Hindawi
International Journal of Intelligent Systems
Volume 2023, Article ID 2956990, 22 pages
https://doi.org/10.1155/2023/2956990

Research Article

Efficient Privacy-Preserving Federated Deep Learning for Network Intrusion of Industrial IoT

Ningxin He,[1] **Zehui Zhang**,[1,2] **Xiaotian Wang**,[1] **and Tiegang Gao**[1]

[1]*College of Software, Nankai University, Tianjin 300071, China*
[2]*China-Austria Belt and Road Joint Laboratory on Artificial Intelligence and Advanced Manufacturing, Hangzhou Dianzi University, Xiasha Higher Education Zone, Hangzhou 310018, China*

Correspondence should be addressed to Tiegang Gao; gaotiegang@nankai.edu.cn

Received 16 September 2022; Revised 18 October 2023; Accepted 20 October 2023; Published 16 November 2023

Academic Editor: Alexander Hošovský

Intrusion detection systems play a very important role in industrial Internet network security. However, in the large-scale, complex, and heterogeneous industrial Internet of Things (IoT), it is becoming more and more difficult to defend network intrusion threats due to the insufficiency of high-quality attack samples. To solve the problem, an efficient federated network intrusion method called EFedID is proposed for industrial IoT, which can allow different industrial agents to collaboratively train a comprehensive detection model. Specifically, the adaptive gradient sparsification method is introduced to alleviate the communication and computation overheads. To protect the data privacy of the agents, a CKKS cryptosystem-based secure communication protocol is designed to encrypt the model parameters through the federated training process. Our proposed system demonstrates exceptional detection performance on the NSL-KDD, KDD CUP 99, and CICIDS 2017 datasets. Notably, on the NSL-KDD dataset, the model compression rate reaches 9 times while the model accuracy reaches 84.31%. On the KDD CUP 99 dataset, the model compression rate reaches 8.9 times while the model accuracy reaches 97.3%. Lastly, on the CICIDS 2017 dataset, the model compression rate reached 6.173 times while the model accuracy reached 95.51%. The experimental results demonstrate that the proposed method is very suitable for effectively developing a high-accuracy detection model while protecting the data information of industrial agents. Furthermore, the method can be extended to other recent deep learning networks for intrusion detection.

1. Introduction

Industrial Internet of Things (IoT) encapsulates communication technologies, edge computing, cloud servers, artificial intelligence (AI), and existing industrial control systems [1, 2]. It aims to connect real-world scenarios with distributed computing principles to realize smart manufacturing, resource management, and other processes. With the rapid development of industrial IoT, the network security threats to IoT are becoming more and more serious [3, 4] and cyber security has become a key issue for industrial IoT. Network intrusion refers to any unauthorized activity on a digital network, which is one of the most common threats in cyber space. It often involves stealing valuable network resources and jeopardizing the security of networks and/or their data. Hence, intrusion detection methods are proposed to monitor network operations in real time and detect suspicious invasions.

In recent years, to proactively detect and respond to network intrusions, researchers have used artificial intelligence (AI) technologies to design intrusion detection methods. Deep learning (DL) is predominant in the recent literature on network intrusion detection. Recent studies [5] have definitely demonstrated that DL techniques can achieve excellent detection accuracy compared to conventional machine learning techniques. Many DL-based models have been used for network intrusion detection. For instance, the paper [6] developed an autoencoder-based intrusion detection framework that harnesses the power of convolutional and recurrent neural networks to proactively identify cyber

threats in IIoT networks. The work placed a strong emphasis on model explainability, empowering security administrators to interpret the underlying data evidence and causal reasoning behind intrusion alerts. The framework applies a two-step sliding window (SW) to better learn the latent representations of the data features, effectively extracting features including malicious pattern contexts. Ismail et al. [7] investigated electricity theft attacks in smart grid cyber-physical systems and proposed a deep learning-based intrusion detection system. Wazzeh et al. [8] proposed a DTL-(deep transfer learning-) based residual neural network (ResNet) to effectively detect various network threats against the heterogeneous Internet of Things. More recent related works are presented in Section 2.1.

Training high-performance deep learning models depends on a large number of high-quality data. Currently, most of the existing DL-based intrusion detection methods assume that developers have sufficient high-quality cyberattack data. However, it is difficult to achieve this assumption in practical scenarios because it is usually very difficult and time-consuming for one industrial IoT owner to collect a large number of cyberattack samples. In addition, the traditional centralized learning approach (CL), which centralizes distributed owners' data to a central server for model building, is also difficult to implement because industrial IoT owners are usually unwilling to share their attack samples with third parties out of security, privacy, and business interest considerations [8–11]. The open troublesome problem of insufficient training samples has become a major obstacle in training high-quality intrusion detection models. Therefore, how to solve this difficulty and develop accurate and efficient intrusion detection methods becomes a challenge for protecting intelligent networks in practical applications.

The application of federated learning to solve the insufficient data problem is a relatively new research area [9, 12–14]. Since federated learning can organize different participants to collaboratively train a comprehensive model, it has great potential to utilize more data from different participants and get better model performance [12, 15, 16]. There are also some federated learning-based intrusion detection methods [17, 18] that have achieved good performance, but traditional federated learning requires a large number of parameters to be transmitted during the training process, which may not be applicable to resource-constrained industrial environments. Moreover, it is very possible to be attacked by an adversary during the parameter transmission process, which can cause data privacy leakage. In this paper, we first develop a new method for multiple industrial IoT owners to cooperatively build a comprehensive intrusion detection model to alleviate the problem of insufficient high-quality attack samples while preserving their local data. In addition, to be more applicable to industrial IoT environments with resource-constrained devices, we introduce adaptive gradient sparsification technology, which sends sparse vectors of the model parameters, to alleviate the cryptographic computing and communication overheads. Considering the security of the model parameters in the data transmission process, we design a secure communication protocol based on the CKKS cryptosystem. Compared to other traditional homomorphic encryption schemes, the CKKS scheme has a faster encryption/decryption speed and supports both additive and multiplicative homomorphic encryption. The main contributions of this study can be concluded as follows.

First, we present a network intrusion method named EFedID, which (1) relieves the open troublesome problem of insufficient training samples and (2) supports data preprocessing at each industrial agent and preserves their local data information.

Second, an adaptive gradient sparsification method, named AGS, is developed to alleviate the resource overheads of the EFedID system while retaining high efficiency so that it can be deployed to a large number of resource-constrained devices.

Third, the CKKS cryptosystem-based secure communication protocol is designed for the federated learning system, by which the security and privacy of model parameters through the training process can be well preserved.

We present related works in Section 2 and describe the design of EFedID in Section 3. In Section 4, we analyze the security and functionality of our method. In Section 5, we give comparison experiments to verify the proposed method's performance. Finally, we conclude this study in Section 6. Table 1 shows a summary of the acronyms used in this paper.

2. Related Work

2.1. Intrusion Detection Schemes for Industrial CPSs (Cyber-Physical Systems). To fight against cyberattacks, various intrusion detection methods have been proposed. Wang et al. [19] proposed an intrusion detection framework method based on SVM with feature augmentation. However, traditional machine learning-based detection methods are not suitable for massive and high-dimension network traffic data detection. In recent years, owing to the rapid development of deep learning (DL) technologies, many researchers proposed DL-based intrusion detection methods. For instance, Li et al. [20] introduced a convolutional neural network (CNN) to design a network intrusion detection model for industrial IoT, which achieves high accuracy on the NSL-KDD dataset. However, it has been experimented on only one dataset. In practice, the performance of the classifier may fluctuate due to some redundant or inefficient features in different datasets. To alleviate this problem, Wu and Li [21] proposed some feature selection methods and introduced a combination of neural networks and random forests to improve the detection performance. Compared to similar methods, their approach provides better results in general by identifying important and closely related features. However, this scheme requires features to be extracted from existing training data samples and it lacks generalization. The paper [22] proposed a real-time industrial IoT intrusion detection system based on deep autoencoders. This system utilizes a statistical feature mining approach to extract relevant features from network traffic data, which is designed to be helpful in improving the model's generalization and

Table 1: Summary of acronyms used.

Acronyms	Full name
IoT	Internet of Things
AI	Artificial intelligence
CL	Centralized learning
DL	Deep learning
FL	Federated learning
ANN	Artificial neural network
DNN	Deep neural network
CNN	Convolutional neural network
Multi-CNN	Multi-convolutional neural network
AGS	Adaptive gradient sparsification method
SVM	Support vector machine
HMM	Hidden Markov model
BBFO	Binary bacterial foraging optimization
HFL	Hierarchical federated learning
CPS	Cyber-physical system
KGC	Key generation center
MGD	Momentum gradient descent
MLP	Multilayer perceptron
ReLU	Rectified linear unit
CPA	Chosen-plaintext attack
TLS/SSL	Transport Layer Security/Secure Sockets Layer
CR	Compression rate
IID	Identically and independently distributed
Non-IID	Not identically and independently distributed

addressing the issues of low detection rates and high false positive rates (FPRs). Nevertheless, it is difficult to obtain enough high-quality data samples for detection model training in practical scenarios. Moreover, due to the sensitivity, privacy, and high value of industrial IoT data, data owners are usually reluctant to share data. Tang et al. [17] proposed a network intrusion detection method based on federated learning. Although this scheme alleviates the problem of insufficient attack samples, it does not consider the FL system resource consumption problem, which does not apply to resource-constrained devices. Khan et al. [18] proposed the federated-SRU IDS model, which employs the improved simple recurrent unit architecture to reduce computational cost and mitigate the gradient vanishing problem in recurrent networks for enhancing the model intrusion detection performance. Moreover, the system facilitates model aggregation through multiple communication rounds within the federated learning architecture, allowing multiple ICS networks and stakeholders to collaboratively build comprehensive IDS models while preserving their data privacy. However, they did not consider the problem that the model parameters may be attacked by an adversary during transmission which is not able to protect the privacy and security of the local data.

