can mainly be divided into two categories. One is data poisoning attack exists in local data collection. And another is the model poisoning attack, which occurred in local model training. In [28], Khazbak et al. established a system with poisoning attack mitigation called MLguard for distributed FL, which prevented malicious users from poisoning training data. To reduce the risk of model poisoning attacks, Short et al. [29] proposed a blockchain-based defense scheme, where each model update was verified separately without the information of the training data size. Singh et al. [30] proposed an approach based on micro aggregation which clusters the participants with similar attribute values and then trains the federated learning model for each cluster, attempting to achieve a balance between the fighting model poisoning and the accommodating diversity. Additionally, to improve the data security without sacrificing the efficiency of FL, Qu et al. [31] proposed a novel blockchain-enabled FL scheme which guaranteed both the safety and the efficiency of fog computing which suffered from poisoning attacks. Specifically, in the scheme, FL only required the swap of training updates, thus achieving high efficiency.

Overall, most of the research on privacy and security is designed for the entire FL framework, but they did not consider the communication and computation efficiency for their implementation in edge computing. To reduce the overhead of privacy and security mechanisms for those resource-constrained devices during FL, we designed a robust aggregation method for FL in edge computing. Thereinto, most additional overhead is undertaken by ESs, improving the extensibility of the method to more devices with low computing power.

TABLE I
MAIN NOTATIONS

Notation	Description
$\mathcal{E}$	The set of ESs, where $\mathcal{E} = \{ES_1, ES_2, ..., ES_P\}$
$\mathcal{M}$	The set of corresponding models, where $\mathcal{M} = \{M_1, M_2, ..., M_P\}$
P	The number of ESs
K	The number of ESs participating in the aggregation
D_i	The dataset owned by E_i
X_i	The inputs in dataset D_i
Y_i	The labels in dataset D_i
N_i	The size of dataset D_i
α	The learning rate of each ES
t	The aggregation round
T	The maximum aggregation round
$L(t)$	The loss function of the aggregated model at aggregation round t
$w(t)$	The weight of aggregated model at aggregation round t
ϵ_i	The privacy budget of ES_i
ϵ^r	The maximum privacy budget of a single round
ϵ^{total}	The maximum privacy budget

III. SYSTEM MODEL AND PROBLEM FORMULATION

In this section, firstly a communication framework with FL is given out. Then, the local training model, the threat model, and the aggregation model are described, respectively. Finally, the problem is formulated. The main symbolic representations of this paper are summarized in Table I.

A. Distributed Training with FL in Edge Computing

Owing to the powerful feature representation and self-learning ability, deep learning has been adopted in many applications of edge computing, such as computation offloading and content caching. Previously, the data of end devices (EDs) collected by edge services (ESs) need uploading to the cloud for model training, which poses a threat of privacy leakage during the data transmission. Different from conventional centralized learning on the cloud, in FL, the model is trained on the ESs locally and ESs only need to upload the model update to the cloud for aggregation.

As depicted in Fig.1, the communication framework consists of the cloud server (CS), several ESs and EDs. Data are generated by the EDs and collected by the ESs for local model training. At the end of each training phase on the edge, the CS will select parts of ESs for aggregation and the selected ESs upload their weight update(model update) to the CS for model aggregation and global model optimization. After that, the new

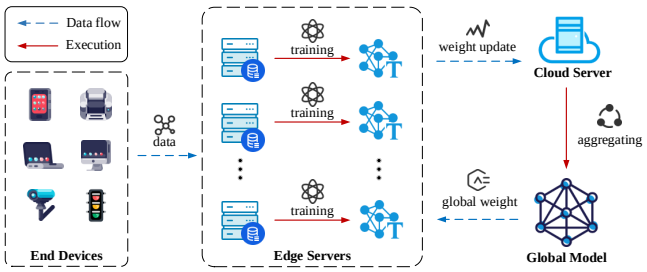


Fig. 1. A communication framework with FL in edge computing

global model will be sent back to all ESs, which continue the next training phase until the accuracy of the local model can not improve further. Eventually, with the trained local model, each ES can satisfy the service requirements of the EDs by model inference. In the whole training process, the data of the EDs are only accessed by the corresponding ES, preventing the cloud or other ESs from approaching the privacy data. Meanwhile, since the datasets collected by multiple ESs are utilized in training, the performance of the local models will be better than that trained by the dataset of a single ES.

B. Local Training Model

The essential purpose of the ESs is to improve the model accuracy through iterative training on the local dataset. Denote the set of ESs as $\mathcal{E} = \{ES_1, ES_2, ..., ES_P\}$ and the set of corresponding models as $\mathcal{M} = \{M_1, M_2, ..., M_P\}$, where P represents the number of ESs. Each ES_i has its own dataset D_i, with the the inputs and the labels denoted as X_i and Y_i, respectively. Mean squared error (MSE) is utilized as the loss function of M_i on D_i, which is defined as

$$L_i(D_i, w_i(t)) = \frac{1}{2N_i} \left\| M_i\left(X_i; w_i(t)\right) - Y_i \right\|^2, \tag{1}$$

where $w_i(t)$ is the weight of M_i at aggregation round t and N_i is the size of dataset D_i. Hence, the update of weight on D_i is calculated by

$$\Delta w_i^{D_i} = -\alpha \nabla L_i(D_i, w_i), \tag{2}$$

where α is the learning rate of each ES and $\alpha \nabla L_i(D_i, w_i)$ is the gradient of M_i. The aggregation round t is omitted for simplicity, similarly hereinafter.

C. Threat Model

Through local model training and weight update sharing, FL achieves the goal of training efficiency. However, privacy and security problems exist during communication and training. In this paper, we assume that the CS (i.e., the aggregator) is honest-but-curious, but the ESs (i.e., the participants) are untrusted, which means there may be malicious participants in the whole training process. In the following, the threat model of inference attacks and poisoning attacks is analyzed.

Although transmission of weight update avoids the risk of privacy leakage compared with raw data transmission, the weight update still reveals model characteristics, which brings inference attacks. Specifically, the additional features of the private dataset may be learned by the neural network, particularly for those large networks with many hidden layers. These features could be captured by adversaries if the communication is tapped during model aggregation. Besides, the adversary can infer the changing of the model by snapshotting the model update and observing their differences at continuous time periods.

Besides, as mentioned before, not all participants are definitely honest. Malicious participants can launch poisoning attacks during the aggregation, which may hinder or even fail the training process. One of the common ways of poisoning attacks is the Byzantine attack. The malicious participants deliberately upload random weight updates to reduce the overall aggregation performance of the model. Since model training consumes computing resources, if those honest participants can not get the corresponding incentive (e.g., the improvement of model accuracy), their enthusiasm for participating in the model aggregation will be greatly reduced. Therefore, the total training process may fail due to the attack.

Considering the multiple threats to the training process, an efficient privacy-preserving and secure scheme are necessary for FL in edge computing. LDP is leveraged in this paper to overcome the threats. To prevent inference attack, a noise is added to the weight update to satisfy the requirement of LDP before the uploading as

$$\Delta \tilde{w}_i^{D_i} = \Delta w_i^{D_i} + r_i, \tag{3}$$

For privacy preservation, the uploaded date needs to meet certain constraints. D_i and D_i' are two datasets with hamming distance of 1. The max-divergence of $\Delta \tilde{w}_i^{D_i}$ and $\Delta \tilde{w}_i^{D_i'}$ is defined as

$$\Omega_\infty(\Delta \tilde{w}_i^{D_i} \| \Delta \tilde{w}_i^{D_i'}) = \max_{w \in \Delta \tilde{w}_i^{D_i}} \ln \frac{\Pr[\Delta \tilde{w}_i^{D_i} = w]}{\Pr[\Delta \tilde{w}_i^{D_i'} = w]}. \tag{4}$$

When the divergence is closer to 0, it is more difficult for the adversary to distinguish from which dataset the weight update comes. A privacy budget ϵ_i is introduced so that $\Omega_\infty(\nabla \tilde{w}_i^{D_i} \| \nabla \tilde{w}_i^{D_i'})$ is limited within

$$\Omega_\infty(\Delta \tilde{w}_i^{D_i} \| \Delta \tilde{w}_i^{D_i'}) \leq \epsilon_i, \tag{5}$$

which can be simplified as

$$\Pr[\nabla \tilde{w}_i^{D_i} \in \mathbb{S}] \leq e^{\epsilon_i} \Pr[\nabla \tilde{w}_i^{D_i'} \in \mathbb{S}], \tag{6}$$

where $\mathbb{S}$ is the set of output [32].

