TYPE Original Research
PUBLISHED 22 December 2022
DOI 10.3389/fphy.2022.1102892

Intrusion detection framework based on homomorphic encryption in AMI network

Jing Wang[1], Zhuoqun Xia[1], Yaling Chen[2,1]*, Chang Hu[1] and Fei Yu[1]

[1]School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha, China, [2]School of Electrical and Information Engineering, Hunan Institute of Traffic Engineering, Hengyang, China

OPEN ACCESS

EDITED BY
Chunbiao Li,
Nanjing University of Information Science and Technology, China

REVIEWED BY
Qiang Lai,
East China Jiaotong University, China
Yuexi Peng,
Xiangtan University, China

*CORRESPONDENCE
Yaling Chen,
chenyaling0307@163.com

SPECIALTY SECTION
This article was submitted to Interdisciplinary Physics, a section of the journal Frontiers in Physics

RECEIVED 19 November 2022
ACCEPTED 05 December 2022
PUBLISHED 22 December 2022

CITATION
Wang J, Xia Z, Chen Y, Hu C and Yu F (2022), Intrusion detection framework based on homomorphic encryption in AMI network.
Front. Phys. 10:1102892.
doi: 10.3389/fphy.2022.1102892

In order to alleviate the privacy issue of traditional smart grids, some researchers have proposed a power metering system based on a federated learning framework, which jointly trains the model by exchanging gradients between multiple data owners instead of raw data. However, recent research shows that the federated learning framework still has privacy and security issues. Secondly, since the server does not have direct access to all parties data sets and training process, malicious attackers may conduct poisoning attacks. This is a new security threat in federated learning - poisoning attack. However, solving the two problems at the same time seems to be contradictory because privacy protection requires the inseparability of the training gradients of all parties, and security requires the server to be able to identify the poisoned client. To solve the above issues, this paper proposes an intrusion detection method based on federated learning client-side security in AMI networks, which uses CKKS to protect model parameters. In addition, to resist the poisoning attack in federated learning, the model trained by the data processing center and the model trained by each client are firstly calculated for the direction similarity, and the similarity value is scaled as the adaptive weight of the aggregation model. Then, the size of each client model update is normalized to be the same size as the data processing center model update. Finally, the normalized updates and adaptive weights are weighted averaged to form a global model update. The research results show that the method in this paper can effectively resist inference attacks and poisoning attacks. In the AMI network, the intrusion detection method based on federated learning can maintain a good detection performance.

KEYWORDS

advanced metering infrastructure, intrusion detection, federated learning, homomorphic encryption, privacy protection, poisoning attack

1 Introduction

Because smart grid Advanced Metering Infrastructure (AMI) is an important part of the power system, and the power system is also related to information security and computer network, it is vulnerable to various network threats, and advanced technologies need to be adopted to protect the security of smart grid AMI. With the gradual development of machine learning technology, machine learning is widely used in the field of smart grid, such as intrusion detection Li et al. [1], electricity stealing detection Hasan et al. [2], private data sharing Su et al. [3], and so on. However, despite the rapid development of machine learning-based power systems, their privacy and security problems are still exposed. Federated learning is proposed as a promising privacy-preserving method, which can jointly learn a common machine learning model in a decentralized training manner under the coordination of a central server. During this training process, each participant uses the local data set to learn the model parameters, and the local data does not need to be uploaded to the data center, so it will not be exposed. All in all, federated learning can protect the privacy of training data and detection data to a certain extent. However, recent work shows that federated learning still confront many privacy and security issues.

From a privacy perspective, direct communication of gradients may reveal sensitive information. Whether it is a semi-honest server or a third party, once the gradient information is obtained, a large amount of sensitive information about the client may be obtained through inference, thereby revealing the privacy of the client. From a security point of view, if there are malicious actors in the federated training process, they can perform poisoning attacks on data or models, send poisonous local model updates to the server, or insert hidden backdoors into the global model. So in order to build a privacy and security federated learning framework, these two problems must be settled.

In order to deal with the privacy attacks faced by federated learning, researchers have proposed many solutions, which are mainly based on the following technologies: methods such as differential privacy, secure multi-party computation, and homomorphic encryption. For federated learning poisoning attack defense, the main idea is to eliminate malicious models before local model parameters are aggregated. Then the corresponding defense methods are as follows: similarity-based defense, statistics-based defense, performance-based defense, and feature extraction-based defense. Both the privacy and poisoning attacks of federated learning are solved by constructing linear or non-linear functions. The machine learning model is composed of linear and non-linear models. For example, the neural network (Neural Network,NN) used in the machine learning model is a mathematical model that simulates the structure and function of biological networks and is composed of neurons and synapses. Among them, neurons have complex non-linear characteristics and are the basic unit of biological information processing nervous system [4–6].

In order to solve the privacy and poisoning attack security problems of smart grid AMI federated intrusion detection concurrently, this paper proposes a privacy protection method using homomorphic CKKS encryption. When the client upload gradient is not obtained by the untrustworthy, we calculate the directional similarity between the client training model and the data center training model, and then extract the similarity by using a non-linear function (logarithmic function). Specifically, this paper makes three contributions:

1) This paper adopts a privacy-enhanced federated learning framework based on CKKS in the smart grid AMI network, which can prevent malicious clients from inferring client information through their uploaded gradients (malicious clients may be sham clients injected by attackers or real clients compromised by attackers), thereby revealing client privacy. It also prevents semi-honest data centers and control servers from violating client privacy.
2) This paper improves a poisoning attack defense method. First, use the Pearson correlation coefficient to calculate the directional similarity between the model trained by each client and the model trained by the data center, and then scale the similarity value. The data center model is obtained by training on a clean dataset. Second, the gradients are normalized to avoid malicious attackers from affecting the global model by enlarging the gradient size. Finally, the scaled similarity is used as the adaptive weight to calculate the aggregation model.
3) This paper provides algorithm, security analysis and tests on the dataset to prove that the proposed scheme can resist the Trim attack of federated learning.

The rest of this article is organized as follows. In Section 2, we analyze related work. In Section 3, the relevant knowledge is needed to supplement the research content. In Section 4, we make a problem statement. Section 5 then details a privacy-enhanced robust federated learning framework. Next, analysis and performance evaluation are performed in Section 6. Finally, Section 7 concludes the paper.

2 Related work

2.1 Research status of federated learning privacy issues

For user privacy in federated learning, the existing work mainly focuses on guaranteeing the secrecy of gradients. These solutions are mainly based on the following techniques: Differential Privacy [7–9], Secure Multi-Party Computation [10–12], Homomorphic Encryption [13–15] and other methods.

Wu et al. [7] proposed a new Local Differential Private (LDP) algorithm that redesigned the training process. This enables data owners to add layers of randomization before the data leaves and reaches potentially untrusted machine learning services. Kumar et al. [8] proposed Privacy-aware and asynchronous Deep-Learning-assisted IoT applications (PADL) to enable several data collection sites to cooperatively train deep neural networks (Deep Neural Networks). Neural Networks, DNN) while maintaining the secrecy of private data. Liu et al. [9] proposed an adaptive privacy-preserving federated learning framework APFL. These differential privacy-based methods may result in lower federated learning accuracy.