2.2. Federated Learning-Based Industrial Applications. Owing to its outstanding performance, many researchers proposed various FL-based industrial applications. For instance, Zhang et al. [23] proposed a rolling bearing fault diagnosis method based on federated learning and convolutional neural network. Lu et al. [24] proposed a federated learning scheme for the digital twin networks by incorporating IoT technologies, which improves communication efficiency and reduces the transmission energy cost. Zhang et al. [25] proposed a dynamic fusion-based FL for medical diagnosis to classify COVID-19 infections, which can adaptively determine the participants according to their local model performance and model aggregation scheme based on participants' training time. In 2022, Aloqaily et al. [26] proposed a hierarchical federated learning (HFL) solution based on blockchain, which can provide fast, safe, and accurate decision making for industrial machines.

3. Our Approach

In this section, we introduce our proposed EFedID, which combines our designed adaptive gradient sparsification (AGS) method and the CKKS cryptosystem-based secure communication protocol. We first describe the system model and then present the detailed operations within the method.

3.1. System Model. Federated learning offers a solution to the challenge of insufficient data by facilitating collaborative model training among multiple institutions, all without the need to disclose their individual data to each other or to a central server. In this process, instead of transmitting raw data, each agent sends model parameters to a cloud server. We use the generic setting for the federated learning system, where a cloud server and K industrial agents collaboratively train a model for intrusion detection. As shown in Figure 1, the system is composed of three parties: (1) a key generation center (KGC), (2) a cloud server, and (3) K industrial agents. These parties in the FL system are described as follows:

(1) KGC: The KGC is a trusted third-party organization, which is responsible for generating the keys based on the CKKS cryptosystem and distributing the keys to industrial agents. KGC does not send keys to entities outside the system or without access.

(2) Cloud server: the cloud server contains three functions: (a) it establishes different communication channels for the industrial agents, (b) it collects the trained models from the industrial agents and then aggregates the models to obtain a new global model, and (c) it adjusts the sparsification rate φ.

(3) Industrial agents: Each industrial agent collects and stores the raw data. They are responsible for training a model locally and uploading the trained model to the cloud server until the end of training. In our system, each agent sparsifies the model parameters before sending them.

3.2. Adversary Model. We assume that the cloud server and all the industrial agents are honest-but-curious entities. Honest-but-curious entity means that it will faithfully follow the designed protocol and not tamper with the calculation results but will attempt to infer private information from the input of other entities in the scheme (industrial agents in this article). Meanwhile, the KGC is considered a trusted third

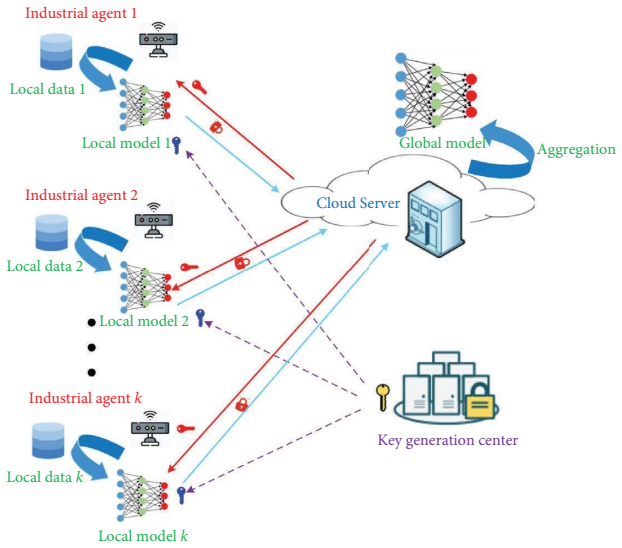

Figure 1: Federated learning system architecture.

party, and it will only distribute the keys to the industrial agents in the system.

3.3. The Workflow of the Proposed Method. The workflow of our EFedID includes four stages (see also Figure 2 and Algorithm 1).

3.3.1. System Setup. As mentioned in Section 3.1, our system consists of three types of entities: the KGC, the cloud server, and the industrial agents. During the setup phase, they have different tasks. The KGC conducts KeyGen (λ) (see more details in Section 3.6) to produce the public key pk and private key based on the CKKS cryptosystem and sends the keys to all agents. At the same time, the cloud server establishes different secure channels between the cloud server and each industrial agent to protect transmission data. Then, each industrial agent uploads the encrypted initial model to the cloud server. The cloud server aggregates the encrypted initial models uploaded by all industrial agents to generate the initial global model and broadcasts the initial global model to all industrial agents.

3.3.2. Each Industrial Agent Trains the Local Model. After receiving the encryption sparse global model parameters Enc(sparse ($\mathbf{v}_{\text{glo}}$)) and the sparsification rate φ from the cloud server, each industrial agent trains a local DL-based intrusion detection model, using its private data resource D_k. Since the sparse global model parameters are encrypted, each agent needs to decrypt using the public key pk.

$$\text{sparse}\left(\mathbf{v}_{\text{glo}}\right) = \text{CkkDec}\left(\text{Enc}\left(\text{sparse}\left(\mathbf{v}_{\text{glo}}\right)\right)\right), \tag{1}$$

where CkkDec(•) denotes the decryption operation (see more details in Section 3.6).

After obtaining the sparse model parameters sparse ($\mathbf{v}_{\text{glo}}$), each industrial agent updates the local model.

$$\mathbf{w}_{\text{a},k} = \mathbf{w}_{\text{a},k} - \text{spares}\left(\mathbf{v}_{\text{glo}}\right), \tag{2}$$

where $\mathbf{w}_{\text{a},k}$ denotes the local model parameters of the agent k.

Then, each agent uses the local data to train the local model, which can be expressed as

$$\mathbf{w}_{a,k} = \text{MGD}\left(\mathbf{w}_{a,\text{com}}, D_{\text{batch},k}\right), \tag{3}$$

where MGD denotes the momentum gradient descent algorithm and $D_{\text{batch},k}$ denotes the mini-batch data of the agent k.

3.3.3. Each Industrial Agent Uploads Model Parameters. Each industrial agent processes the local model parameters according to the sparsification rate φ downloaded from the cloud server to obtain sparse ($\mathbf{v}_{\text{a},k}$) (see more details in Section 3.4).

After sparse ($\mathbf{v}_{\text{a},k}$) is obtained, each agent encrypts the local sparse model parameters and uploads the encrypted parameters to the cloud server. The encryption formula is shown in the following equation:

$$\text{Enc}\left(\text{sparse}\left(\mathbf{v}_{a,k}\right)\right) = \text{CkkEnc}\left(\text{sparse}\left(\mathbf{v}_{a,k}\right)\right), \tag{4}$$

where CkkEnc(•) denotes the encryption operation (see more details in Section 3.6).

3.3.4. Cloud Server Updates Global Parameters. The server receives the encrypted sparsification local parameters Enc (sparse ($\mathbf{v}_{\text{a},k}$)) from all the industrial agents and computes the average model parameters Enc (sparse ($\mathbf{v}_{\text{glo}}$)) to update the global model parameters. Encrypted sparsification local parameters Enc (sparse ($\mathbf{v}_{\text{a},k}$)) can be aggregated without decryption because the CKKS encryption algorithm is homomorphic. The server uses the weighted federated averaging method, which assigns varying weights to individual agents based on the proportion of data each agent holds relative to the total dataset size. The aggregation formula is shown in the following equation:

$$\text{Enc}\left(\text{sparse}\left(\mathbf{v}_{\text{glo}}\right)\right) = \sum_{k=1}^{K}\left(\vartheta_k \cdot \text{Enc}\left(\text{sparse}\left(\mathbf{v}_{a,k}\right)\right)\right), \tag{5}$$

where ϑ_k denotes data contribution ratios calculated by $\vartheta_k = |D_k|/|D_{\text{all}}|$, $|D_k|$ denotes the number of the data of the agent k, and $|D_{\text{all}}|$ denotes the number of all agent's data. The global sparsification model parameters are obtained by adding all encrypted sparsification model parameters according to the data contribution ratios.

Then, the cloud server uses the AGS to make some adjustments to φ (see more details in Section 3.4).