D. Aggregation Model

When the CS receives the weight update with noise from the selected ESs, the global model update can be calculated by model aggregation as

$$\Delta \tilde{w} = \sum_{i=1}^{K} q_i \Delta \tilde{w}_i^{D_i}, \tag{7}$$

where K is the number of ESs which participate in the model aggregation. Since there may be malicious participants and the dataset size of ESs is different, the update of each model

M_i is weighted by q_i as the aggregation weight. Compared to aggregating updates on average, the aggregator can reduce the impact of those malicious participants on training effectiveness by reducing their aggregation weights. The new global model weight of this iteration can be calculated by

$$w(t+1) = w(t) + \Delta\tilde{w}(t). \tag{8}$$

The metric of aggregation quality in this iteration is the size of the overall loss function, which is calculated by

$$L(t+1) = \sum_{i=1}^{P} \frac{N_i}{N} L_i(D_i; w(t+1)), \tag{9}$$

where N is the size of all datasets.

E. Problem Formulation

The final goal of our privacy-preserving FL scheme is to maximize the model accuracy while protecting privacy. Hence, the optimization problem can be formulated as

$$\min_{w(t),q(t)} L(t) \quad \forall t \leq T \tag{10}$$

$$\begin{aligned} \text{s.t.} \quad & \Pr[\Delta\tilde{w}_i^{D_i} \in \mathbb{S}] \leq e^{\epsilon_i} \Pr[\Delta\tilde{w}_i^{D_i'} \in \mathbb{S}] \quad \forall i \leq K, \\ & \epsilon_i \leq \epsilon^r \quad \forall i \leq K, \\ & Composition(\epsilon^r, T) \leq \epsilon^{total}, \end{aligned} \tag{11}$$

where T is the maximum aggregation round, and ϵ^r is the maximum privacy budget of single round. $Composition(\epsilon^r, T)$ represents the total privacy budget through T rounds composition and ϵ^{total} stands for the maximum privacy budget after composition.

IV. A Hybrid Privacy-preserving Mechanism for Data Uploading in FL

In the proposed scheme, to guarantee honesty and to improve the efficiency of model aggregation, the ESs need to upload other parameters (e.g., dataset size, model loss, and verification result) for the formulation of the aggregation weights and the trust verification. If these parameters are tampered with during transmission, it will lead to an incredibly negative impact on the model aggregation process. Therefore, a hybrid privacy-preserving mechanism is proposed to guarantee the privacy of weight updates and other parameters uploading.

As depicted in Fig.2, two different privacy-preserving strategies are utilized for parameter and weight update uploading. For parameter uploading, asymmetric encryption is employed to avoid the uploaded parameter being tapped or modified. First, the key generator of the CS selects a specific asymmetric encryption algorithm (e.g., RSA, ECC, etc.) to generate public and private keys. Then the public key is sent to the ES, which encrypts the parameters with the public key and uploads the parameters to the CS. Afterward, the CS deciphers the encrypted parameters with the private key. Additionally, for the uploading of weight updates, LDP is employed to disturb the update before uploading it to CS for aggregation.

LDP is a novel privacy preservation technology based on traditional centralized differential privacy (CDP) [33]. In conventional DP [32], the data need uploading to the data center for privacy processing, and then the statistics that satisfy differential privacy will be published to the data processing unit. During data processing, a trusted third party is indispensable. Compared to CDP, LDP transfers the data privacy processing to each user, thereby guaranteeing the data privacy during the phase of data collection and avoiding the privacy leakage caused by the untrusted third party. In FL-RAEC, LDP is employed for the uploading of weight updates during the whole training process.

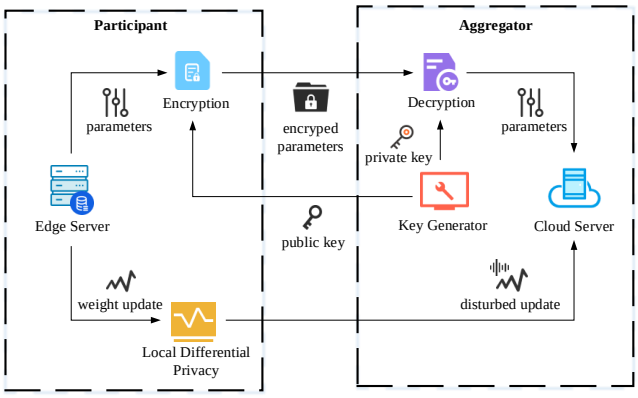


Fig. 2. A hybrid privacy-preserving mechanism for data uploading

Considering that it is difficult to meet the strict definition of ϵ-DP in the actual application scenario, a relaxed (ϵ, δ)-DP is applied to the weight uploading process, which is defined as

$$\Pr[\Delta\tilde{w}_i^{D_i} \in \mathbb{S}] \leq e^{\epsilon_i} \Pr[\Delta\tilde{w}_i^{D_i'} \in \mathbb{S}] + \delta, \tag{12}$$

where δ is a relaxation term which allows a violation of strict ϵ-DP with probability δ. In order to make the weights update uploaded by the ESs meet the above DP definition, the sensitivity of the weight update needs to be calculated, which determines the distribution of noise added to uploaded weight. The sensitivity of $\Delta w_i^{D_i}$ is calculated as

$$S(\Delta w_i^{D_i}) = \max_{D_i, D_i'} \|\Delta w_i^{D_i} - \Delta w_i^{D_i'}\|. \tag{13}$$

To restrict the range of sensitivity $S(\Delta w_i^{D_i})$, the weight update needs to be clipped as

$$\Delta w_i^{D_i} \leftarrow \frac{\Delta w_i^{D_i}}{\max\left\{1, \left\|\Delta w_i^{D_i}\right\| C^{-1}\right\}}, \tag{14}$$

where C is the hyperparameter which decides the clipping extent, making $\|\Delta w_i^{D_i}\| \leq C$. Through the weight clipping, the sensitivity of the weight update is limited within

$$S(\Delta w_i^{D_i}) = \max_{D_i, D_i'} \|\Delta w_i^{D_i} - \Delta w_i^{D_i'}\| \leq 2C \tag{15}$$

After calculating the sensitivity of $\Delta w_i^{D_i}$, the ES adds the corresponding noise according to the single-round privacy budget and the data sensitivity to the weight before uploading. The Gaussian mechanism [34] is applied to the added noise and the uploaded weight, which is defined as

$$\Delta\tilde{w}_i^{D_i} = \Delta w_i^{D_i} + \mathcal{N}(0, \sigma_i^2), \tag{16}$$

where the noise satisfies the Gaussian distribution with an expectation of 0 and a variance of σ_i^2. The standard deviation σ_i conforms to

$$\begin{aligned}\sigma_i &> S(\Delta w_i^{D_i})\frac{\sqrt{2\ln(1.25/\delta_i)}}{\epsilon_i} \\ &\geq 2C\frac{\sqrt{2\ln(1.25/\delta_i)}}{\epsilon_i}\end{aligned}, \tag{17}$$

where ϵ_i is the single-round privacy budget of ES_i and $\delta_i \in (0,1)$ is the relaxation term.