Bonawitz et al. [10] proposed a novel, communication-efficient, fault-robust protocol for secure aggregation of high-dimensional data. Mohassel and Zhang [11] proposed SecureML, which conducts privacy-preserving learning through Security Multi-Party Computation (SMC). During the initial setup phase, data owners operate, encrypt, and secretly share their data between two non-colluding servers. Riazi et al. [12] pointed out an SMC-based protocol that enables multiple participants to collaboratively compute an agreed-upon function without revealing any participant's input information except that which can be inferred from the computation results. As a result, secure multi-party computation will not only lead to large communication overhead, but also cannot fully guarantee the prevention of information leakage.

Aono et al. [13] conducted model training through asynchronous SGD, and then homomorphically encrypted the gradient, which could obtain high accuracy while protecting the security of federated learning. Xu et al. [14] designed a new solution to decrease the negative influence of irregular users on training accuracy, thereby guaranteeing that the training results are mainly computed from the contribution of high-quality data, while using chaotic circuits and additional homomorphic ciphers system to ensure the confidentiality of all user-related information. Fang and Qian [15] encrypted the uploaded gradients using an improved paillier algorithm, which is almost the same as the training accuracy of the multi-party privacy-preserving machine learning framework.

According to the analysis, this paper proposes a homomorphic CKKS encryption method to protect the privacy of the gradient uploaded by the client.

2.2 Research status of federated learning poisoning attack

In order to solve the federated learning poisoning attack, researchers have proposed some defense schemes, which are divided into four types: similarity-based defense, statistics-based defense, performance-based defense and feature extraction-based defense.

Similarity-based defense is to design aggregation algorithms based on the similarity between gradients, such as Krum Blanchard et al. [16] for Euclidean distance, which selects one gradient with the lowest predefined score as the aggregated gradient from all uploaded gradients. Xia et al. [17] iteratively filter the models that are farthest from the mean of the remaining models, taking the mean of the last remaining models as the aggregated model. These methods are more effective when there are few malicious attackers. However, when there are many attackers colluding with each other, they will lead to choosing a malicious value as the comparison standard, then the similarity calculation value lacks reliability.

Statistics-based defense is to use statistical features such as median and mean to circumvent malicious model parameters. Yin et al. [18] first eliminated some extreme values and selected the median result of the corresponding dimension of the client model as the aggregated model. Chen et al. [19] used the gradient dimension median as a global update. These methods can bypass malicious models with high probability by selecting the median or mean as the aggregated result. However, the global model lacks a lot of normal gradient information, which weakens the accuracy of the model.

Fang et al. [20] argue that neither of the above two aggregation rules is efficacious enough for an opponent with certain knowledge, and they can carefully design a similar set of gradients to confuse the aggregation rules.

Performance-based defense is to use auxiliary datasets to test the accuracy, loss value, *etc.* Of each client model to eliminate malicious models. Xie et al. [21] used the validation dataset to directly test the loss value and modulo length of the model, and calculate the score of each model to remove outliers. Zhao et al. [22] assign each client's submodel to other clients for cross-validation to find toxic updates. These defenses are to use a new dataset on the server to test the client's model for anomalies.

Feature extraction-based defenses can extract features from high-dimensional models and then discriminate feature spaces. Zhao et al. [22] directly decomposed the uploaded model by Principal Components Analysis (PCA), and then considered the model corresponding to the principal component to be the normal model. Nguyen et al. [23] introduced the FLAME defense framework for estimating a sufficient amount of noise to inject to ensure backdoor elimination. To minimize the amount of noise required, model clustering and weight clipping methods are also used to maintain good performance of the aggregated model. A problem with these methods is that which type represents normal and which type represents abnormal, and a benchmark needs to be determined in advance, which is a technical problem.

3 Preliminaries

3.1 Homomorphic encryption

Homomorphic encryption is an encryption scheme that can still perform homomorphic computations in an encrypted situation. Under the same operation rules, the result of its ciphertext calculation is the same as the ciphertext of the plaintext calculation result. According to different operation rules, homomorphic encryption can be divided into two types: one is additive homomorphism and the other is multiplicative homomorphism. According to the supported operation types and operation times, homomorphic encryption schemes can be divided into three types: Partially Homomorphic Encryption (PHE), somewhat Homomorphic Encryption (SHE), and Fully

Homomorphic Encryption (FHE). Some of the homomorphic encryption methods can only perform one type of homomorphic calculation, addition or multiplication. Somewhat homomorphism supports addition and multiplication operations on ciphertext at the same time, but can only perform ciphertext operations for a limited number of times. While the homomorphic encryption method can simultaneously perform addition and multiplication homomorphic calculations, and can support infinite ciphertext operations. CKKS Cheon et al. [24] is a fully homomorphic encryption method, which supports the addition and multiplication of floating-point vectors in the ciphertext space. Homomorphic encryption CKKS consists of key generation (KeyGen), key distribution (DisKey), encryption algorithm (Enc), and decryption algorithm (Dec).

- **KeyGen**: Responsible for the generation of the public key (*PK*) and private key (*SK*) of the entire system.
- **DisKey**: Responsible for distributing keys.
- **Encryption**: Indicates the encryption function, which uses the public key to encrypt the plaintext to attain the ciphertext.
- **Decrypt**: Indicates the decryption function, which uses the private key to decrypt the ciphertext to obtain the corresponding plaintext.

The homomorphic encryption CKKS used satisfies the additive homomorphic property and the multiplicative homomorphic property as shown in Equations 1, 2.

$$E(x_1) + E(x_2) = E(x_1 + x_2) \tag{1}$$

$$E(x_1)*r = E(r*x_1) \tag{2}$$

3.2 Pearson correlation coefficient

The Pearson correlation coefficient is a method of measuring the similarity between vectors, which measures whether the vectors are linearly related. In addition, it can be regarded as the cosine similarity after centering the vectors. The disadvantage of cosine similarity is that the two vectors calculated cannot be empty in each dimension, and the Pearson correlation coefficient overcomes this defect.

When there are two vectors $X = [x_1, x_2, \ldots, x_n]$, $Y = [y_1, y_2, \ldots, y_n]$, the similarity calculation between vector X and vector Y is shown in formula (3).

$$\rho(X,Y) = \frac{Cov(X,Y)}{\sigma(X)*\sigma(Y)} = \frac{\sum_{i=1}^{n}(x_i - \bar{X})*\sum_{i=1}^{n}(y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{X})^2}*\sqrt{\sum_{i=1}^{n}(y_i - \bar{Y})^2}} \tag{3}$$

The Pearson correlation coefficient value $\rho(X, Y)$ is between -1 and 1. Different values of the Pearson correlation coefficient indicate that the two vectors have different relationships, and different value ranges represent different correlation strengths of the vectors.

When the Pearson correlation coefficient value is 0, it means that there is no linear correlation between the two vectors.

When the value of the Pearson correlation coefficient is between (-1, 0), it means that when the value of one vector increases (decreases), the value of the other vector decreases (increases), indicating an inverse relationship between them.

When the value of the Pearson correlation coefficient is between (0,1), it means that when the value of one vector increases (decreases), the value of the other vector increases (decreases), indicating the same change between them.