Finally, the cloud server broadcasts the global model parameters Enc (sparse ($\mathbf{v}_{\text{glo}}$)) and adjusts φ to all agents to start the next round of federated training until the end. This

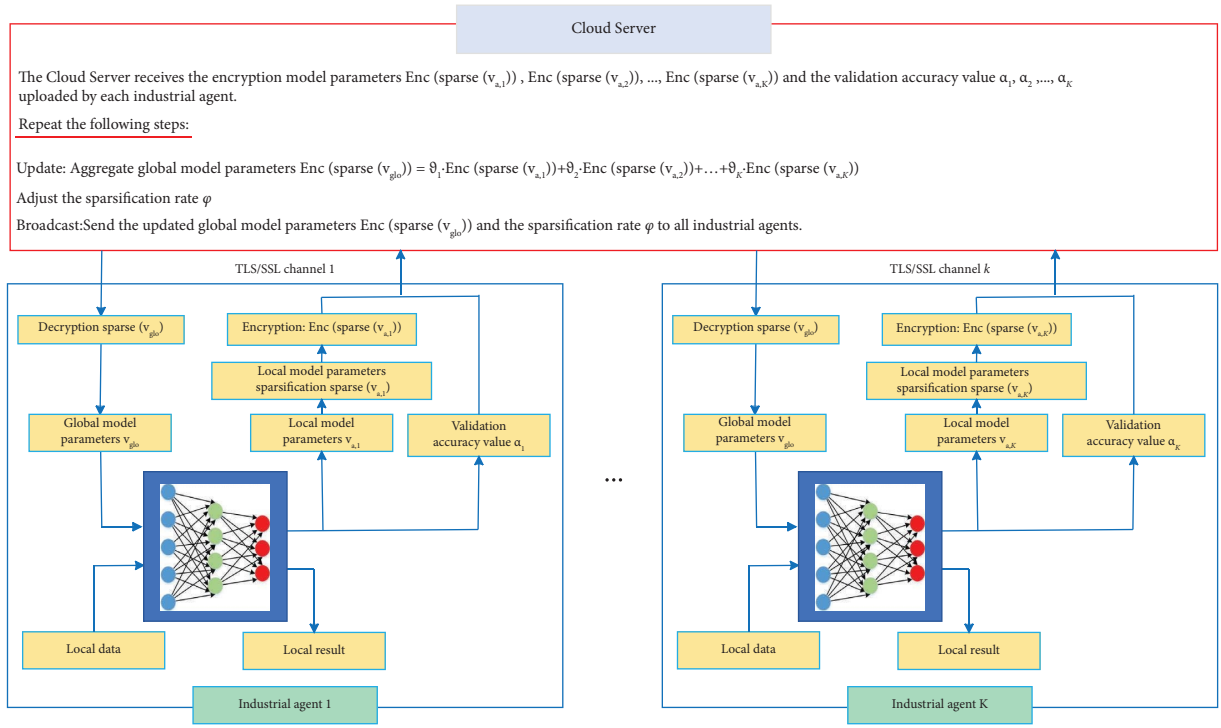

Figure 2: The workflow of the EFedID.

Input: Model, training parameters
Output: $\mathbf{w}_{glo}$
(1) **Initialization**:
(2) (a). Create separate communication channels between industrial agents and the cloud server;
(3) (b). Initialize the global model $\mathbf{w}_{glo}$;
(4) **(I). Industrial agent side:**
(5) **for industrial agent** $k = 1, \ldots, K$ **do**://**Parallel running**
(6) Receive Enc (sparse ($\mathbf{v}_{glo}$)) and φ;
(7) sparse ($\mathbf{v}_{glo}$) $\longleftarrow$ CKKDec (Enc (sparse ($\mathbf{v}_{glo}$)));
(8) $\mathbf{w}_{a,k} \longleftarrow \mathbf{w}_{a,k}$ − sparse ($\mathbf{v}_{glo}$);
(9) Compute the validation accuracy valueα_k;
(10) Train the model locally and obtain the model parameters $\mathbf{v}_k$;
(11) Perform sparse operation on model parameters to obtain sparse($\mathbf{v}_{a,k}$);
(12) Enc (sparse ($\mathbf{v}_{a,k}$)) $\longleftarrow$ CKKEnc (sparse ($\mathbf{v}_{a,k}$));
(13) Send Enc (sparse ($\mathbf{v}_k$)) andα_k to the server;
(14) **End**
(15) **(II). Cloud server side**:
(16) Receive Enc(sparse($\mathbf{v}_{a,k}$)) and α_k;
(17) Enc (sparse ($\mathbf{v}_{glo}$)) $\longleftarrow \sum_{k=1}^{K} (\vartheta_k \cdot$ Enc (sparse ($\mathbf{v}_{a,k}$)));
(18) Use our AGS algorithm to make some adjustments to φ;
(19) Broadcast Enc (sparse ($\mathbf{v}_{glo}$)) and φ to all agents;
(20) **Return** $\mathbf{w}_{glo}$.

Algorithm 1: EFedID.

procedure of multiple local update steps followed by global aggregation repeats until training convergence.

3.4. Adaptive Gradient Sparsification. The FL system communication process requires the transfer of a large number of parameters, which can put a huge strain on industrial IoT environments with limited network resources. In order to protect the data privacy of industrial agents, homomorphic encryption technology is usually used in the FL system. Encrypting and decrypting model parameters requires a lot of resources, especially when there are many agents in the system, which is often difficult on resource-constrained devices. To reduce the consumption of computing

resources and improve communication efficiency, a gradient sparsification method is introduced in our scheme to reduce the number of parameters for communication transmission.

In this paper, the momentum gradient descent (MGD) optimizer [27] is used to minimize $L(b_k, \mathbf{w}_t^{a,k})$. The optimization objective is formulated as shown in the following equation:

$$\min_{\mathbf{w}_{\mathrm{FL}}} \mathrm{E}_{(\mathbf{x},\mathbf{y})\sim D_{a,k}} L\left[f\left(x; \mathbf{w}_{\mathrm{FL}}\right), y\right], k = 1, \ldots, K, \tag{6}$$

where the local data of industry agent k are denoted by $D_{a,k}$.

At every iteration t, industrial agent k computes the loss $L(b_k, \mathbf{w}_t^{a,k})$ and the model parameters $\mathbf{v}_t^{a,k}$ with regard to $\mathbf{w}_t^{a,k}$.

$$\mathbf{v}_{t+1}^{a,k} = \gamma \mathbf{v}_t^{a,k} + \frac{1}{b_k} \sum_{i=1}^{b_k} \nabla_{\mathrm{w}_t^{a,k}} L\left(f\left(x_i; \mathbf{w}_t^{a,k}\right), \mathrm{y}_i\right). \tag{7}$$

For $t = 1, 2, 3, \ldots, N$, where $\mathbf{w}_t^{a,k}$ is the weight vector obtained at the end of the current iteration t, $L(f(x_i; \mathbf{w}_t^{a,k}), y_i))$ is the loss obtained at the end of the previous iteration $t(t = 0$ corresponds to model initialization), and $\nabla_{\mathbf{w}_t^{a,k}} L(f(x_i; \mathbf{w}_t^{a,k}), y_i) \in \mathbb{R}^{\mathrm{T}}$ is the sparse gradient of the local loss in iteration t with T defined as the dimension of the weight vector.

Formally, in every iteration t, the weight vector of the industrial agent k is adjusted by MGD as follows:

$$\mathbf{w}_{t+1}^{a,k} = \mathbf{w}_t^{a,k} - \eta \mathbf{v}_t^{a,k}. \tag{8}$$

When all industrial agents upload model parameters to the server, the global model parameters are computed as

$$\mathbf{v}_{\mathrm{glo}} = \frac{1}{K} \sum_{k=1}^{K} \mathbf{v}_{a,k}. \tag{9}$$

In every iteration t, the global model weight vector is optimized by MGD by

$$\mathbf{w}_{t+1}^{\mathrm{glo}} = \mathbf{w}_t^{\mathrm{glo}} - \mathbf{v}_t^{\mathrm{glo}}. \tag{10}$$

The main goal of gradient sparsification is to exchange only a small number of important gradients. The cloud server calculates the sparse global gradients according to these gradients and sends the updated sparse global gradients to each agent. The gradient sparsification scheme is very effective in reducing the computation and communication costs in the FL system. Han et al. [28] provided theoretical analysis to prove that local and global models can still converge after gradient sparse. Setting the sparsification rate in the gradient sparsification scheme is critical and requires a trade-off between model performance and resource savings. A high sparsification rate can significantly reduce the resource overhead, but it can also significantly degrade model performance. A low sparsification rate guarantees a limited loss of model performance but saves very few computing and communication resources. To better balance model performance and resource consumption, we adopt the adaptive gradient sparsification method referring to the adaptive learning rate method [15]. We consider a slightly different procedure in which instead of using a fixed compression ratio, we adaptively adjust the compression rate according to the training information of model performance. We will see in the experiments in Section 5.3 that our AGS performs better than the fixed compression ratio approach (GS).

In AGS, we use the accuracy value as the model performance metric. Industrial agent k computes the local accuracy value denoted by α_k. The local accuracy values α_k are sent from each industrial agent to the cloud server, and the server calculates the average of the accuracy values, which can be expressed as

$$\alpha_{\mathrm{glo}} = \sum_{k=1}^{K} \left(\vartheta_k \cdot \alpha_k\right). \tag{11}$$

We adjust the sparsification rate φ when the value of α_{glo} is continuously below the highest accuracy τ times. The adjustment formula is as follows:

$$\varphi = \max\left(\varphi - d_{\mathrm{rate}}, 0\right), \tag{12}$$

where d_{rate} denotes the decay rate. We set the value of d_{rate} to 0.001 in this paper based on experimentation. The τ value affects the sensitivity of the proposed scheme. We set the value of τ to 2 in this paper based on experimental.