Since training data from the same batch of the dataset is selected in the local training process, the uploaded weight updates of the same ES at different rounds are not independent. The privacy budget will accumulate after multiple rounds of uploading. With the composability of differential privacy, the overall privacy budget can be calculated by various composition theorems. To minimize privacy overhead while ensuring the $(\epsilon^{total}, \delta^{total})$-DP, we utilized the strong composition theorem [35] as the privacy accountant, where the total privacy budget is calculated as

$$\epsilon^{total} = \epsilon^r(\sqrt{2Tln(1/\delta')} + T(e^{\epsilon^r} - 1)), \tag{18}$$

where δ' is a new customized relaxation term. According to the parallel composition theorem, $\epsilon^r = \max \epsilon_i$ is the privacy budget of each round. The total relaxation term is calculated as

$$\delta^{total} = T\delta^r + \delta'. \tag{19}$$

V. Model Aggregation with Anomaly Detection and Anonymous Trust Verification

Although the hybrid privacy-preserving mechanism secures the integrity and privacy of data during uploading, the malicious ESs can still upload a modified update to affect the global model, thereby indirectly inferring data characteristics of other ESs from the received change of the global model. Moreover, the aggregation may be disrupted due to the introduction of anomaly updates, causing the entire learning process to fail to reach the predetermined goal. In this section, a phased aggregation strategy is proposed, which consists of anomaly detection and anonymous trust verification. The overall process of the phased aggregation strategy is described at the end.

A. Autoencoder-Based Anomaly Detection for Uploaded Weight Update

Autoencoder is a neural network which is capable to learn the main characteristics of high-dimensional data in low-dimensional features through unsupervised learning. An autoencoder includes two components: an encoder and a decoder. The extraction process of the efficient representation of the input data is called coding, whose outputs' dimension is generally much smaller than that of the input data. Hence, the autoencoder can be used for data dimensionality reduction. The target of the decoder is to reconstruct the input data with the learned effective representation. This process is called decoding. Due to the robust feature expression and data restoration ability of the autoencoder, it is adopted in various applications, including anomaly detection [36].

For anomaly detection in model aggregation in FL, the input of the autoencoder is the uploaded weight update. However, due to the added noise and the uncertainty of weight changing, it is difficult for the autoencoder to learn the efficient representation of the weight update. Therefore, instead of encoding the weight update, we treat the model weights as the input of the encoder. As the model aggregation is given by (7) and (8), which are equivalent to

$$w(t+1) = \sum_{i=1}^{K} q_i(t)\tilde{w}^{D_i}(t), \tag{20}$$

where $\tilde{w}^{D_i}(t)$ can be restored by the aggregator as

$$\tilde{w}^{D_i}(t) = w(t) + \Delta\tilde{w}^{D_i}(t). \tag{21}$$

Denote the training dataset of the autoencoder as $\mathcal{TD} = \{tw_1, tw_2, ..., tw_i\}$, and define the mapping relations of the encoder and the decoder as

$$f : \Psi \to \Lambda, \quad g : \Lambda \to \Psi, \tag{22}$$

where Ψ is the input space, i.e., $\mathcal{TD} \in \Psi$, and Λ is the feature space. The goal of the encoder is to extract the low-dimensional features of the model weights, and the decoder's goal is to reconstruct the input weights from the low-dimensional features. Therefore, the final goal of the autoencoder is presented as

$$f, g = \arg\min_{f,g} \frac{1}{|\mathcal{TD}|}\sum_{i=1}^{|\mathcal{TD}|}\phi(tw_i), \tag{23}$$

where error $\phi(tw_i)$ is calculated as

$$\phi(tw_i) = \|tw_i - g(f(tw_i))\|, \tag{24}$$

where $f(tw_i)$ represents the low-dimensional features of the input weights and $g(f(tw_i))$ represents the reconstructed weights from the features. Through continues gradient descent, the low-dimensional feature distribution of the input weights can be finally learned by the autoencoder.

When the CS receives the distributed weights uploaded by the ESs participating in the aggregation, the pre-trained autoencoder will detect the uploaded weights anomaly to infer the anomaly score. For complex deep neural networks with multiple layers, the dimensionality of the uploaded weights may be huge, which greatly improves the computational complexity of the autoencoder. In view of the commonality, we select the weights after the last hidden layer as the input to perform dimensionality reduction. For each uploaded weight, $\phi(\tilde{w}_i)$ measures the degree of deviation between it and the output of the autoencoder. The anomaly score of $\tilde{w}_i$ is calculated as

$$A_i = e^{\beta(1-a_i)}, \tag{25}$$

where β is the anomaly impact factor which determines the impact of the anomaly score on the whole aggregation. And a_i is calculated as

$$a_i = \begin{cases} 1 & \phi(\tilde{w}_i) \leq \mu^\phi + 3\sigma^\phi \\ \frac{\phi(\tilde{w}_i)}{\min\limits_{j\leq K}\phi(\tilde{w}_j)} & \phi(\tilde{w}_i) > \mu^\phi + 3\sigma^\phi \end{cases}, \tag{26}$$

where μ^{ϕ} is the mean of the top 50% minimum errors of the uploaded weights, and σ^{ϕ} is calculated as

$$\sigma^{\phi} = \mu^{\phi} - \min_{j \leq K} \phi(\tilde{w}_i). \tag{27}$$

The threshold is based on the assumption that the proportion of the attackers generally does not exceed 50% of the total participants, and $3\sigma^{\phi}$ tolerates the errors caused by the added noise. The calculated anomaly score and the trust score introduced in the next subsection determine the aggregation weight together.

B. Anonymous Trust Verification of ESs During Training

Autoencoder-based anomaly detection is usually effective against random poisoning attacks which aim at reducing the accuracy of the global model. Since this type of attack is designed to disrupt the aggregation process, its model weights distribution is very unconventional from other regular participants, which can be easily captured by the autoencoder with larger $\phi(w)$. However, for some model poisoning attacks (such as sign-flip attacks [37]), the weight distributions of the poisoned models are similar to the common models because the features of the training data are unchanged. Moreover, alternating minimization strategies such as parameter estimation are utilized to evade detection, making the anomalous weights difficult to be identified by the autoencoder.

Meanwhile, as an essential part of aggregation weight, the loss on the local validation dataset also needs to be uploaded to the aggregator. The malicious ESs tend to upload the unfaithful loss to improve their influence in the aggregation phase. Considering the above shortcomings, the reliable verification of ESs is necessary to ensure the correctness of uploaded loss. However, the aggregator does not hold the dataset, which means the verification is executed by participants. Therefore, the privacy of verified models needs to be considered during verification as well.