It is assumed that the larger the absolute value of the correlation coefficient of two vectors, the stronger their correlation. And the closer their correlation coefficient values are to 0, the weaker the correlation between the two vectors, as shown in Table1.

This paper uses the Pearson correlation coefficient to compute the correlation between the federated learning data center training model and the client-side training model. When the correlation coefficient value is less than 0.25, we consider that the model trained by the client has nothing to do with the model trained by the data center, and set the aggregation weight of the model trained by the client to 0, thereby suppressing the poisoning model. Here we have tested and set the Pearson correlation coefficient to 0.25, 0.5 equivalent. It is found that when the Pearson correlation coefficient is 0.25, the effect is better. The focus of this paper is to use CCKS and similarity based robust aggregation algorithm to build our intrusion detection framework and implement intrusion detection. Therefore, the setting process of Pearson correlation coefficient correlation value is not shown in the paper.

4 Problem statement

4.1 Threat model

When the intrusion detection method based on federated learning is deployed in the smart grid AMI network, there may still be security risks. For example, an attacker may want to obtain client information, and can intercept the model updates uploaded by each client to conduct inference attacks. Other attackers want to influence certain decisions of smart grid AMI by reducing global model performance. Among them, these clients may be sham clients injected by attackers, or real clients invaded by attackers. But the attacker will not compromise the data center and control server (too expensive), which means that the data center and control server can honestly do everything correctly and according to the regulations. Client privacy may also be compromised as data centers and control servers have access to model updates for individual clients. Therefore, we consider the data center and control server to be semi-honest, and furthermore, assume that the four entities do not collude with each other.

When the attacker has the following information about the FL system: training data, local updates, loss function and learning

TABLE 1 Correlation table.

Absolute value	0–0.2	0.2–0.4	0.4–0.6	0.6–0.8	0.8-1
Correlation	extremely weak	weak	medium	Strong	extremely Strong

rate on the client. When the attacker knows the training data of the client, he may carry out a data poisoning attack; When an attacker knows the local update, loss function and learning rate of the client, he has the ability to modify the local model, that is, to poison the model. We believe that the proposed method can defend against attacks such as Trim attacks initiated by attackers possessing the above information.

4.2 Client security objective

The goal of this paper is to design a scheme that achieves robustness against poisoning attacks launched by malicious clients while preserving privacy and without loss of accuracy. Specifically, we have three design goals:

- **Privacy**: An attacker can reveal privacy by inferring gradients or parameters to recover training samples. In order to protect the privacy of users, it is proposed to protect the gradient uploaded by the client.
- **Robustness**: refers to the presence of attacks, the proposed method can still maintain the classification accuracy of the global model. That is, regardless of whether there is an attack, the proposed method should have the same performance as FedAvg without an attack.
- **Accuracy**: Poisoning attacks cause model performance degradation by poisoning datas or models, such as inaccurate predictions, misclassifications, *etc.* Therefore, the federated learning intrusion detection method must guarantee that the accuracy is within a rational range.

4.3 System model of smart grid AMI

As shown in Figure 1, the system model of this AMI has four types of entities:

- **Key Generation Center (KGC)**: An independent and trusted third-party agency responsible for distributing and managing the public and private keys required by the AMI federation system.
- **Clients**: The data owner trains a unified model under the coordination of the data center. Based on security considerations, the client uses its own data for local training, and then uploads the cryptographic gradients or parameters to the data center, and we presume that the data of each client is independent and identically distributed.
- **Data Center (DC)**: Receives model updates from individual clients and aggregates them. DC has a small dataset of no attack samples, which provides a basis of trust for the model to resist poisoning attacks.
- **Control Server (CS)**: completes the aggregation work together with the DC, and has the public-private key pair (pk_c, sk_c) generated by the KGC for encryption and decryption.

For ease of reading, symbols and descriptions appearing in this paper are listed in Table 2.

5 Federated learning intrusion detection method based on homomorphic encryption

In this paper, the full connected neural network is used to train the detection model, and the trained model parameters or gradients are floating point, while homomorphic encryption CKKS can support the addition and multiplication of floating point vectors in the ciphertext space. Therefore, In order to settle the issue that the client model update of each client suffers from inference attacks and leaks user privacy, this paper uses the homomorphic encryption method (CKKS) to encrypt the data generated by each entity, so as to protect the client upload gradient from being obtained by untrusted people. A trusted key generation center generates keys for individual entities. All authorized entities can access the public key generated by KGC for other entities, and the private key generated by KGC for an entity is kept by that entity. For example, the key (pk_c, sk_c) generated by KGC for the control server, in which all authorized entities (each client, data center) can use the public key of the control server, and the private key is stored by CS.

This paper believes that there is a deviation between the malicious gradient vector and the benign gradient vector, and judges whether the gradient is malicious according to the similarity with the benign gradient, so that the influence of the malicious gradient can be identified and reduced. This article is based on such a foundation. In order to train an intrusion detection model that can resist poisoning attacks, this paper also considers the model updates of the DC and each client to jointly build a global model. Specifically, in each iteration, the data center trains a benchmark model Cao et al. [25] based on its data set, and performs directional

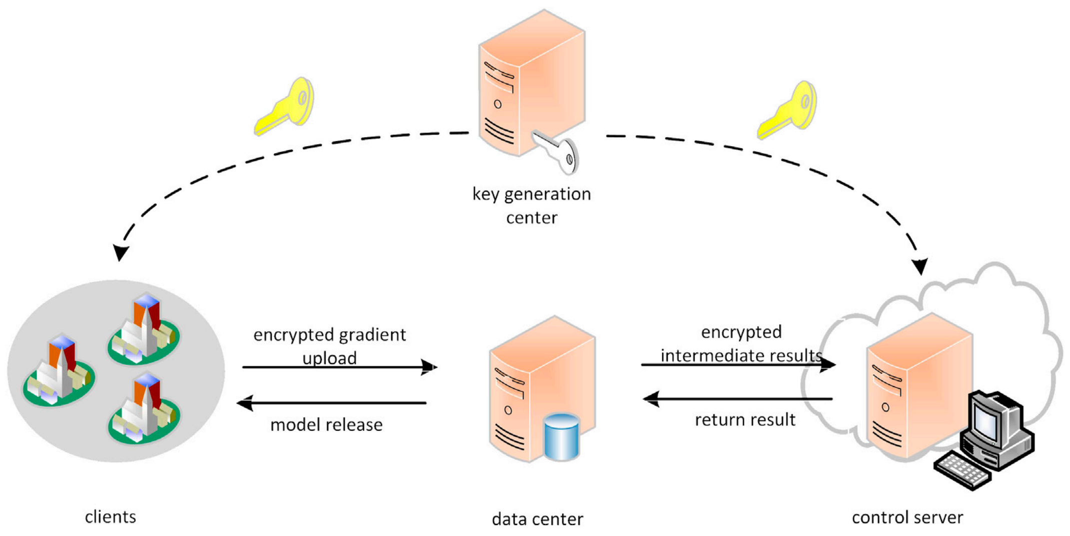

FIGURE 1
Smart grid AMI system model.

TABLE 2 Symbol definition.