The cloud server sends the sparsification ratio φ to all industrial agents. After the agents receive φ, each agent calculates the absolute value $\mathrm{abs}(\mathbf{v}_t^{a,k})$ of the local model parameters $\mathbf{v}_t^{a,k}$ and then sorts $\mathrm{abs}(\mathbf{v}_t^{a,k})$ of the local model from smallest to largest values, setting the value at φ% position as the local sparsity threshold θ for the t-th iteration.

$$\mathrm{abs}\left(\mathbf{v}_t^{a,k}\right) = \left|\mathbf{v}_t^{a,k}\right|. \tag{13}$$

Each industrial agent updates model parameters $\mathbf{v}_t^{a,k}$ whose absolute value is evaluated to exceed θ instead of all model parameters.

$$\mathrm{sparse}\left(\mathbf{v}_t^{a,k}\right) = \mathrm{sparse}\left(\gamma \mathbf{v}_{t-1}^{a,k} - \eta \frac{1}{b_k} \sum_{i=1}^{b_k} \nabla_{\mathbf{w}_{t-1}^{a,k}} L\left(f\left(x_i; \mathbf{w}_{t-1}^{a,k}\right), \mathrm{y}_i\right)\right) = \mathbf{v}_t^{a,k} \ominus \left[\mathrm{abs}\left(\mathbf{v}_t^{a,k} >= \theta\right)\right], \tag{14}$$

where sparse($\mathbf{v}_t^{a,k}$) denotes the sparse model parameters. $\boldsymbol{\Theta}[\cdot]$ denotes the identity function that is equal to $\mathbf{v}_t^{a,k}$ if the condition is satisfied and zero otherwise.

The rate of agent k can be computed as

$$\varphi_t = \frac{\left|\text{sparse}\left(\mathbf{v}_t^{a,k}\right)\right|}{\left|\mathbf{v}_t^{a,k}\right|}, \tag{15}$$

where the total number of sparse ($\mathbf{v}^{a,k}$) is denoted by |sparse ($\mathbf{v}^{a,k}$)| and the total number of $\mathbf{v}^{a,k}$ is denoted by $|\mathbf{v}^{a,k}|$.

Finally, the cloud server receives the encrypted sparse model parameters Enc (sparse ($\mathbf{v}_{a,k}$)) and computes the sparse global model parameters Enc (sparse ($\mathbf{v}_{\text{glo}}$)) according to equation (5).

The overall process is shown in Algorithm 2. Next, we analyze the resource consumption of the AGS algorithm. U_A denotes the number of agents, the total number of model parameters is denoted by M, and each model parameter takes up 4 bits.

3.4.1. Communication Cost. According to our algorithm, the communication cost mainly arises from data transmission. At each training, industrial agents send sparsification local models and local accuracy information to the cloud server, which incurs a communication cost of $[(1-\varphi)\ M+1] * 4\ U_A$ bits. The cloud server then returns the aggregated global model and the updated sparsification rate φ, which is of size $4(M+1)$ bits. Thus, the communication cost is about $4[U_A M(1-\varphi)+M+U_A+1]$ bits per training round.

3.4.2. Computational Cost. Before uploading the local model, the industrial agent needs to perform a sparsification operation on the local model parameters. The time complexity of obtaining the absolute value of the local model parameters $\mathbf{v}_t^{a,k}$ is $O(T)$ and T is the dimension of the local model parameters. The time complexity of computing the sparsification threshold is $O(T\log T)$. The time complexity of selecting the updated model parameters is $O(T)$. The total time complexity of the AGS algorithm is $O(T\ \log T+2T)$.

3.5. The CNN-Based Intrusion Detection Models. Now, we describe our designed intrusion detection model in detail.

3.5.1. Model Structure. Figure 3 shows the CNN-based intrusion detection model structure, which consists of a CNN module, a MLP (multilayer perceptron) module, and a softmax layer. The structure of the input data is resized to $5 * 5$ square, as expected by the CNN model. The CNN module contains two convolutional blocks, and each convolutional block contains one convolutional layer, one batch normalization layer, and one max-pooling layer. The activation function of each hidden layer is the rectified linear unit (ReLU). Cblock1 in the CNN module extracts $5 * 5$ feature map as the input of Cblock2. Cblock2 extracts $5 * 5$ feature map with 16 channels and inputs it into the MLP module. The MLP module is used to predict classes, which contain two fully connected layers and one dropout layer. The dropout layer with a dropout rate of 0.5 is adapted to control overfitting. Finally, we use the softmax layer that transforms the MLP output from exponential to probabilistic form. The cross-entropy function is used as the loss function, and the formula is as follows:

$$L = \frac{1}{B}\sum_{i=1}^{B}\sum_{j=0}^{J} y_{i,j}\log \widehat{y}_{i,j}, \tag{16}$$

where $y_{i,j}$ represents the true label of i-th sample, $\widehat{y}_{i,j}$ represents the probability label of i-th sample classified by softmax layer, J denotes the number of categories, and B denotes the size of the batch data.

3.5.2. Model Training. Model training is performed on the industrial agent side. At round t, local models of all industrial agents are initialized to the global model $\mathbf{w}_t^{\text{glo}}$. Then, each industrial agent trains our designed intrusion detection model locally on their private data resource D_k. $\mathbf{w}_{t+1}^{a,k}$ is updated locally based on MGD optimizer, which can adjust the weights of the model to reduce the cross-entropy function value.

3.6. CKKS Cryptosystem-Based Secure Communication Protocol. Now, we introduce our designed secure communication protocol based on CKKS [29]. Compared with Paillier homomorphic encryption algorithm, CKKS has great advantages in computing speed [30, 31], which is very helpful in improving the operation speed of federated learning. It is noticeable that TLS/SSL is used in our protocol to create secure communication channels, which helps reduce the risk of potential external adversary attacks during parameter transmission between the cloud server and industrial agents. Our CKKS cryptosystem-based secure communication protocol contains a total of four functions: *KeyGen*, *Ckk*Enc, *Ckk*Agg, and *CkkDec*.. The detailed algorithms are as follows:

(1) *KeyGen* (λ): The *key generation center* generates the public key and the private key by executing *KeyGen* (λ). The key generation center is given the security parameter λ. It selects a prime p and an integer q_0, L, τ, sets $q_l = p^l \bullet q_0$, where $l = 1, 2, \ldots, L$. The parameter $N = N(\lambda, q_L)$ and B-bound error distribution $\chi = \chi(\lambda, q_L)$ selected reasonably as parameters. Next, a random number $s = \text{HWT}(h)$ is chosen to generate the security key $\text{sk} \longrightarrow (1, s) \in Z_{q_L}^{N+1}$, where HWT ($h$) is a signed set of n-dimensional $\{1, 0, 1\}^N$ vector with Hamming weight h. In addition, a random number $A \longrightarrow Z_{q_L}^{N*\tau}$, $e \longrightarrow \chi^\tau$ are selected to generate the public key $\text{pk} = (-\text{As} + e(\text{mod}\, q_L), A) \in Z_{q_L}^{\tau*(N+1)}$. Finally, the key pairs are distributed to industrial agents by the *key generation center*.

(2) *CkkEnc* (sparse ($\mathbf{v}_{a,k}$), pk): The *CkkEnc* function is executed by the industrial agents to encrypt the local model parameters before uploading them to the cloud server. Randomly select a vector $r \leftarrow \{0, 1\}^\tau$ and obtain the ciphertext of the local model parameters *CkkEnc*(sparse ($\mathbf{v}_{a,k}$)). It is formulated as

Input: ϕ_0, η, γ
Initialize $\alpha_0 \longleftarrow 0$, $\mu \longleftarrow 0$
(1) for $t = 1, \ldots, N$ **do:**
(2) Each industrial agent $k = 1, \ldots, K$:
(3) Receive Enc (sparse ($\mathbf{v}_{\text{global}}$)) and ϕ
(4) sparse ($\mathbf{v}_{\text{glo}}$) $\longleftarrow$ CKKDec (Enc (sparse ($\mathbf{v}_{\text{glo}}$)))
(5) $\mathbf{w}_{a,k} \longleftarrow \mathbf{w}_{a,k} - \eta$sparse ($\mathbf{v}_{\text{glo}}$)
(6) Train the local model
(7) $\mathbf{v}_{t+1}^{a,k} \longleftarrow \gamma \mathbf{v}_{t+1}^{a,k} + \sum_{i=1}^{b_k} \nabla_{w_t^{a,k}} L(f(x_i; w_t^{a,k})), y_i$
(8) abs ($\mathbf{v}_t^{a,k}$) = $|\mathbf{v}_t^{a,k}|$
(9) Spare ($\mathbf{v}_t^{a,k}$) $\longleftarrow \mathbf{v}_t^{a,k} \ominus [\text{abs}(\mathbf{v}_t^{a,k}) >= \theta]$
(10) Enc (sparse ($\mathbf{v}_{a,k}$)) $\longleftarrow$ CKKEnc (sparse ($\mathbf{v}_{a,k}$))
(11) Compute the validation accuracy value α_k
(12) Send α_k and Enc (sparse ($\mathbf{v}_{a,k}$)) to cloud server
(13) The cloud server:
(14) sparse ($\mathbf{v}_t^{\text{glo}}$) $\longleftarrow 1/b_k \sum_{k=1}^{K}$ sparse ($\mathbf{v}_t^{a,k}$)
(15) $\alpha_{\text{glo}} \longleftarrow \sum_{k=1}^{K} (\delta_k \cdot \alpha_k)$
(16) If $\alpha_{\text{glo}} > \alpha_0$:
(17) $\alpha_0 \longleftarrow \alpha_{\text{glo}}$
(18) else:
(19) $\mu \longleftarrow \mu + 1$
(20) if $\mu >= \tau$:
(21) $\varphi \longleftarrow$ max $(\varphi - 0.001, 0)$
(22) $\mu \longleftarrow 0$
(23) Send ϕ and sparse ($\mathbf{v}_t^{\text{glo}}$)

Algorithm 2: AGS.