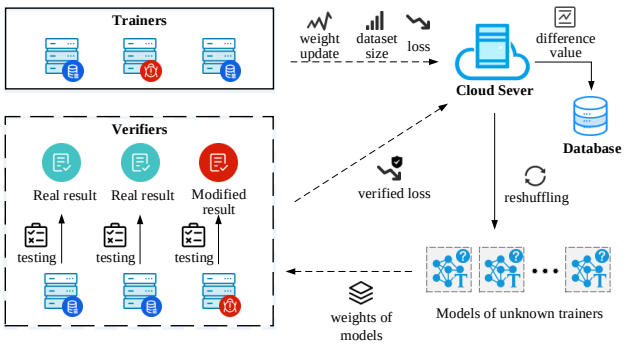


Fig. 3. The anonymous trust verification

As depicted in Fig.3, the CS randomly selects part of the ESs as the trainers to perform local training at each round. After local training, the selected trainers upload their weight update, sizes of the dataset, and losses on the validation dataset for aggregation. In the figure, the red ESs represent the malicious participants who deliberately upload lower losses to improve the aggregation weights of their models. Against this dishonest behavior, the CS randomly selects other parts of the ESs as the verifiers for trust verification. Technically, when the CS receives the update from the trainers, it restores the model weights of each trainer besides aggregating them. Then, the CS will reshuffle these models and distribute them to the selected verifiers in the next round. When trainers are training their local models, the verifiers are testing the model of the last round with their local dataset at the same time, but they do not know from which ES the model comes. Owing to the model reshuffling and random verification, it is almost impossible for a single ES to track the model changes of other particular ES through verification. After anonymous verification, the verifiers will upload the verified loss on their dataset. To interfere with the verification process, the malicious ES usually uploads the enlarged loss for those models with lower loss, which brings the instability of the trust score at an early stage. After the CS collects all verified losses, it will calculate the difference value according to the verification of this round and record the result of each verified ES. With continuous random verification, the CS can eventually calculate a stable trust score.

At the u-th round of the verification for ES_i, the difference value is calculated as

$$diff_i(u) = \frac{\sum_{j=1}^{G_i(u)} V_{i,j}(u) \sum_{i=1}^{K} G_i(u)}{\sum_{i=1}^{K} \sum_{j=1}^{G_i(u)} V_{i,j}(u)}, \tag{28}$$

where $G_i(u)$ is the number of verifiers participating in the verification of M_i at verification round u. The verification error $V_{i,j}(u)$ is calculated as

$$V_{i,j}(u) = |loss_{i,j}^{ve} - loss_i^{tr}|, \tag{29}$$

where $loss_{i,j}^{ve}$ is the verified loss of ES_j which verifies M_i, and $loss_i^{tr}$ is the loss uploaded by trainer ES_i at the last round.

In (28), $\sum_{j=1}^{G_i(u)} V_{i,j}(u)$ is the overall verification error of ES_i. Since the restored weight is added with noise and the dataset of each ES is different, calculating the absolute value directly may lead to large differences in the trust scores of different ESs. Therefore, the average value of verification error $\frac{\sum_{i=1}^{K} \sum_{j=1}^{G_i(u)} V_{i,j}(u)}{\sum_{i=1}^{K} G_i(u)}$ is used to normalize it. For the honest participant ES_i, its difference value $diff_i$ will be equal to or less than 1. In contrast, The malicious ESs' will be greater than 1.

After multiple rounds of verification, the mean different value of M_i is calculated by

$$diff_i^m = \frac{\sum_{u=1}^{U_i} diff_i(u)}{\sum_{u=1}^{U_i} G_i(u)}, \tag{30}$$

where U_i is the total rounds in which M_i is verified. According to $diff_i^m$, the trust score of M_i is calculated as

$$R_i = \begin{cases} 1 & diff_i^m \leq 1 \\ (\frac{1}{diff_i^m})^{\lambda} & diff_i^m > 1 \\ 1/2 & \sum_{u=1}^{U_i} G_i(u) = 0 \end{cases}, \tag{31}$$

where λ is calculated as

$$\lambda = \frac{\sum_{j=1}^{P} H_i}{\sum_{j=1}^{P} diff_j^m H_i}, \tag{32}$$

where H_i is a binary variable, determined by

$$H_i = \begin{cases} 1 & diff_i^m \leq 1 \\ 0 & diff_i^m > 1 \end{cases}. \tag{33}$$

When the ES has not been verified once, it has no mean difference value. In this case, the $diff_i^m$ will be equal to $1/2$ as compensation. In (31) and (32), λ measures the deviation between 1 and the $diff_i^m$ of those honest ESs. When ESs are all honest participants, $diff_i^m$ will be distributed around 1. In the presence of malicious ESs, the $diff_i^m$ of honest ESs and malicious ESs will show a trend of polarization, which leads to a greater λ and lower trust score R_i for those malicious ESs.

Algorithm 1: random trust verification

Input: The restored weights of trainers $\tilde{w}_i$, and the corresponding loss $loss_i^{tr}$
Output: The mean difference value of trainers $diff_i^m$

1 **for** *each restored weight* $\tilde{w}_i$, $i \leq K$ **do**
2 Select random $G_i(u)$ ESs as the verifiers of model M_i;
3 **for** *each selected verifiers* ES_j, $j \leq G_i(u)$ **do**
4 Distribute the restored weight $\tilde{w}_i$ to ES_j;
5 Calculate and upload the verified loss $loss_{i,j}^{ve}$;
6 **end**
7 Calculate the overall verification error of ES_i $\sum_{j=1}^{G_i(u)} V_{i,j}(u)$;
8 **end**
9 **for** *each verified model* M_i, $i \leq K$ **do**
10 Calculate the difference value $diff_i(u)$ at this verification round according to (28);
11 Update the mean difference value $diff_i^m$ according to (30);
12 **end**
13 return $diff_i^m$;

The process of trust verification is presented in Algorithm 1. At the end of each training round, the aggregator selects parts of the ES as the verifiers for each trainer (Line 3). Each verifier tests the restored weight $\tilde{w}_i$ on the local dataset and uploads the calculated verified loss (Lines 5-6). After all verified losses of M_i are uploaded, the aggregator will compare the $loss_i^{tr}$ with $loss_{i,j}^{ve}$ and calculate the overall verification error as $\sum_{j=1}^{G_i(u)} V_{i,j}(u)$ (Line 7). According to the verification errors, the difference value at this round is calculated for each trainer (line 10). At the same time, the number of times which M_i has been verified is accumulated and the mean difference value is updated (line 11).

C. Score-based Model Aggregation Strategy in Phase

Considering the situation of aggregation and verification in different phases, a phased model aggregation strategy is designed. At the beginning phase of training, the global model has lower accuracy, and thus needs cursory multi-round model aggregations to improve accuracy fleetly. In this phase, the aggregation process is mainly guaranteed by the autoencoder-based anomaly detection. Those malicious updates which greatly affect the speed and the efficiency of aggregation will be detected and screened out. At the same time, due to the interference of malicious participants on verification, the previous trust score is in a relatively unstable state. Therefore, in the first phase, the total score of each participating ESs at aggregation round t is calculated as

$$score_i(t) = (1 - \frac{loss_i^t}{\sum_{i=1}^K loss_i^t})A_i, \quad t < T^{trust}, \tag{34}$$

where T^{trust} is the round that launch the trust score.

In the middle and later phases of training, with the progress of random trust verification, the trust score will stabilize. Those covert malicious ESs will show little credibility, which is reflected by a lower trust score. At this phase, the total score of each participating ES at aggregation round t is calculated as

$$score_i(t) = (1 - \frac{loss_i^t}{\sum_{i=1}^K loss_i^t})A_i R_i, \quad T^{trust} \leq t \leq T. \tag{35}$$

The aggregation weight of each ES is constituted by the total scores and the size of its dataset, which is calculated by

$$q_i(t) = \frac{N_i score_i(t)}{N \sum_{i=1}^K score_i(t)}. \tag{36}$$

Eventually, through the weight update from the ESs $\tilde{w}_i$ and the corresponding aggregation weight q_i at each aggregation round, the weights of a final global model $w(T)$ can be obtained.