Symbol	Describe
Pk	public key
Sk	private key
ω	model weights
G	Gradient
α	learning rate
V	magnified person correlation coefficient
T	Federated Learning Iterations
t	Federated Learning Iteration Index
K	number of clients
A	Number of malicious clients

similarity calculation with the model update of each client, and then scales the calculated value as a model Aggregated adaptive weights. The closer the directions are, the greater the adaptive weight. This protects against attacks where an attacker manipulates the direction of the client's model. At the same time, the model update amplitude of each client is normalized according to the data center model update amplitude, so that it has the same size as the data center model update, so that it can resist the scaling attack of the client model by the attacker. The entire federated learning model building process repeats three steps.

5.1 Client local training

In round t, the client $U_x, x \in [1, 2, \ldots, K]$ participating in the training obtains the global model parameters $[\omega_x^{t-1}]_{pk_x}$ issued by the DC, wherein the global model parameters are encrypted with pk_x. After decryption by the private key sk_x, use the client's data for training to update the model, the expression is $\omega_x^t = \omega_x^{t-1} - \alpha \nabla G_X^{t-1}$. In order to prevent attackers or semi-honest data centers from obtaining updates of each client and compromising user privacy, the gradient $[\omega_x^t]_{pk_c}$ is obtained by encrypting the local update with the public key pk_c of the control server.

5.2 Secure aggregation

The DC interacts with the CS to identify and defend against user-initiated poisoning attacks. Specifically, since the data saved by DC is safe, the gradient $[G_d^t]_{pk_c}$ obtained by its training model is used as a comparison benchmark.

Since an attacker may manipulate the direction of model updates on malicious clients, so that the global model is updated in the opposite direction to the update direction, we use the Person correlation coefficient to calculate the encryption gradient $[G_x^t]_{pk_c}{}_{x=1}^{x=m}$ of the selected m clients in a similar direction to the baseline encryption gradient, as shown in Algorithm 1. To compute the similarity without exposing client privacy, DC blurs the gradients of client and DC by formula (4) and formula (5):

$$E_x = [G_x^t]_{pk_c} * r_x, x \in [1, 2, \ldots, m] \tag{4}$$

$$E_d = [G_d^t]_{pk_c} * r_d \tag{5}$$

where r_x and r_d are two randomly selected non-zero numbers.

```
Input: DC owns $[G_x^t]_{pk_c}$ and $[G_d^t]_{pk_c}, x \in [1,2,\ldots,m]$, CS owns private key $sk_c$
Output: similarity between $[G_x^t]_{pk_c}, x \in [1,2,\ldots,m]$ and $[G_d^t]_{pk_c}$
DC:
  1: Randomly select two non-zero numbers $r_i, i \in \{x,d\}$
  2: Using Equation 4 and 5 to get: $E_i = [G_i^t]_{pk_c} * r_i, i \in \{x,d\}$
  3: Send $E_i$ to CS
CS:
  1: Decrypt $E_i$ using its private key $sk_c$: $D_i = Dec(sk_c, E_i), i \in \{x,d\}$
  2: Calculate the similarity $\rho_{x,d}$between client parameters and data center parameters using formula (6)
```

Algorithm 1. Similarity Calculation

The DC then sends the obfuscated data to the CS, which decrypts using its private key to obtain D_x and D_d. CS uses the Pearson correlation coefficient to calculate the update direction similarity of the decrypted client model gradient and data center model gradient, as shown in formula (6):

$$\rho_{x,d} = \frac{Cov(D_x, D_d)}{\sigma(D_x) * \sigma(D_d)}, x \in [1,2,\ldots,m] \tag{6}$$

Among them, the decrypted data and are still obscure.

After the DC performs the similarity calculation, the Person correlation coefficient is scaled by Eq. 7.

$$v_x = max\left\{0, In\left(\frac{1+\rho_{x,d}}{1-\rho_{x,d}}\right) - 0.5\right\} \tag{7}$$

For those with low similarity, we consider them to be abnormal. Eq. 7 indicates that the higher the correlation is, the more information is extracted by the scaling function, and the lower the correlation is, the less information is extracted by the scaling function. For below a certain value, the aggregation weight v_x is 0.

Since it is possible for an attacker to launch an attack by expanding the scale of client model updates Cao et al. [25], thus affecting the global model update scale, this paper normalizes the magnitude of model updates for each client. The specific operation is to normalize the gradient obtained by each client training model to the same magnitude as the gradient of the DC training model, as shown in Eq. 8.

$$\bar{G}_i = \frac{\|G_d\|}{\|G_i\|} * G_i \tag{8}$$

where G_i represents the gradient of the ith client in the current training round, $\overline{G_i}$ represents the normalized gradient of the ith client, G_d represents the gradient obtained by data center training, and $\|.\|$ represents the L2 norm of a vector. Normalization guaranteess that updates to individual client models do not have too much influence on the aggregated global model. At the same time, the normalization in this paper also enlarges the smaller value of the update of the client to make it the same as the update value of the data center, because we think that the smaller update is more likely to be a normal model. Such normalization can mitigate the influence of malicious models on the global model.

Considering that the higher the similarity, the closer the model parameters are to the model trained by the data center, and the greater the impact on the final model. In this paper, the scaled similarity value is used as the weight of model aggregation, and the parameters of the global model are shown in formula (9).

$$\omega^t = \omega^{t-1} - \alpha \sum_{i \in [1,2,\ldots,m]} \frac{v_i}{\sum_{i \in [1,2,\ldots,m]} v_i} \overline{G_i} \tag{9}$$

where α is the global learning rate.

Algorithm 2 represents safe aggregation based on similarity and normalized gradient magnitude, which enables safe aggregation in the case that neither DC nor CS knows the true gradient of the client and CS does not know the true gradient of the data center.

```
Input: DC owns $[\omega^{t-1}]_{pk_c}$, $[G_x]_{pk_c}, x \in [1,2,\ldots,m]$ and $[G_d]_{pk_c}$, CS owns private key $sk_c$ and $v_x$, $x \in [1, 2, \ldots, m]$
Output: $[G^t]_{pk_c}$
DC:
  1: randomly select $m$ non-zero numbers $r_x$, $x \in [1, 2, \ldots, m]$ and one non-zero positive number $r_d$
  2: Calculate $E_x' = [G_x^t]_{pk_c} * r_x, x \in [1,2,\ldots,m]$ and $E_d' = [G_d^t]_{pk_c} * r_d$
  3: Send $E_i'$ and $E_d'$ to CS
CS:
  1: Decrypt $E_i'$ using its private key $sk_c$: $D_i = Dec(sk_c, E_i'), i \in \{x,d\}$
  2: Use Eq. 7 to obtain $v_x$
  3: Compute the client-side model normalization using Eq. 8 to obtain $r_d * \overline{G_x} = \frac{\|D_d\|}{\|D_x\|} * D_x$
  4: Calculate $e_x = \alpha \frac{v_x}{\sum_{i \in [1,2,\ldots,m]} v_x} * r_d * \overline{G_x}, x \in [1,2,\ldots,m]$
  5: Use CS's public key $pk_c$ for encryption: $E_x = [e_x]_{pk_c}, x \in [1,2,\ldots,m]$
  6: send $\{E_{e_x}\}_{x=1}^{x=m}$ to DC
DC:
  1: Remove noise: $F_x = E_{e_x} * \frac{1}{r_d}, x \in [1,2,\ldots,m]$
  2: Update the global model to: $[\omega^t]_{pk_c} = [\omega^{t-1}]_{pk_c} - (\sum_{i \in [1,2,\ldots,m]} F_x)$
```