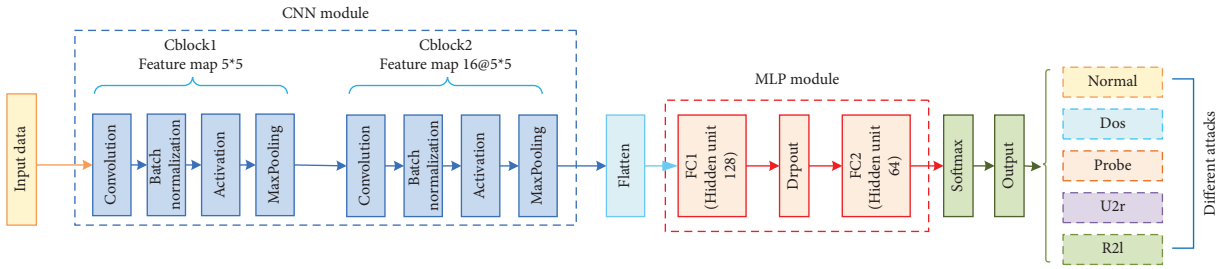

Figure 3: The structure of the proposed CNN-based intrusion detection model.

$$\text{CkkEnc}\left(\text{sparse}\left(\mathbf{v}_{a,k}\right)\right) \leftarrow \left(\mathbf{v}^{\text{sparse}\left(\mathbf{v}_{a,k}\right)}, 0\right) + pk^T \cdot r \in Z_{q_L}^{N+1}. \tag{17}$$

(3) *CkkAgg*(*CkkEnc*(*sparse*($\mathbf{v}_{a,k}$)), ..., *CkkEnc* (*sparse* ($\mathbf{v}_{a,K}$))): In each communication round k, once all the industrial agents have uploaded their encrypted parameters δk to the cloud server, the server executes the *CkkAgg* function to aggregate the local models and obtain the global model parameters. It is formulated as

$$\text{CkkEnc}\left(\text{sparse}\left(\mathbf{v}_{\text{glo}}\right)\right) = \sum_{k=1}^{K} \text{CkkEnc}\left(\text{sparse}\left(\mathbf{v}_{a,k}\right)\right)^{\delta_k}. \tag{18}$$

(4) *CkkDec* (*CkkEnc* (*sparse* ($\mathbf{v}_{\text{glo}}$)), *sk*): the *CkkDec* function is executed by the industrial agent to decrypt the global model parameters *CkkEnc* (*sparse* ($\mathbf{v}_{\text{glo}}$)) downloaded from the cloud server, which can be expressed by

$$\text{sparse}\left(\mathbf{v}_{\text{glo}}\right) = \langle \text{CkkEnc}\left(\text{sparse}\left(\mathbf{v}_{\text{glo}}\right)\right), \text{sk} \rangle \left(\text{mod } q_l\right). \tag{19}$$

4. Analysis

In this section, we present the security analysis and functionality analysis of our EFedID scheme.

4.1. Security Analysis

Definition 1 (CPA security). A private-key encryption scheme Π = Gen, Enc, Dec) has indistinguishable encryption under a chosen-plaintext attack (CPA) if for all probabilistic

polynomial-time adversaries A there exists a negligible function

$$\Pr\left[\mathrm{Priv}K_{A,\Pi}^{\mathrm{cpa}}(n)=1\right]\leq\frac{1}{2}+\mathrm{negl}(n), \tag{20}$$

where the probability is taken over the random coins used by A.

The CKKS cryptosystem-based secure communication protocol is proved to be indistinguishable against chosen-plaintext attack (CPA) based on the decisional composite residuosity problem, which means the ciphertexts will leak no bit of information about the plaintexts. Next, we would demonstrate that our scheme can protect industrial agents' data privacy.

Theorem 2. *Our proposed method can guarantee that no industrial agents' data information will be leaked if the CKKS cryptosystem-based secure communication protocol can against Chosen Plaintext Attack (CPA) when all components involved in the system are noncolluding.*

Proof. According to CPA-secure, we assume that there is an adversary in our system who intercepts the ciphertext of all the model parameters. However, in the absence of collusion, the adversary has no way to obtain the decrypted private key *sk* to decrypt the ciphertext. The key generation center will not distribute key pairs to entities other than agents. So, the adversary cannot infer the true value of the model parameters. For eavesdroppers in the process of parameter transmission, we have employed TLS/SSL (Transport Layer Security/ Secure Sockets Layer) to establish separate communication channels between the server and each agent to provide additional protection for the system to prevent eavesdroppers. Moreover, each industrial agent cannot obtain the model parameters of other agents because each industrial agent uses a separate channel to communicate with the cloud server. Therefore, in our system, we can ensure the privacy of the parameters of all models and the privacy of the industrial agent's data. □

4.2. Functionality Analysis. Now we compare the functionality of the latest FL deep learning models as shown in Table 2. MFL [27] uses the momentum term to accelerate convergence but does not take privacy into account. DPFL [32] and PFL [33] do not use the momentum term to speed up convergence, although privacy protection is considered. Compared with these schemes, our proposed EFedID uses AGS to reduce resource consumption while considering data privacy protection and model convergence rate during the training process.

5. Implementation and Evaluation

5.1. Settings

5.1.1. Environment. The experiment is performed on a computer that has NVIDIA 1080-Ti and 32 GB RAM. The EFedID is developed by CKKS-Python and PyTorch.

Table 2: Comparison of functionality with the latest FL models.

Function	MFL	PFL	DPFL	EFedID
Noninteractive	×	√	×	√
Privacy-preserving	×	√	√	√
Momentum	√	×	×	√
Gradient sparsification	×	×	×	√

5.1.2. Datasets and Data Preprocessing. We select three network intrusion detection datasets from different domains to validate the effectiveness of our scheme. The KDD CUP 99 [34] dataset is extracted based on packet traces from military network environments, and it is one of the most widely used datasets in the field of intrusion detection. The NSL-KDD [35] dataset is an updated version of the KDD CUP 99, which eliminates redundant data and selects a number of records from each difficulty level group that is inversely proportional to the percentage of records in the original KDD CUP 99 dataset. Therefore, the classification rates of different machine learning methods vary over a wider range, which makes the accurate assessment of different learning techniques more effective. In the recent literature [20, 27], many researchers use the NSL-KDD dataset as a valid baseline dataset, which can help researchers to compare different intrusion detection methods. CICIDS 2017 [36] is collected by the Canadian Institute for Cybersecurity Research in 2017, which contains the benign and newest common types of attacks similar to real-world data. Therefore, we decide to use the above three datasets to test the benchmark performance of our approach. The number of data records and characteristics in different datasets are given below:

(i) KDD CUP 99 dataset: This dataset consists of 5 million records, and we use a 10% training subset and a test set for our experiments. The training set has 494,021 training samples, and the test set has 311,092 test samples. The dataset includes 41 features, and there are 24 types of attacks in the training set and 38 types of attacks in the test set. The types of attacks can be classified as denial-of-service (DoS), attack from remote to local machine (R2L), unauthorized access to local administrator user (U2R), and probing attack 4 types.

(ii) NSL-KDD dataset: In our paper, we have used the "$\mathrm{KDDTrain}^{+}$" dataset for the model's training and the "$\mathrm{KDDTest}^{+}$" dataset for testing. The "$\mathrm{KDDTrain}^{+}$" dataset has 125973 records and the "$\mathrm{KDDTest}^{+}$" dataset has 22544 records. Each record contains 41 feature attributes and one label attribute. There are 9 discrete features and the rest are continuous features.

(iii) CICIDS 2017 dataset: This dataset consists of 3429031 records and we use 80% of the data as the training set and 20% of the data as the test set. Each record has 80 features, and the dataset has 14 types of attacks.

We assume that the data are independently and identically distributed (IID). The data are shuffled and

partitioned to each industrial agent. Before model training, the industrial agents in the FL system process their local data to generate training and testing samples. First, the random forest algorithm is used to perform feature analysis on the data, and the top 25 features are selected for model training. Next, we use one-hot encoding to encode the three features of "protocol_type," "service," and "flag." This operation can convert data into numerical values, which is convenient for neural network processing. In addition, we use Min-Max normalization to scale the samples to the range of 0-1.

5.1.3. Models. We design experiments to study the performance of the proposed EFedID for multiclass classification on the NSL-KDD, KDD CUP 99, and CICIDS 2017 datasets. We use our designed CNN-based intrusion detection model as the local model of industrial agents. A dropout layer with a dropout rate of 0.5 is used between the first fully connected layer and the second fully connected layer to control overfitting. Rectified linear unit (ReLU) is used as the activation function of each hidden layer. A momentum gradient descent (MGD) optimizer with a momentum rate of 0.5 is adopted to train models. The loss function is the cross-entropy cost function. The mini-batch size, the aggregation round, the learning rate, and the decay rate d_{rate} used in training the networks are 512, 1500, 0.05, and 0.001, respectively.

5.1.4. Performance Metrics. In this paper, we use the accuracy, precision, recall, and compression rate to evaluate the performance of the different methods: (1) accuracy—the proportion of correctly classified samples to the total number of samples; (2) precision—the percentage of records predicted to be of categories are indeed those categories; (3) recall—the proportion of all correctly predicted category records to exact types of categories; (4) F1-score—the harmonic mean of the accuracy and recall, with the maximum value of 1 and the minimum value of 0.5; and (5) compression rate—the proportion of the number of the transmission parameters computed as $\text{CR} = \sum_{t=1}^{N}\sum_{k=1}^{K} |\text{sparse}(\mathbf{v}_t^{\text{a},k})| / \sum_{t=1}^{N}\sum_{k=1}^{K} |\mathbf{v}_t^{\text{a},k}|$, where N denotes the aggregation rounds.