The phased model aggregation is illustrated in Algorithm 2. At the beginning of each round, the aggregator randomly selects K ESs as trainers to train the local model(Line 4). After the training, the trainers upload the distributed weight updates and the losses on the local dataset (Lines 5-9). For the uploaded weight update, the aggregator will perform anomaly detection and calculate the anomaly score before aggregation (Lines 10-11). In the early aggregation rounds, the trust score of each ES is fluctuant, so the score is only determined by the anomaly score (Line 13). As the random trust verification continues, the training enters the second stage, and the trust score is adopted in the score evaluation (Lines 15-16). Based on the score, the aggregation weights are determined and the weight updates are aggregated with q_i (Lines 18-19). Then, the global model is updated and distributed to each ES (Lines 20-21). The new round of trust verification for these trainers begins. (Line 22). After T rounds of aggregation, the weight of the global model is obtained.

VI. Experiment Evaluation and Results

In this section, we evaluate the proposed scheme FL-RAEC with a standard MNIST dataset for handwritten digit recognition. The MNIST dataset is a widely used dataset for classification tasks, which is with 60000 train examples and 10000 test examples. Each example is a 28x28 size greyscale image and belongs to a certain set of numbers from 0-9 [38]. Our model is a fully connected network with two hidden layers. The numbers of hidden units in the two layers are both 200. The prediction accuracy can reach above 98% after 100 rounds of concentrated training.

Algorithm 2: Phased model aggregation

Input: Private dataset of ESs $\mathcal{D} = \{D_1, D_2, ...D_i\}$, Initial global model weight w_o
Output: Final model weight $w(T)$

Initialize global model weight $w(0) = w_o$;
Distribute initial model to all ESs;
for *each aggregation round* $t \leq T$ **do**
  Select random K ESs as trainers;
  for *each trainer* ES_i, $i \leq K$ **do**
    Perform local training and get weight update Δw_j;
    Add Gaussian noise according to (16);
    Upload the disturbed update $\Delta\tilde{w}_i$ and the loss on local dataset $loss_i^{tr}$
  end
  Perform anomaly detection on $\Delta\tilde{w}_i$;
  Calculate anomaly score A_i according to (25) and (26);
  if $t < T^{trust}$ **then**
    Calculate $score_i$ according to (34);
  else
    Calculate trust score R_i according to (31) and (32);
    Calculate $score_i$ according to (35);
  end
  Calculate aggregation weight q_i according to (36);
  Aggregate weight updates according to (7);
  Update global model weight $w(t+1)$ according to (8);
  Distribute global model weight $w(t+1)$ to all ESs;
  Execute random trust verification for these trainers.
end
return $w(T)$;

A. Experiment Environment and Settings

In the simulation experiment, we randomly distribute the dataset to 100 ESs. In view of the communication overhead of the ESs, we use FedAVG[39] which has fewer communication rounds through increasing local training rounds as our benchmark scheme instead of FedSGD. At each aggregation, the number of the participating ESs, i.e., K, is 10, and each ES trains 5 rounds before uploading weight update, with learning rate lr set to 0.01. The max aggregation round T is 100 and the round launching trust score T^{trust} is set to 40. Experiments are implemented in python 3.6 and conducted on the server with Intel i7-8700 CPU (3.2 GHz, 6 cores), 32 GB DRAM and NVIDIA RTX 3090 GPU.

B. Impart on Privacy Budget

The employment of LDP protects the privacy of the local dataset of the ESs to a certain extent. The degree of protection depends on the overall privacy budget allocated. As Fig.4 shows, the uploading with no privacy has the best accuracy, and the accuracy of the global aggregation model gradually decreases as the standard deviation of added Gaussian noise increases. Three standard deviations corresponding to the privacy budgets $\epsilon = 1$, $\epsilon = 0.5$, $\epsilon = 0.25$ with relaxation term $\delta = 10^{-5}$ are utilized. In practice, the given privacy budgets and relaxation term can provide effective privacy protection. Under the relatively strict privacy constraint ($\epsilon = 0.25, \delta = 10^{-5}$) which corresponds to $\sigma = 0.2$, the model accuracy still reaches more than 94%. Considering privacy preservation and global model accuracy, the schemes in the following experiment ensure $(0.5, 10^{-5})$-DP.

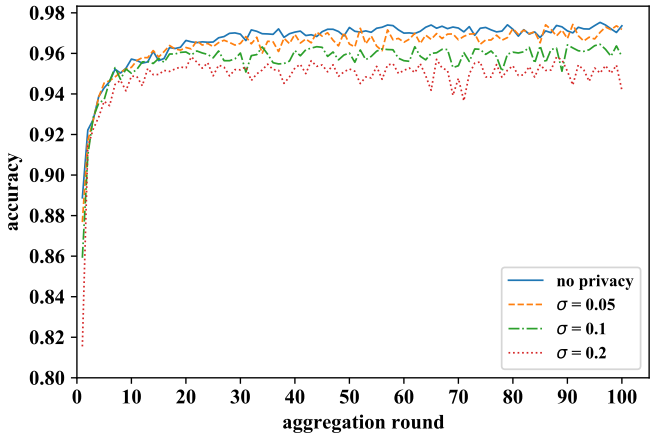


Fig. 4. The model accuracy with different privacy budget

C. Comparison of Different schemes under Two Attacks

To evaluate the robustness of FL-RAEC, two hostile attack models in FL are selected in the test scenarios[39], [40]: extra-noise attack and sign-flip attack. The malicious ESs following the extra-noise attack model will add the noise to weight update, far beyond the requirement of current privacy budgets. And for the sign-flip attack model, the malicious ESs will flip the sign of weight before adding noise and uploading weight updates.

The compared schemes are as follows.

- FedAVG[39]: FedAVG calculates the average of all uploaded weight updates as the aggregated result, then updates the weight of the global model with the aggregated result.
- median-AG[41]: Median-AG selects the median of all uploaded weight updates as the aggregated result to update the weight of the global model.
- trmean-AG[42]: Trmean-AG clips the uploaded weight update according to the threshold, which is calculated by the assumed proportion of attackers. Afterward, trmean-AG calculates the average of the clipped result as the weight update of the global model.

Fig.5 depicts the comparison of different schemes in extra-noise attack scenarios, where the malicious ESs account for 20% of the overall ESs. FL-RAEC has increased robustness than other schemes and performed almost the same as the benchmark without attacks on accuracy. In extra-noise attack scenarios, the accuracy of trmean-AG increases steadily, while the accuracy of the other two schemes shows a varying degree of decline as the aggregation progresses. For trmean-AG, due to the requirement to set a fixed threshold of

attacker proportion, which is a priori knowledge for FL in edge computing, trmean-AG cannot guarantee to filter out all malicious updates. For FedAVG and median-AG, their aggregation schemes are similar, which are taking the average and the intermediate value as the aggregation result. Since huge noise is constantly added to the weight update during aggregation, the boundary of weight is amplified gradually. Correspondingly, the error caused by the noise is multiplied as well. Therefore, the superposition of errors makes the accuracy of the two schemes continue to decline. And FedAVG, which averages all weight updates, is undoubtedly more susceptible to updates that deviate from the normal range.