Algorithm 2. Safe Aggregation Based on Similarity and Normalized Gradients

5.3 Model distribution

Since the global model parameters after security aggregation are encrypted with the public key of CS, if they are sent directly to the client, the client cannot decrypt it without the private key [22]. Therefore, it is necessary to communicate between the DC and CS, obtain the global model parameter $[\omega^t]_{pk_x}$ re-encrypted by the public key held by all users, and then broadcast the parameter to all clients. The global model security broadcast Algorithm 3 is as follows:

```
Input: DC owns $[\omega^t]_{pk_c}$, CS owns private key $sk_c$
Output: Client public key $pk_x$ encrypted $[\omega^t]_{pk_x}$
DC:
  1: Randomly choose $n$ non-zero numbers $r$
  2: Calculate $R = [\omega^t]_{pk_c} + [r]_{pk_c}$
  3: Send $R$ to CS
CS:
  1: Decrypt $R$ using its private key $sk_c$: $d = Dec(sk_c, R)$
  2: Re-encrypt with public key $pk_x$: $R' = [d]_{pk_x}$
  3: Send $R'$ to DC
DC:
  1: Remove noise: $[\omega^t]_{pk_x} = R' + [-r]_{pk_x}$
  2: broadcast $[\omega^t]_{pk_x}$ to all clients
```

Algorithm 3. Global Model Safe Broadcast Algorithm

6 Analysis and performance evaluation

This section analyzes the algorithm, security and experimental results of the proposed method.

6.1 Analysis of algorithms

1) **Optimization problem**: optimize the algorithm by reducing the number of communications. For Algorithm 1, when calculating the correlation coefficient between the gradient of the data center and the gradient of the client, there are m clients training the model, then call Algorithm 1 m times, and the model parameters trained by the data center will be sent each time, so it can be sent by sending 1-time data center model parameters to optimize. Furthermore, we found that E_i and E_i' in Algorithm 1 and Algorithm 2 are independent of each other and blinded in the same way, where E_d' needs to be blinded with a positive number, so one blinding can be performed in DC (using Algorithm 2 blinded). And reduce the number of communication rounds by sending E_I' to CS in one round.
2) **Correctness of the security aggregation algorithm**: In order to ensure that the proposed scheme can effectively discern malicious gradients, we need to ensure that the data with blinding factors added in Algorithm 1 can be calculated correctly. It can be seen from the literature Cheon et al. [24] that the Pearson correlation coefficient can calculate the blinded variables in the right direction.

6.2 Security analysis

1) This paper discusses the security of three subjects: raw data, model parameters, and Pearson's correlation coefficient. For the original data, in the framework based on federated learning in this paper, the security of the raw data has been greatly guaranteed. The data analysis platform in the AMI network is transferred from the data center to the concentrator, and the small distance transmission of the original data from the smart meter to the concentrator reduces the risk of long-distance movement from the smart meter to the data center, so the privacy of data has been greatly protected. For model parameters, the model parameters trained using the CKKS encryption concentrator are then uploaded to the data center for aggregation. In order to prevent the semi-honest data center from leaking client privacy, let the control server and the data center aggregate model parameters together, so that neither the data center nor the control server can know the original data of the model parameters. This paper believes that it is unnecessary to protect the privacy of the Pearson correlation coefficient. Although the control server can directly obtain the Person correlation coefficient, since the control server is semi-honest, it will perform operations correctly as required, so it will not change the correlation coefficient value. Although there is literature showing that the similarity between gradients contains more information about the training data than the gradient values, samples can be reconstructed using gradient similarity. However, this method relies on the sign of the gradient, and in the context of homomorphic encryption, the method of obtaining data information through gradient similarity is ineffective Cheon et al. [24].
2) Discuss security when it comes to secure aggregation and model distribution. During secure aggregation, the client first uses CKKS to encrypt its training parameter $[G_x^t]_{pk_c}$, and then sends it to the DC. The DC performs a homomorphic operation on the encrypted parameters (to achieve the purpose of fuzzy parameters) to obtain $E_i = [G_i^t]_{pk_c}{}^{\star} r_i, i \in \{x, d\}$, and sends it to the CS. After the CS is decrypted, the similarity calculation is performed to obtain the similarity value $\rho_{x,d}$. CS calculates the aggregation weight v_x according to the similarity value, then CS normalizes the client model to obtain $r_d{}^{\star}\overline{G_x}$, then calculates the aggregated gradient $e_x = \alpha \frac{v_x}{\sum_{x \in [1,2,\ldots,m]} v_x} r_d{}^{\star}\overline{G_x}$

with noise added, and then encrypts it and sends it to DC. After DC removes noise, the model is updated. In this aggregation process, except for Person correlation coefficient and aggregation weight, DC and CS have no direct contact with parameters, and all parameters are invisible during the aggregation process, so the data in the aggregation process is safe. Among them, Person correlation coefficient privacy does not need to be protected (explained in the previous paragraph), and v_x is obtained by Person correlation coefficient, so it does not need to be protected, so the secure aggregation process is safe. When distributing the model, the model $[\omega^t]_{pk_c}$ owned by the DC is encrypted with the CS public key. The homomorphic encryption operation of $[\omega^t]_{pk_c}$ is performed to obtain $R = [\omega^t]_{pk_c} + [r]_{pk_c}$, and R is sent to CS. CS re-encrypts it to obtain $R' = [d]_{pk_x}$, and then sends it to DC, DC After removing the noise, the $[\omega^t]_{pk_x}$ is obtained and sent to each client. Neither DC nor CS obtains the original model ω^t in this distribution process, so the model distribution process is safe.

6.3 Performance evaluation

6.3.1 Experimental setup

1) **Dataset**: In order to evaluate the performance of the intrusion detection method based on federated learning in the smart grid AMI network, this paper adopts the NSL-KDD dataset for testing, and the data includes training set and test set. The data set is randomly allocated on the client side, and the data center has non-attack NSL-KDD data. It is assumed that the sample distribution of the data center is biased towards a certain type, that is, the proportion of the samples of the data center to a certain type of samples is q, and the other types the sample probability is $(1 - q)/4$, and the remaining types of samples have the same proportion, this q is called the data center sample bias.
2) **Evaluated Poisoning Attack**: Trim attack is a kind of off-target local model poisoning attack optimized for Trim mean and median aggregation rules. Because attackers poison the client's data, it is also reflected in the impact training model. Therefore, this paper uses Trim attack to test the method in this paper, which can protect the client from data poisoning attacks and model poisoning attacks.
3) **Evaluation Metrics**: Since the Trim attack aims to improve the test error rate, this paper uses the test error rate of the global model to evaluate the robustness, where the test error rate of the global model refers to the proportion of labels that the global model mispredicts. When the proposed method obtains a lower test error rate under this attack, the method is robust against this attack. This paper also uses the accuracy rate as one of the evaluation metrics of our method.
4) **System settings**: In this paper, the number of clients is set to 50, the proportion of malicious clients is 40%, the data of each client is randomly allocated, the data center is allocated 100 training data, and the distribution of data in the data center is the same as the distribution of the overall training data. The model in this paper is an intrusion detection model with 2000 federated learning iterations under the fully connected neural network. Unless otherwise stated, experiments were performed with this system setup. In order to test whether the proposed method can resist the poisoning attack and reasoning attack of federated learning in smart grid AMI intrusion detection, this paper uses a simple fully connected neural network, which can be replaced by other neural networks later.