5.2. Case 1. In this experiment, we conduct a comprehensive evaluation by comparing our proposed EFedID approach with three other prominent methods: centralized learning (CL), privacy-preserving federated learning (PFL), and PFL-GS using a fixed sparsification rate of 0.8. The comparison is performed across various dimensions to assess the efficiency and effectiveness of EFedID.

5.2.1. Accuracy. Accuracy is a critical metric for evaluating intrusion detection models. As illustrated in Figures 4(a), 5(a), and 6(a), our EFedID initially exhibits lower accuracy in the early aggregation rounds due to its higher sparsification rate during this phase. Then, with the increase of the number of aggregation rounds, the accuracy of EFedID steadily improves, eventually converging to levels closely matching those of PFL. Further insights are provided in Figures 7–9, which present histograms derived from experimental results.

Notably, the centralized learning (CL) approach attains the highest accuracy, achieving 84.33%, 97.35%, and 95.85% on different datasets, making it the top-performing model among the four learning approaches. For the purpose of baseline comparison, we chose PFL. Importantly, it is worth highlighting that EFedID and PFL-GS ($\varphi = 0.8$) demonstrate comparable model performance to PFL. This result underscores that EFedID does not affect model performance on different datasets.

5.2.2. Resource Costs. Resource consumption is a critical concern in practical deployments. Figures 4(c), 5(c), and 6(c) show transmission parameter curves on different datasets, while Figures 7–9 present histograms of these transmission parameters. The number of transmission parameters plays an important role in determining cryptographic computation and communication overheads. Observing Figures 4(c), 5(c), and 6(c), it becomes evident that the number of transmission parameters in PFL significantly increases with each iterative round across diverse datasets. In contrast, the transmission parameter count in PFL-GS exhibits a more gradual increment. The histograms of the experimental results clearly illustrate the efficacy of our GS method, reducing the transmission parameter count by a factor of 5 ($\varphi = 0.8$). Moreover, EFedID further enhances compression rates by 1.92 times, 1.78 times, and 1.23 times on different datasets. This observation underscores the AGS method's capability to further reduce resource consumption while maintaining model performance.

5.3. Case 2. In this section, we delve into the validity of our AGS method by contrasting it with PFL utilizing different fixed sparsification rates ($\varphi = 0.8$ and $\varphi = 0.9$) for comparison with our EFedID approach. Figure 10 provides an overview of the experimental curves, while Figure 11 presents detailed experimental results on the NSL-KDD dataset. An initial observation reveals that EFedID's accuracy convergence appears slower than that of PFL-GS ($\varphi = 0.9$) and PFL-GS ($\varphi = 0.8$) in the early rounds of aggregation. This divergence can be attributed to the comparatively higher sparsification rate employed by EFedID during these initial aggregation rounds. Figure 10(c) underscores this point by showing that the number of transmission parameters gradually increases with iterative rounds in PFL-GS ($\varphi = 0.8$). However, when compared to PFL-GS ($\varphi = 0.8$), both EFedID and PFL-GS ($\varphi = 0.9$) exhibit a more gradual increase in the transmission parameter count. From Figure 11, we observe that the accuracy of PFL-GS ($\varphi = 0.8$) reaches 84.45%, surpassing the other two learning approaches, with EFedID closely following. Conversely, the lowest accuracy, 83.07%, is attributed to PFL-GS ($\varphi = 0.9$), underscoring the significant impact of the sparsification rate on model accuracy. Regarding the reduction in the number of transmitted parameters, PFL-GS ($\varphi = 0.9$) achieves a remarkable

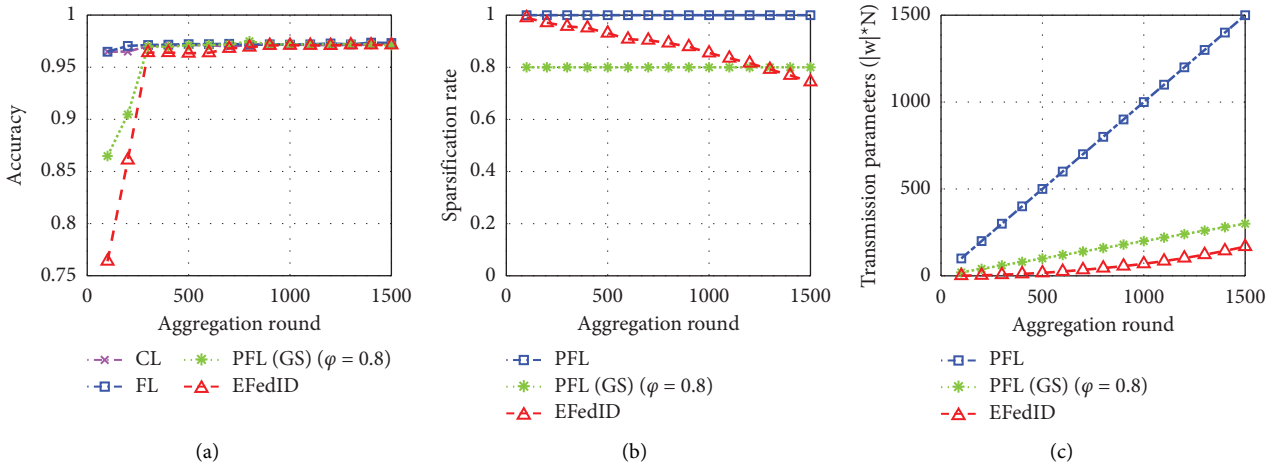

FIGURE 4: Curves of the different learning models (dataset: KDD CUP 99). (a) Accuracy curves. (b) Sparsification rate curves. (c) Transmission parameter curves.

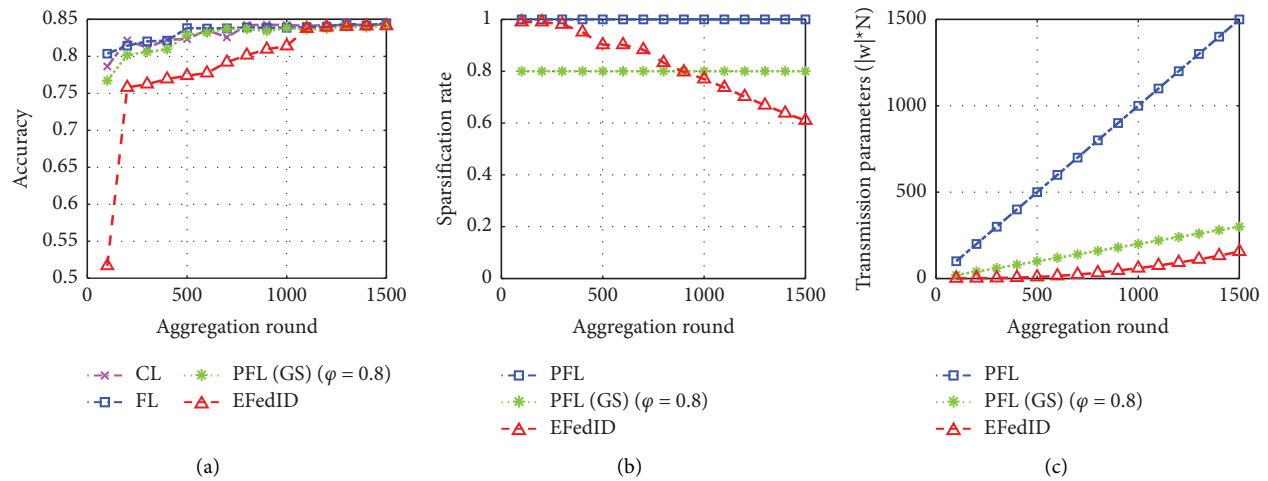

FIGURE 5: Curves of the different learning models (dataset: NSL-KDD). (a) Accuracy curves. (b) Sparsification rate curves. (c) Transmission parameter curves.

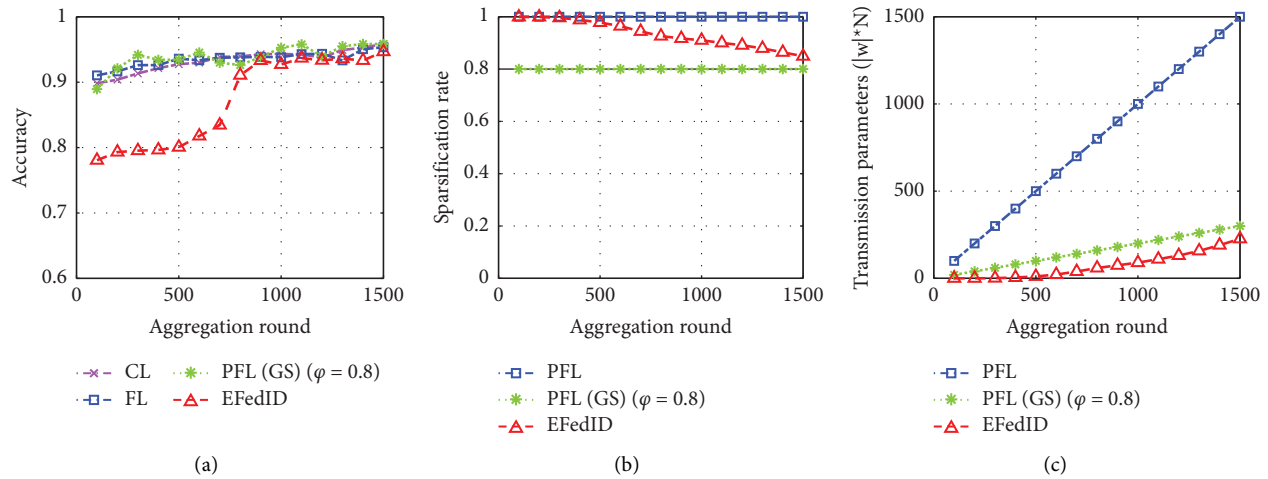

FIGURE 6: Curves of the different learning models (dataset: CICIDS 2017). (a) Accuracy curves. (b) Sparsification rate curves. (c) Transmission parameter curves.