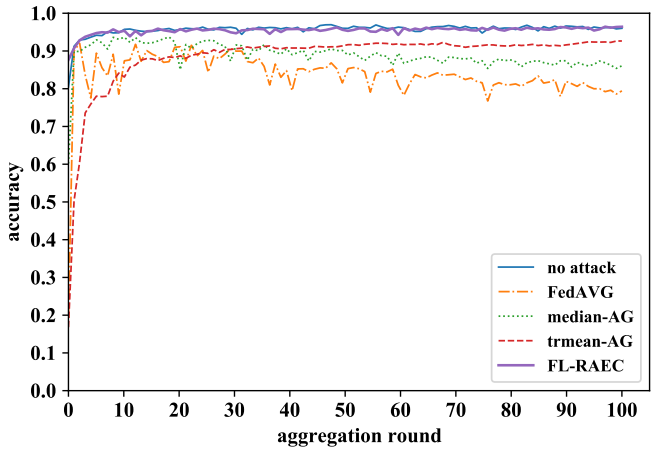


Fig. 5. The model accuracy of different schemes under extra-noise attack

Fig.6 depicts the comparison of different schemes in sign-flip attack scenarios, where the malicious ESs account for 20% of the overall ESs. Unlike extra-attack, the flipped weight makes the model inference perform completely differently. Once the update with the flipped sign is added to the aggregated model, it will greatly break the original model construction, which explains why the results show great fluctuations. Each steep decline is caused by excessive flipped weight updates being introduced into the aggregation. Except for trmean-AG, the accuracy of other schemes fluctuates greatly in the early stage. Trmean-AG excludes abnormal updates by a certain percentage before aggregation, so its accuracy is more stable throughout the aggregation process than other schemes. On the contrary, the accuracy of median-AG is at a low level for most of the period. The flipped weights make the intermediate value of all data usually near 0, which greatly reduces the effect of aggregation. Owing to the similar amplitude and structure to normal weight, it is difficult for the autoencoder to detect the anomaly, which leads the performance aggregation round to be poor before the 40th round. After the 40th round, which is the round of launching the trust score, the steady of aggregation and accuracy of the global model improve greatly.

D. Performance under Different Proportions of Malicious ESs

1) Model Accuracy: In Fig.7, we discuss the influence of malicious ES proportion on the performance under extra-noise attacks. The three sub-figures illustrate the performance

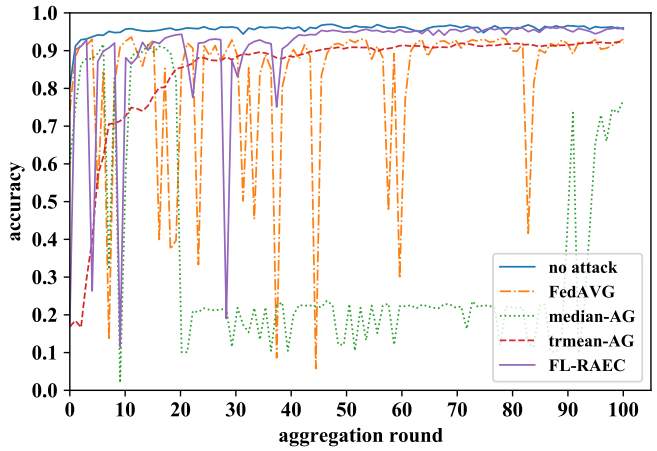
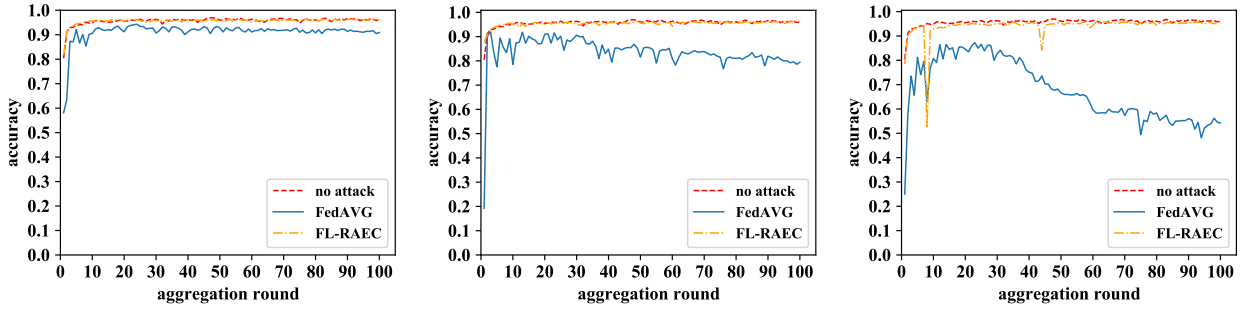


Fig. 6. The model accuracy of different schemes under sign-flip attack

of FL-RAEC when the malicious ES proportion is 10%, 20%, and 40%. With the continuous increase of malicious ES proportion, the performance of FedAVG declines rapidly. When the proportion of malicious ESs is greater or equal to 20%, the model is unable to converge, the accuracy decreasing with the aggregation. And in the worst case, the accuracy of FedAVG drops to about 50% at the end. Note that when the proportion is equal to 40%, the accuracy of FL-RAEC falls back at some early aggregation rounds. In this case, the proportion of malicious participants selected as trainers is nearly half and sometimes exceeds half due to the random ES selection at each round. The μ^{ϕ} of top 50% minimum reconstruction losses given by the autoencoder will become larger, causing the threshold to become loose. So the influence of malicious updates cannot be completely eliminated by the calculated anomaly score. After 40 rounds of aggregation, the impact of malicious updates will be further reduced by the launched trust score.

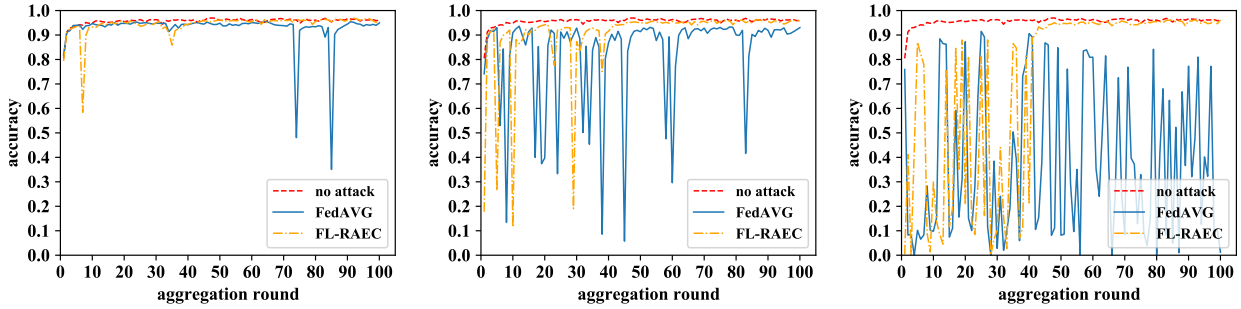
Fig.8 depicts the influence of the malicious ES proportion in sign-flip attack scenarios. The malicious ES proportions of the three sub-figures are the same as Fig.7. In the case of different proportions of attackers, the accuracy of FL-RAEC cannot maintain a steady increase in the early stage. That means the anomaly detection based on autoencoder cannot fully defend against sign flip attacks. In the later stage of training, the accuracy of FL-RAEC is always stable. Although FedAVG has more rounds of a sharp drop in accuracy with an increase in the proportion of malicious ESs, FedAVG can still aggregate to a relatively high accuracy rate when the proportion is 10% and 20%. However, when the proportion reaches 40%, the aggregation of FedAVG is unable to reach the available accuracy. In contrast, FL-RAEC can still maintain stable aggregation and reach high accuracy.

2) Mean Difference Values: To study the specific implementation of anonymous trust verification, the mean difference values of ESs are presented in the form of a 10x10 matrix in Fig. 9. Thereinto, each grid represents the mean difference value of a single ES. According to (30), those ESs with lower mean difference values have a higher probability of being honest participants. For the situation of 10% malicious



(a) 10% malicious ESs (b) 20% malicious ESs (c) 40% malicious ESs

Fig. 7. The model accuracy under different proportions of malicious ESs in extra-noise attack



(a) 10% malicious ESs (b) 20% malicious ESs (c) 40% malicious ESs

Fig. 8. The model accuracy under different proportions of malicious ESs in sign-flip attack

ESs, the mean difference values of most ESs are much lower than 1, and those malicious ESs have a higher level of mean difference value (5∼8) which far exceeds the value of honest ESs. With the increase of malicious ESs, the hot grid gradually occupies the entire heat map. In contrast, the discrepancy of mean difference values between honest ESs and malicious ESs is decreasing. Due to numerous malicious ESs joining training/verification, the mean verification errors will increase, which reduces the difference value of verified ESs after normalization at each round. But even if the proportion of attackers reaches 40%, the mean difference value of those malicious ESs is still between 2-5 which is much higher than the normal value.