TABLE 3 Test error rate.

Method	FedAvg	FLTrust	PEFL	Proposed method
No attack	0.1465	0.1602	0.1523	0.1494
Trim attack	0.2881	0.1611	0.1553	0.1523

6.3.2 Experimental results

1) **Robustness evaluation**: This paper compares with FLTrust Cao et al. [25], FedAvg McMahan et al. [26], PEFL Liu et al. [27] methods. Table 3 shows the test error rate when the distribution of the data center samples is the same as the distribution of the overall training samples. At this time, the number of clients is 50, the data of each client is randomly allocated, the data center allocates 100 training data, and the distribution of data in the data center is the same as that of the overall training data. When there is no attack, the proportion of malicious clients is 0%, and when there is an attack, the proportion of malicious clients is 40%.

Without the attack, the test error rate of the proposed method is lower, while FedAvg, FLTrust and PEFL have higher test error rate. It shows that this method has no negative impact on the federated aggregation process when there is no attack. For example, the test error rate of the proposed method is 0.1494, while the test error rates of FedAvg, FLTrust and PEFL are 0.1465, 0.1602 and 0.1523. The results show that the proposed method is more accurate than other methods against poisoning attacks in the absence of attacks. In the presence of Trim attack, the test error rate of the proposed method, FLTrust and PEFL is higher than that of FedAvg without attack, but the test error rate of the proposed method is the closest to FedAvg without attack, which indicates that the proposed method is robust. When there is an attack, the proposed method has a lower test error rate than other methods (FedAvg, FLTrust, PEFL) because our method integrates all client data information for detection. FedAvg has no defenses, and when there is a malicious attack, it may be

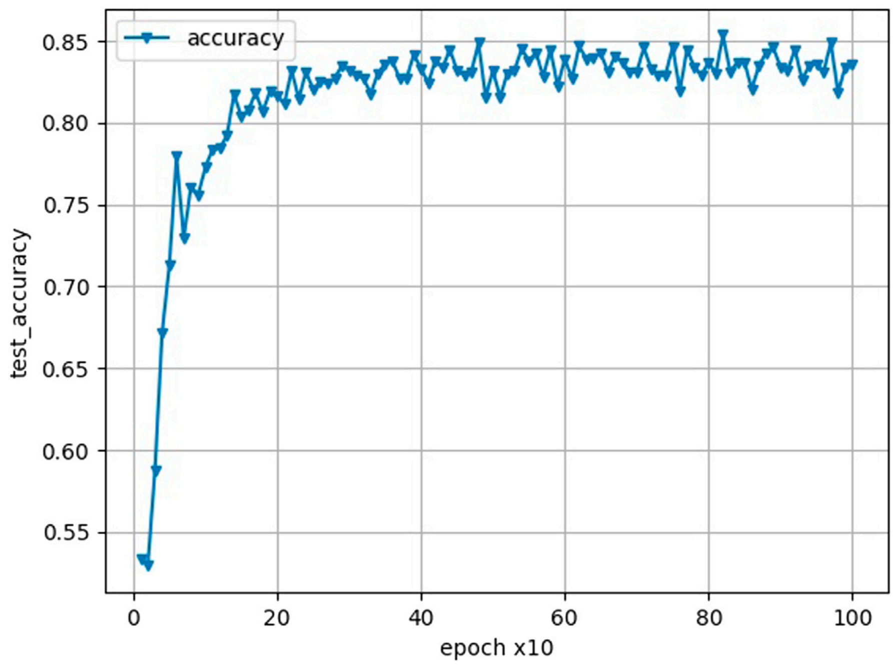

FIGURE 2
The effect of iteration on accuracy.

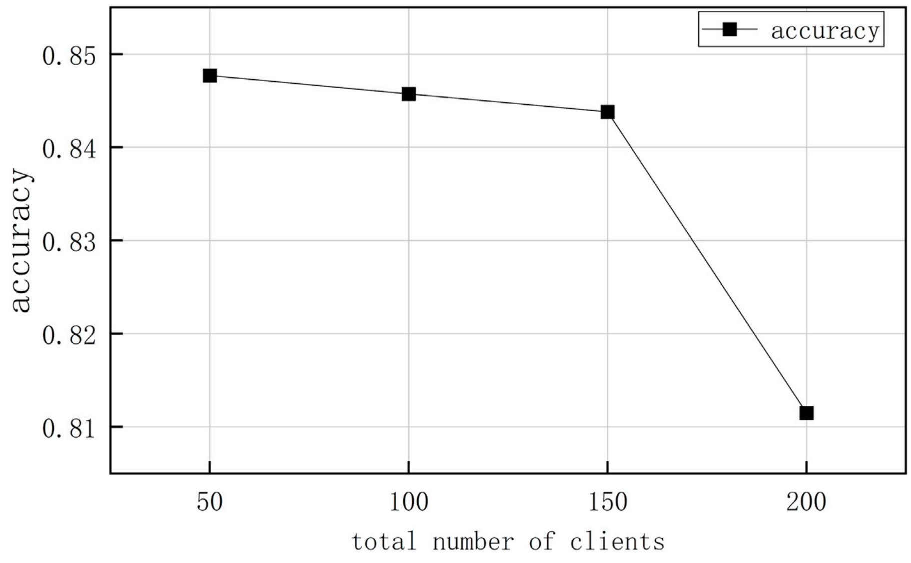

FIGURE 3
The effect of the total number of clients on the accuracy.

guided by the malicious attack model. FLTrust updates the model through the joint training of the data center model and the client model, and uses the pruning cosine method to eliminate the client models with a similarity less than zero, and retain all the client models greater than zero. So when the client model with low similarity and not less than zero participates in model aggregation, the global model will also be affected. The accuracy of PEFL is similar to the

TABLE 4 The effect of malicious client ratio on accuracy.