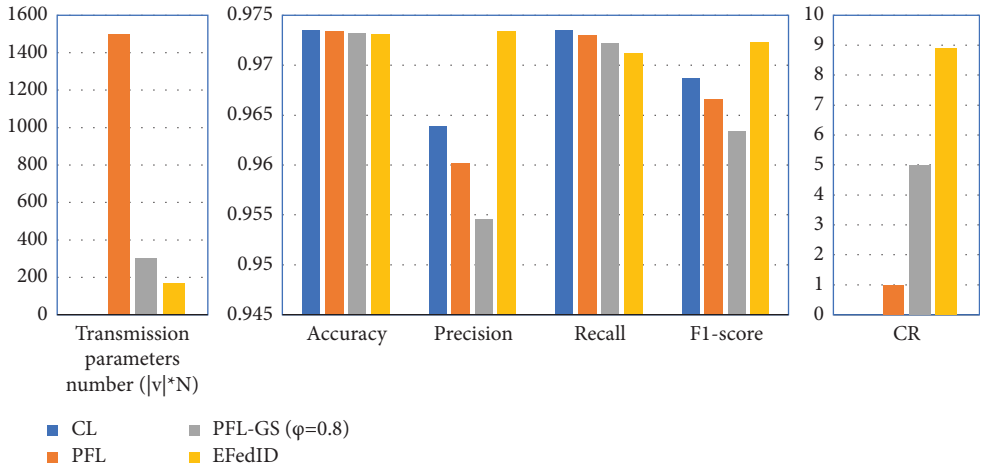

Figure 7: Results of the different learning models (dataset: KDD CUP 99).

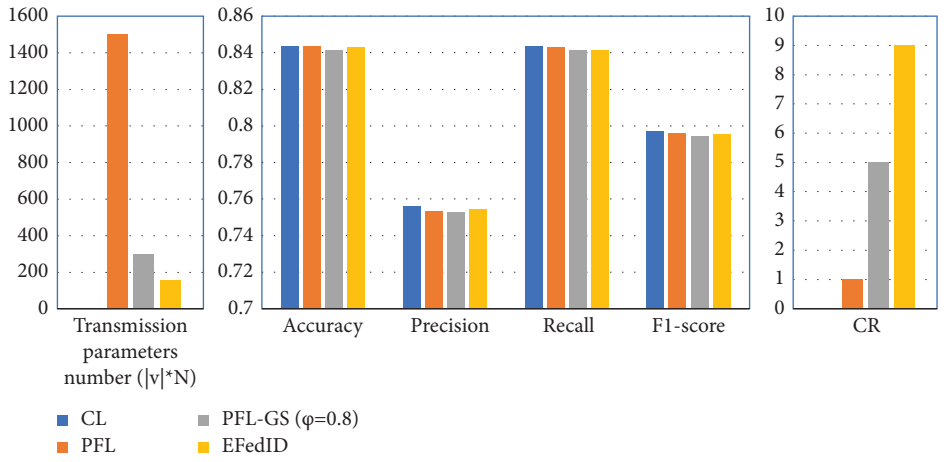

Figure 8: Results of the different learning models (dataset: NSL-KDD).

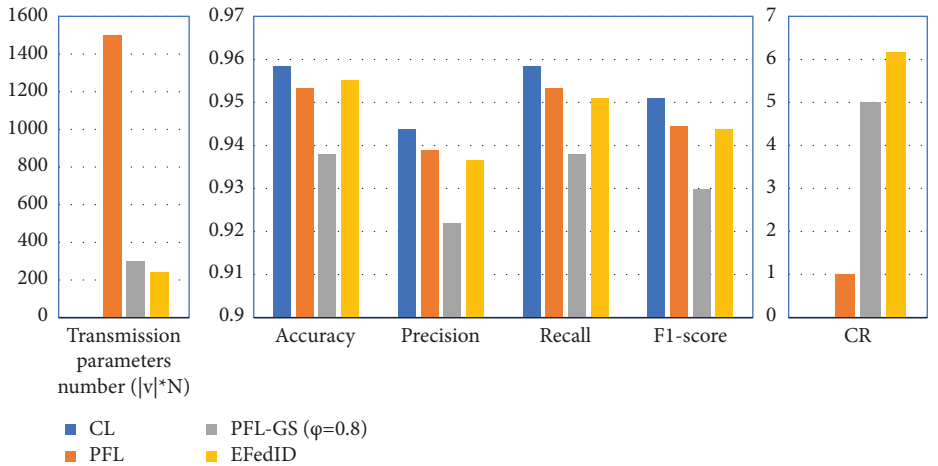

Figure 9: Results of the different learning models (dataset: CICIDS 2017).

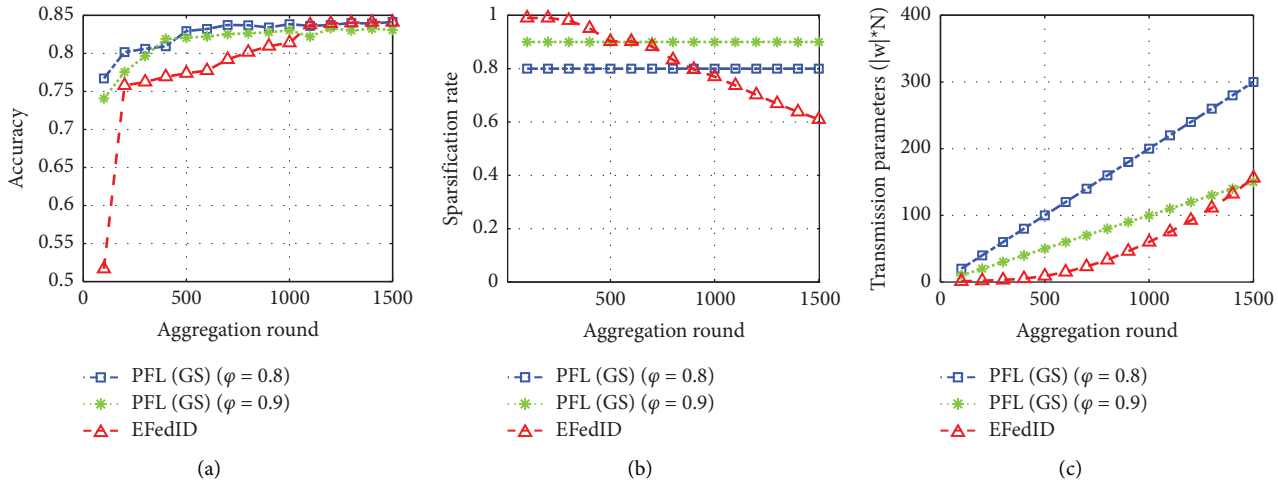

FIGURE 10: Curves of the PFL with different sparsification rates (dataset: NSL-KDD). (a) Accuracy curves. (b) Sparsification rate curves. (c) Transmission parameter curves.

compression ratio of 10 times, followed by EFedID at 9.6 times. However, EFedID outperforms PFL-GS ($\varphi = 0.9$) in terms of accuracy improvement. Notably, EFedID manages to halve the number of transmitted parameters compared to PFL-GS ($\varphi = 0.8$).

To validate our AGS method further, we extend our analysis to the KDD CUP 99 and CICIDS 2017 datasets, as illustrated in Figures 12–15. Remarkably, similar trends emerge across these datasets, affirming the consistent improvement in training efficiency achieved by our adaptive sparsification rate method compared to the fixed sparsification rate approach.

In conclusion, our results underscore the effectiveness of our AGS method in enhancing training efficiency, particularly when compared to fixed sparsification rate methods. EFedID's adaptability to different datasets and its ability to maintain accuracy while significantly reducing resource consumption make it a promising choice for privacy-preserving federated learning in various applications.

5.4. Case 3. In this subsection, we explore the adaptability of our EFedID method to a distributed computing environment and investigate the influence of the momentum rate (γ) on its convergence rate.

5.4.1. Varying the Number of Agents N. We begin by varying the number of industrial agents (N) to assess how EFedID performs under different distributed scenarios. The curves of EFedID for N ranging from 10 to 50 are depicted in Figures 16–18, derived from simulations. As illustrated in Figures 16(a), 17(a), and 18(a), with the increase in the number of aggregation rounds, all accuracy curves of EFedID gradually improve and converge to similar accuracy levels by the end of the training process. Complementary insights are provided in Figures 19–21, presenting histograms based on experimental results. Notably, our EFedID exhibits comparable performance between scenarios with $N = 20$ and $N = 50$ when compared to the baseline scenario with $N = 10$ (chosen as a reference point). This observation suggests that small variations in the number of agents (N) do not significantly affect EFedID's performance. This robustness underscores the adaptability of our proposed scheme to edge-computing industrial environments.