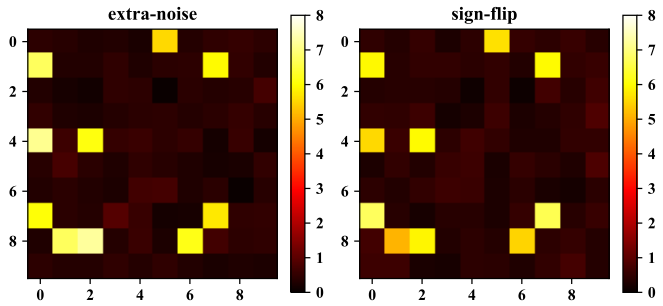
E. Performance with Different Anomaly Impact Factor

As a significant part of FL-RAEC, autoencoder-based anomaly detection secures aggregation to a certain extent in the early stage and guarantees the stable implementation of trust verification in the later stage. Thereinto, the anomaly imparts factor β determines the influence of the anomaly score on the overall score.

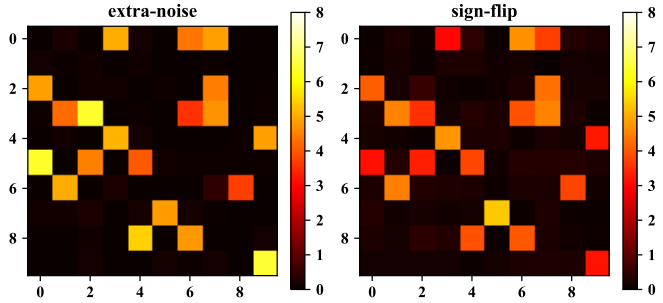
Fig.10 illustrates the performance of FL-RAEC with different β under two attacks. To adequately evaluate the impact of β, the proportion of malicious ESs is set to 40%. When $\beta = 0$, the anomaly detection is inactive in the aggregation and the anomaly scores of all ESs are the same, which are equal to 1. For the scenario under sign-flip attack, the change of β has almost no effect on the aggregation due to the weakness of the autoencoder on the anomaly detection of the sign-flip attack. The accuracy of the model with different β converged to the same level after 40 rounds. In another aspect, whether the anomaly detection is activated has a great impact on the aggregation under extra-noise attack. When β is equal to 0, although the accuracy of the model slowly rebounds due to the launch of the trust score in the later stage, since the malicious updates were not screened in the early stage, abundant noise is introduced into the global model, which makes the subsequent improvement of the accuracy slower. With the activation of anomaly detection ($\beta > 0$), the aggregation tends to be stable with occasional sharp declines and recovers, which is brought by the exceeded numbers of malicious ESs. The increase of β reduces the influence of those weights with higher reconstruction errors to aggregation, expediting the recovery of accuracy from a low level. When $\beta \geq 1$, the accuracy of the model recovers fleetly after the exceeded attack from malicious ESs.

VII. Conclusion and Future Work

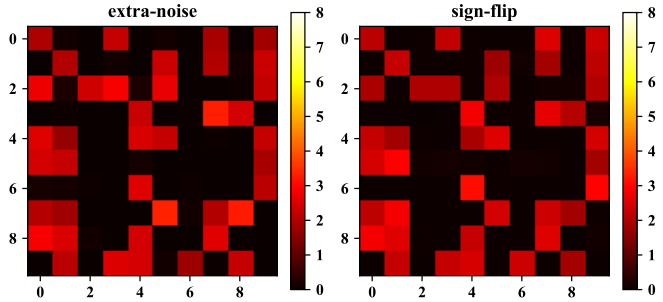
In this paper, we proposed a privacy-preserving FL scheme with robust aggregation in edge computing, named FL-RAEC. In the stage of data uploading, we construct a hybrid privacy-preserving mechanism which consists of asymmetric encryption and LDP to guarantee the parameter integrity and the privacy of weight update. In the stage of model aggregation, we propose a phased model aggregation strategy which is based on anomaly detection and anonymous trust verification. The autoencoder is utilized to execute anomaly detection and the anomaly score is calculated according to the detection result.



(a) 10% malicious ESs



(b) 20% malicious ESs



(c) 40% malicious ESs

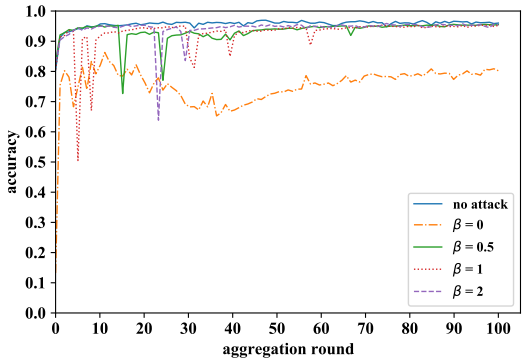
Fig. 9. The mean difference value matrix under different proportions of malicious ESs

Meanwhile, each ES has a trust score through anonymous trust verification, which gradually becomes stable and accurate with the advance of the training. The aggregation is jointly affected by anomaly and trust scores. Eventually, the performance of the proposed scheme is evaluated and compared with other schemes using a real dataset, which proves that FL-RAEC is robust under different attacks, making the global model reach high accuracy.

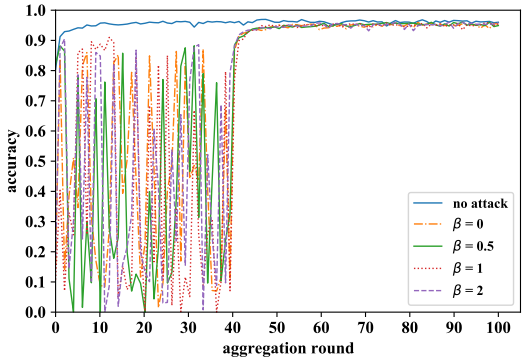
One of the future directions of the research is to implement FL-RAEC on heterogeneous and highly mobile edge devices. Considering the complex network environment and unreliability of a single device, asynchronous FL may be more suitable in this case.

ACKNOWLEDGMENT

This research is supported by the Financial and Science Technology Plan Project of Xinjiang Production and Construction Corps under grant no. 2020DB005, and the National Natural Science Foundation of China under grant no.61872219.



(a) extra-noise attack



(b) sign-flip attack

Fig. 10. Performance with different anomaly impact factor

This research is also supported by the Future Network Scientific Research Fund Project under grant no. FNSRFP-2021-YB-18 and the Priority Academic Program Development of Jiangsu Higher Education Institutions (PAPD) fund.

REFERENCES

[1] S. Verma, Y. Kawamoto, Z. M. Fadlullah, H. Nishiyama, N. Kato, A survey on network methodologies for real-time analytics of massive iot data and open research issues, IEEE Communications Surveys & Tutorials 19 (3) (2017) 1457–1477.

[2] C. G. C. Index, Forecast and methodology, 2016–2021 white paper, Updated: February 1 (2018).

[3] Y. Dai, D. Xu, S. Maharjan, G. Qiao, Y. Zhang, Artificial intelligence empowered edge computing and caching for internet of vehicles, IEEE Wireless Communications 26 (3) (2019) 12–18.

[4] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, K. Chan, Adaptive federated learning in resource constrained edge computing systems, IEEE Journal on Selected Areas in Communications 37 (6) (2019) 1205–1221.