Malicious ratio (%)	0	10	20	40	60	80	90	100
FedAvg	0.8535	0.8271	0.7559	0.7119	0.6807	0.6299	0.4639	0.2490
FLTrust	0.8477	0.8467	0.8428	0.8447	0.5479	0.4648	0.4648	0.4596
PEFL	0.8398	0.8389	0.8359	0.8389	0.8389	0.8379	0.8135	0.2549
Proposed method	0.8506	0.8447	0.8438	0.8477	0.8467	0.8496	0.8350	0.2510

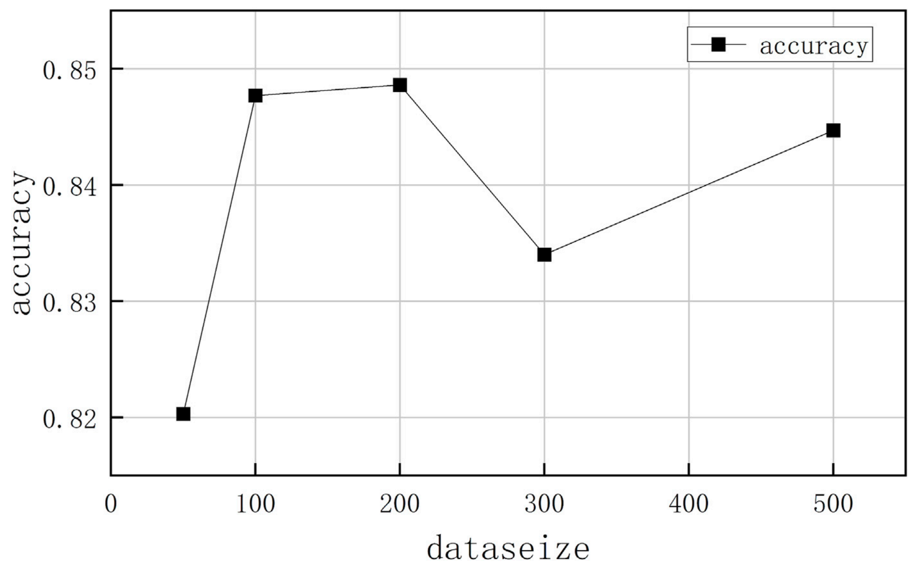

FIGURE 4
The impact of data center sample size on accuracy.

proposed method, because when the number of malicious attackers is less than 50% of the total number, the baseline model selected by PEFL according to the median dimension is highly likely to bypass the malicious model, and the Pearson correlation coefficient and A logarithmic function is used to generate aggregate weights, assigning larger weights to clients with high similarity and suppressing their weight values for smaller clients.

2) **Accuracy evaluation**: Many factors affect the size of the accuracy, s. ch as model type and structure, sample quality, number of federated learning iterations, total number of clients and proportion of malicious clients, and data center data volume and distribution (bias). This paper tests the effect of the number of federated learning iterations, the number of clients and the proportion of malicious clients, as well as the number and distribution of samples in the data center on the accuracy.

Figure 2 shows the change in the accuracy of the proposed method for 1,000 federated learning iterations when the number of clients is 50, the data of each client is randomly allocated, the data center allocates 100 training data, and the distribution of data in the data center is the same as that of the overall training data, and the proportion of malicious clients is 20%. When the number of federated learning iterations raises, the precision of the model improves. Before 250 federated learning iterations, the model's accuracy raises significantly, and after 400 federated learning iterations, the model's precision stabilizes.

Figure 3 shows the effect of the total number of clients on the model accuracy. At this time, the proportion of malicious clients is 40%, the data of each client is randomly allocated, and the data center allocates 100 training data, and the distribution of data in the data center is the same as that of the overall training data. As the number of clients raises, the precision of the model decreases. The total number of clients ranges from 50 to 200. When the total number of clients is 50, 100, 150, and 200, the intrusion detection rates of the proposed method are 0.8477, 0.8457, 0.8438, and 0.8115, respectively. Since the number of samples of the client is more, the training effect of the model is more friendly. Therefore, when the total number of clients increases, the

TABLE 5 The impact of data center sample distribution on accuracy.

deviation	$q = 0.2$	$q = 0.4$	$q = 0.6$	$q = 0.8$	$q = 1$
No attack	0.8411	0.8496	0.8389	0.4951	0.3574
Trim attack	0.8320	0.8154	0.8096	0.3818	0.3496

number of samples on each client decreases, which may have a certain impact on the performance of the model.

Table 4 shows the effect of different malicious client proportions on the model accuracy. At this time, the number of clients is 50, the data of each client is randomly allocated, the data center allocates 100 training data, and the distribution of data in the data center is the same as that of the overall training data. As the number of poisoned clients raises, the precision of all models decreases. This is because the amount of malicious data raises, the amount of normal data decreases, and there is too little correct information for global model training, resulting in a drop in accuracy. However, the method proposed in this paper can maintain a similar accuracy rate as FedAvg without attack when the proportion of malicious clients is 90%. Other methods, such as FedAvg and FLTrust, can only tolerate less than 60% of malicious clients. When the proportion of malicious clients is 20%, the FedAvg accuracy decreases from 0.8535 to 0.7559. When the proportion of malicious clients is 60%, the accuracy of FLTrust decreases from 0.8477 to 0.5479. This shows that the method proposed in this paper has better performance against Trim.

Figure 4 shows the effect of the number of data center samples on the accuracy of the intrusion detection model. At this time, the number of clients is 50, the proportion of malicious clients is 40%, the data of each client is randomly distributed, and the distribution of data in the data center is the same as that of the overall training data. Here, the number of data center samples is set to 50–500, corresponding to different accuracy rates. When the number of samples in the data center is 100, the accuracy rate reaches 84.77%, and then when the number of samples in the data center increases, the accuracy rate decreases slightly, indicating that the data center can verify the proposed method with 100 data samples, and obtain Good detection effect.

Table 5 describes the impact of the data distribution of the data center on the model accuracy. At this time, the number of clients is 50, and the proportion of malicious clients is 40%. The data of each client is randomly allocated, and the data center allocates 100 training data. The accuracy rate is the highest when the sample distribution deviation of the data center is 0.2, and the larger the deviation, the lower the accuracy rate. When $q = 0.2$, it means that the number of various samples in the data center is the same, and the more uniform the distribution of samples in the data center, the better the trained model can reflect the characteristics of various samples. The model trained in the data center is used as the benchmark model for eliminating malicious models. The more reliable the benchmark model is, the more accurate the malicious model can be eliminated.

7 Conclusion

This paper studies the possible inference attacks and poisoning attacks in the joint training of smart grid AMI using federated learning technology. Firstly, the reasons for inference attack and poisoning attack in smart grid AMI using federated learning technology are analyzed. Secondly, the threat model of smart grid AMI scenario when applying federated learning technology is proposed, and our defense target is proposed according to the threat model, and then the system model is proposed. Next, this paper proposes a federated learning intrusion detection method against inference attacks and poisoning attacks, which is achieved through three processes:

1) Client local training: This process is that each local client trains the model and encrypts the trained model using the encryption technology CKKS and then uploads it.
2) Secure aggregation: In the CKKS environment, the Pearson correlation coefficient is used to calculate the directional similarity between the model trained by the client and the model trained by the data center, and the scaled similarity value is used as the adaptive weight value of the server model aggregation. At the same time, the model gradient amplitudes uploaded by each client are normalized according to the data center model gradient amplitudes.
3) Model distribution: The data center distributes the aggregated global model to each client. The initial aggregated global model is encrypted with the public key of the control server. If it is directly distributed to each client, the client cannot access the specific model. Therefore, the global model is re-encrypted using the client's public key with the cooperation of the data center and the control server.

Finally, the optimization analysis of the algorithm, the safety calculation analysis and the performance evaluation of the proposed method are carried out. The research results show that the proposed method can effectively resist inference attacks and poisoning attacks, and the intrusion detection based on federated learning can maintain a good detection performance in the AMI network.

Data availability statement

Publicly available datasets were analyzed in this study. This data can be found here: https://www.unb.ca/cic/datasets/nsl.html.

Author contributions

Writing—original draft, JW; methodology and resources, JW and YC; funding acquisition and supervision, YC and ZX; software and visualization, ZX, CH, and FY. All authors have read and agreed to the published version of the manuscript.

Funding

This work was funded by National Natural Science Foundation of China (NSFC) 52177067, the Research Project on Teaching Reform of General Colleges and Universities in Hunan Province (Grant No. HNJG-2020-0261), China.

Acknowledgments

This is a short text to acknowledge the contributions of specific colleagues, institutions, or agencies that aided the efforts of the authors.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's Note

References

1. Li Y, Qiu R, Jing S. Intrusion detection system using online sequence extreme learning machine (os-elm) in advanced metering infrastructure of smart grid. *PloS one* (2018) 13:e0192216. doi:10.1371/journal.pone.0192216

2. Hasan MN, Toma RN, Nahid A-A, Islam MM, Kim J-M. Electricity theft detection in smart grid systems: A cnn-lstm based approach. *Energies* (2019) 12: 3310. doi:10.3390/en12173310

3. Su Z, Wang Y, Luan TH, Zhang N, Li F, Chen T, et al. Secure and efficient federated learning for smart grid with edge-cloud collaboration. *IEEE Trans Ind Inf* (2021) 18:1333–44. doi:10.1109/tii.2021.3095506

4. Yu F, Kong X, Mokbel AAM, Yao W, Cai S. Complex dynamics, hardware implementation and image encryption application of multiscroll memeristive hopfield neural network with a novel local active memeristor. *IEEE Trans Circuits Syst* (2022) 1. doi:10.1109/tcsii.2022.3218468

5. Shen H, Yu F, Wang C, Sun J, Cai S. Firing mechanism based on single memristive neuron and double memristive coupled neurons. *Nonlinear Dyn* (2022) 110:3807–22. doi:10.1007/s11071-022-07812-w

6. Yu F, Shen H, Yu Q, Kong X, Sharma PK, Cai S. Privacy protection of medical data based on multi-scroll memristive hopfield neural network. *IEEE Trans Netw Sci Eng* (2022) 1–14. doi:10.1109/tnse.2022.3223930

7. Wu C, Luo C, Xiong N, Zhang W, Kim T-H. A greedy deep learning method for medical disease analysis. *IEEE Access* (2018) 6:20021–30. doi:10.1109/access.2018.2823979

8. Kumar P, Kumar R, Srivastava G, Gupta GP, Tripathi R, Gadekallu TR, et al. Ppsf: A privacy-preserving and secure framework using blockchain-based machine-learning for iot-driven smart cities. *IEEE Trans Netw Sci Eng* (2021) 8:2326–41. doi:10.1109/tnse.2021.3089435

9. Liu X, Li H, Xu G, Lu R, He M. Adaptive privacy-preserving federated learning. *Peer-to-peer Netw Appl* (2020) 13:2356–66. doi:10.1007/s12083-019-00869-2

10. Bonawitz K, Ivanov V, Kreuter B, Marcedone A, McMahan HB, Patel S, et al. Practical secure aggregation for privacy-preserving machine learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, Texas, October 30–November 03, 2017 (2017). p. 1175–91.

11. Mohassel P, Zhang Y. Secureml: A system for scalable privacy-preserving machine learning. In: 2017 IEEE symposium on security and privacy (SP); 22-26 May 2017; San Jose, CA, USA. IEEE (2017). p. 19–38.

12. Riazi MS, Weinert C, Tkachenko O, Songhori EM, Schneider T, Koushanfar F. Chameleon: A hybrid secure computation framework for machine learning applications. In: Proceedings of the 2018 on Asia conference on computer and communications security, Incheon, South Korea, June 04–June 04, 2018 (2018). p. 707–21.

13. Phong LT, Aono Y, Hayashi T, Wang L, Moriai S Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Trans Inf Forensics Security* (2017) 13:1333–45.

14. Xu G, Li H, Zhang Y, Xu S, Ning J, Deng R. Privacy-preserving federated deep learning with irregular users. *IEEE Trans Dependable Secure Comput* (2020) 1. doi:10.1109/tdsc.2020.3005909

15. Fang H, Qian Q. Privacy preserving machine learning with homomorphic encryption and federated learning. *Future Internet* (2021) 13:94. doi:10.3390/fi13040094

16. Blanchard P, El Mhamdi EM, Guerraoui R, Stainer J. Machine learning with adversaries: Byzantine tolerant gradient descent. *Adv Neural Inf Process Syst* (2017) 30.

17. Xia Q, Tao Z, Hao Z, Li Q. Faba: An algorithm for fast aggregation against byzantine attacks in distributed neural networks. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence Main track, Macao, China, August 10–August 16, 2019 (2019). p. 4824–30. doi:10.24963/ijcai.2019/670

18. Yin D, Chen Y, Kannan R, Bartlett P. Byzantine-robust distributed learning: Towards optimal statistical rates. In: International Conference on Machine Learning (PMLR), Stockholm Sweden, July 10–July 15, 2018 (2018). p. 5650–9.

19. Chen Y, Su L, Xu J. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *Proc ACM Meas Anal Comput Syst* (2017) 1: 1–25. doi:10.1145/3154503

20. Fang M, Cao X, Jia J, Gong N. Local model poisoning attacks to {Byzantine – Robust} federated learning. In: 29th USENIX Security Symposium, Boston, United States, August 12–August 14, 2020. USENIX Security (2020). p. 1605–22.

21. Xie C, Koyejo S, Gupta I. Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance. In: International Conference on Machine Learning (PMLR), Long Beach, California, United States, June 9–June 15, 2019 (2019). p. 6893–901.

22. Zhao L, Hu S, Wang Q, Jiang J, Shen C, Luo X, et al. Shielding collaborative learning: Mitigating poisoning attacks through client-side detection. *IEEE Trans Dependable Secure Comput* (2020) 18:1–2041. doi:10.1109/tdsc.2020.2986205

23. Nguyen TD, Rieger P, Chen H, Yalame H, Möllering H, Fereidooni H, et al. {FLAME}: Taming backdoors in federated learning. In: 31st USENIX Security Symposium, Boston, MA, United States, August 10–August 12, 2022. USENIX Security (2022). p. 1415–32.

24. Cheon JH, Kim A, Kim M, Song Y. Homomorphic encryption for arithmetic of approximate numbers. In: International conference on the theory and application of cryptology and information security. Springer (2017). p. 409–37.

25. Cao X, Fang M, Liu J, Gong NZ. Fltrust: Byzantine-robust federated learning via trust bootstrapping. *arXiv* (2020) 591, [Preprint]. https://arxiv.org/abs/2012.13995.

26. McMahan B, Moore E, Ramage D, Hampson S, Arcas BAy. Communication-efficient learning of deep networks from decentralized data.// Artificial 593 intelligence and statistics. *PMLR* (2017): 1273–1282.

27. Liu X, Li H, Xu G, Chen Z, Huang X, Lu R. Privacy-enhanced federated learning against poisoning adversaries. *IEEE Trans Inform Forensic Secur* (2021) 16: 4574–88. doi:10.1109/tifs.2021.3108434