[5] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar, A. Y. Zomaya, Edge intelligence: the confluence of edge computing and artificial intelligence, IEEE Internet of Things Journal 7 (8) (2020) 7457–7469.

[6] S. B. Calo, M. Touna, D. C. Verma, A. Cullen, Edge computing architecture for applying ai to iot, in: 2017 IEEE International Conference on Big Data (Big Data), IEEE, 2017, pp. 3012–3016.

[7] P. Mach, Z. Becvar, Mobile edge computing: A survey on architecture and computation offloading, IEEE Communications Surveys & Tutorials 19 (3) (2017) 1628–1656.

[8] L. Li, M. Siew, T. Q. Quek, Learning-based pricing for privacy-preserving job offloading in mobile edge computing, in: ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2019, pp. 4784–4788.

[9] P. Voigt, A. Von dem Bussche, The eu general data protection regulation (gdpr), A Practical Guide, 1st Ed., Cham: Springer International Publishing 10 (2017) 3152676.

[10] Y. Lu, X. Huang, Y. Dai, S. Maharjan, Y. Zhang, Federated learning for data privacy preservation in vehicular cyber-physical systems, IEEE Network 34 (3) (2020) 50–56.
[11] M. Song, Z. Wang, Z. Zhang, Y. Song, Q. Wang, J. Ren, H. Qi, Analyzing user-level privacy attack against federated learning, IEEE Journal on Selected Areas in Communications 38 (10) (2020) 2430–2444.
[12] H. Sedjelmaci, F. Guenab, S.-M. Senouci, H. Moustafa, J. Liu, S. Han, Cyber security based on artificial intelligence for cyber-physical systems, IEEE Network 34 (3) (2020) 6–7.
[13] Y. Lu, X. Huang, Y. Dai, S. Maharjan, Y. Zhang, Differentially private asynchronous federated learning for mobile edge computing in urban informatics, IEEE Transactions on Industrial Informatics 16 (3) (2019) 2134–2143.
[14] A. Bhowmick, J. Duchi, J. Freudiger, G. Kapoor, R. Rogers, Protection against reconstruction and its applications in private federated learning, arXiv preprint arXiv:1812.00984 (2018).
[15] L. Melis, C. Song, E. De Cristofaro, V. Shmatikov, Exploiting unintended feature leakage in collaborative learning, in: 2019 IEEE Symposium on Security and Privacy (SP), IEEE, 2019, pp. 691–706.
[16] J. So, B. Güler, A. S. Avestimehr, Byzantine-resilient secure federated learning, IEEE Journal on Selected Areas in Communications (2020).
[17] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, V. Shmatikov, How to backdoor federated learning, in: International Conference on Artificial Intelligence and Statistics, PMLR, 2020, pp. 2938–2948.
[18] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, C. Miao, Federated learning in mobile edge networks: A comprehensive survey, IEEE Communications Surveys & Tutorials 22 (3) (2020) 2031–2063.
[19] L. Cui, X. Su, Z. Ming, Z. Chen, S. Yang, Y. Zhou, W. Xiao, Creat: Blockchain-assisted compression algorithm of federated learning for content caching in edge computing, IEEE Internet of Things Journal (2020).
[20] S. Luo, X. Chen, Q. Wu, Z. Zhou, S. Yu, Hfel: Joint edge association and resource allocation for cost-efficient hierarchical federated edge learning, IEEE Transactions on Wireless Communications 19 (10) (2020) 6535–6548.
[21] S. Yu, X. Chen, Z. Zhou, X. Gong, D. Wu, When deep reinforcement learning meets federated learning: Intelligent multi-timescale resource management for multi-access edge computing in 5g ultra dense network, IEEE Internet of Things Journal (2020).
[22] M. M. Amiri, D. Gündüz, Federated learning over wireless fading channels, IEEE Transactions on Wireless Communications 19 (5) (2020) 3546–3557.
[23] G. Zhu, Y. Wang, K. Huang, Broadband analog aggregation for low-latency federated edge learning, IEEE Transactions on Wireless Communications 19 (1) (2019) 491–506.
[24] T. Wang, Z. Cao, S. Wang, J. Wang, L. Qi, A. Liu, M. Xie, X. Li, Privacy-enhanced data collection based on deep learning for internet of vehicles, IEEE Transactions on Industrial Informatics 16 (10) (2019) 6663–6672.
[25] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. Quek, H. V. Poor, Federated learning with differential privacy: Algorithms and performance analysis, IEEE Transactions on Information Forensics and Security 15 (2020) 3454–3469.
[26] M. Hao, H. Li, X. Luo, G. Xu, H. Yang, S. Liu, Efficient and privacy-enhanced federated learning for industrial artificial intelligence, IEEE Transactions on Industrial Informatics 16 (10) (2019) 6532–6542.
[27] G. Xu, H. Li, Y. Zhang, S. Xu, J. Ning, R. Deng, Privacy-preserving federated deep learning with irregular users, IEEE Transactions on Dependable and Secure Computing (2020).
[28] Y. Khazbak, T. Tan, G. Cao, Mlguard: Mitigating poisoning attacks in privacy preserving distributed collaborative learning, in: 2020 29th International Conference on Computer Communications and Networks (ICCCN), IEEE, 2020, pp. 1–9.
[29] A. R. Short, H. C. Leligou, M. Papoutsidakis, E. Theocharis, Using blockchain technologies to improve security in federated learning systems, in: 2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC), IEEE, 2020, pp. 1183–1188.
[30] A. K. Singh, A. Blanco-Justicia, J. Domingo-Ferrer, D. Sánchez, 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.
[31] Y. Qu, L. Gao, T. H. Luan, Y. Xiang, S. Yu, B. Li, G. Zheng, Decentralized privacy using blockchain-enabled federated learning in fog computing, IEEE Internet of Things Journal 7 (6) (2020) 5171–5183.
[32] C. Dwork, Differential privacy: A survey of results, in: International conference on theory and applications of models of computation, Springer, 2008, pp. 1–19.
[33] P. Kairouz, S. Oh, P. Viswanath, Extremal mechanisms for local differential privacy, The Journal of Machine Learning Research 17 (1) (2016) 492–542.
[34] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, L. Zhang, Deep learning with differential privacy, in: Proceedings of the 2016 ACM SIGSAC conference on computer and communications security, 2016, pp. 308–318.
[35] C. Dwork, G. N. Rothblum, S. Vadhan, Boosting and differential privacy, in: 2010 IEEE 51st Annual Symposium on Foundations of Computer Science, IEEE, 2010, pp. 51–60.
[36] A. Borghesi, A. Bartolini, M. Lombardi, M. Milano, L. Benini, Anomaly detection using autoencoders in high performance computing systems, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 2019, pp. 9428–9433.
[37] C. Fung, C. J. Yoon, I. Beschastnikh, Mitigating sybils in federated learning poisoning, arXiv preprint arXiv:1808.04866 (2018).
[38] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324.
[39] B. McMahan, E. Moore, D. Ramage, S. Hampson, B. A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: Artificial Intelligence and Statistics, PMLR, 2017, pp. 1273–1282.
[40] L. Li, W. Xu, T. Chen, G. B. Giannakis, Q. Ling, Rsa: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 2019, pp. 1544–1551.
[41] Y. Chen, L. Su, J. Xu, Distributed statistical machine learning in adversarial settings: Byzantine gradient descent, Proceedings of the ACM on Measurement and Analysis of Computing Systems 1 (2) (2017) 1–25.
[42] D. Yin, Y. Chen, R. Kannan, P. Bartlett, Byzantine-robust distributed learning: Towards optimal statistical rates, in: International Conference on Machine Learning, PMLR, 2018, pp. 5650–5659.