In summary, our EFedID method's consistent performance across varying numbers of agents demonstrates its suitability for a distributed computing environment. Additionally, our investigation into the impact of the momentum rate (γ) on the convergence rate will provide further insights into EFedID's optimization potential in such environments.

5.4.2. Impact of γ. In this subsection, we explore the influence of the momentum rate (γ) on the performance of our EFedID method, as illustrated in Figures 22–24. As observed, in the figures, the convergence rates gradually increase as γ is adjusted from 0 to 0.7. This trend indicates that the incorporation of the momentum term enhances the convergence speed of EFedID. Notably, the sparsification rate experiences a slightly faster decay as γ increases. This behavior can be attributed to the acceleration of adaptive adjustments caused by the model rapidly approaching the adjustment threshold. To provide detailed insights into EFedID's performance with different γ values, we present the results in Figures 25–27. In this evaluation, we use EFedID with $\gamma = 0$ as the baseline for comparison. Across the NSL-KDD dataset, EFedID with $\gamma = 0.7$ achieves the highest accuracy at 0.8450, followed by EFedID with $\gamma = 0.5$ (accuracy of 0.8431) and EFedID with $\gamma = 0.3$ (accuracy of 0.8414). Notably, all models with γ values outperform the baseline model with $\gamma = 0$, highlighting the positive impact of the momentum term on accuracy improvement. A similar trend is observed in the KDD CUP 99 dataset, where EFedID with $\gamma = 0.7$ attains the highest accuracy of 0.9735, followed by EFedID with $\gamma = 0.5$ (accuracy of 0.9712) and EFedID with $\gamma = 0.3$ (accuracy of 0.9706). Once again, all models with varying γ values

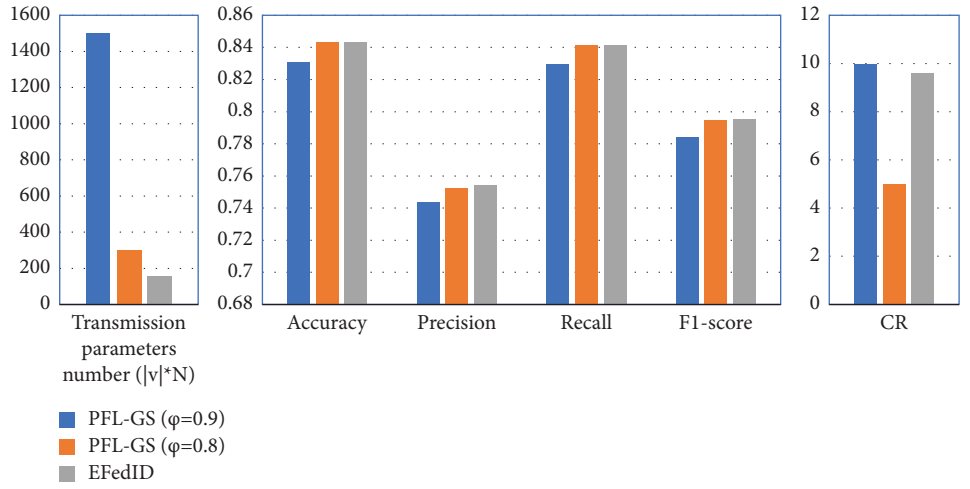

Figure 11: Results of the PFL with different sparsification rates (dataset: NSL-KDD).

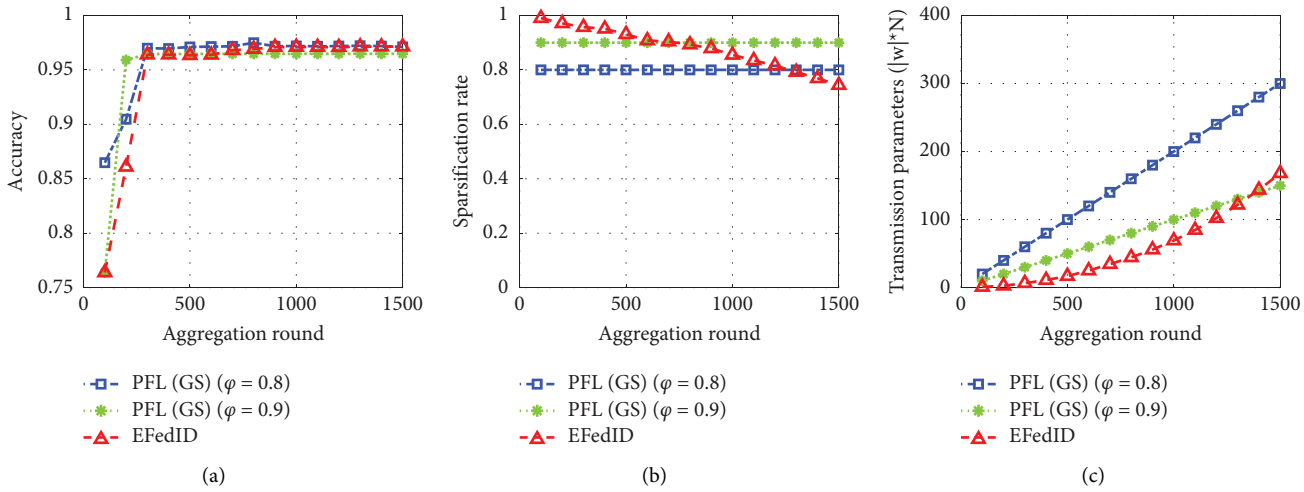

Figure 12: Curves of the PFL with different sparsification rates (dataset: KDD CUP 99). (a) Accuracy curves. (b) Sparsification rate curves. (c) Transmission parameter curves.

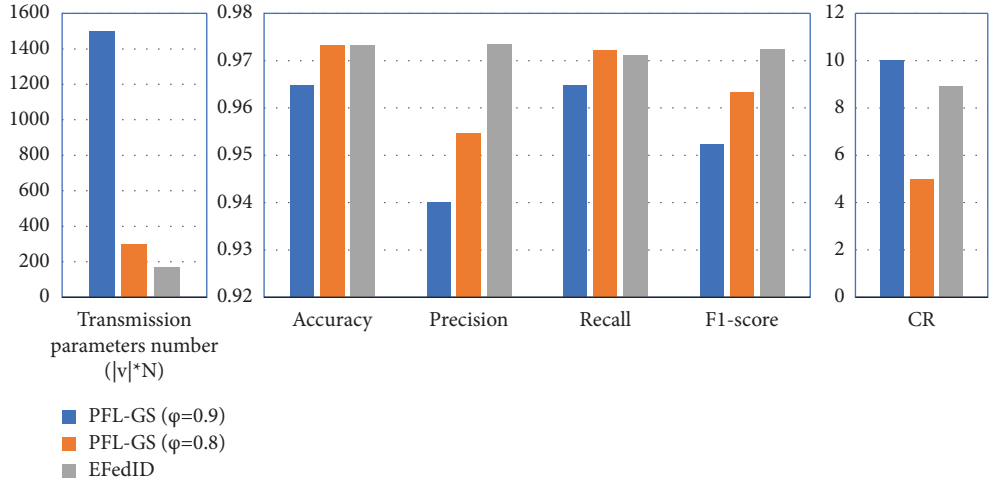

Figure 13: Results of the PFL with different sparsification rates (dataset: KDD CUP 99).

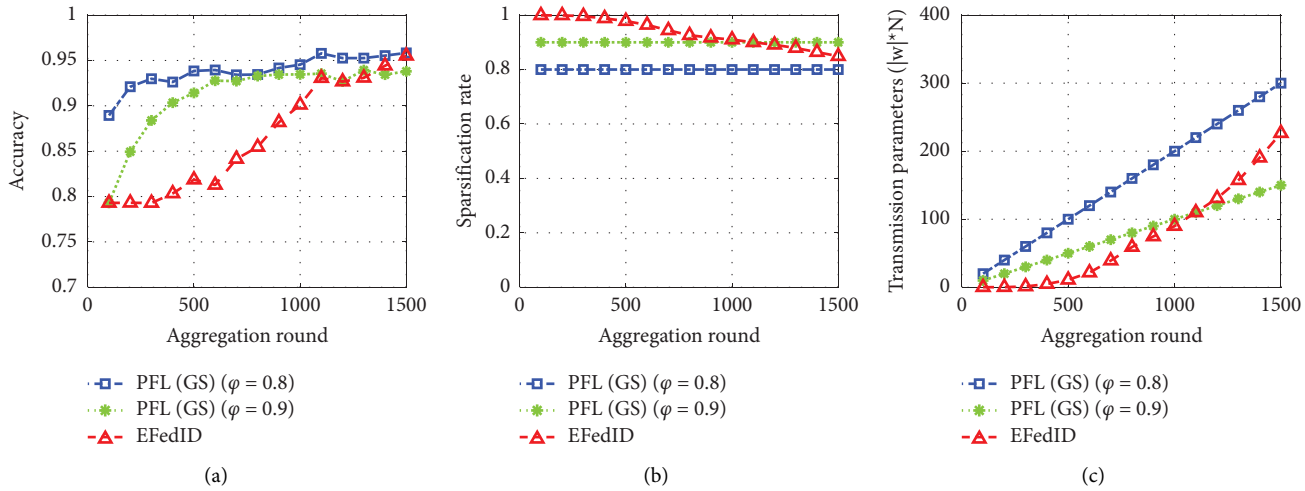

Figure 14: Curves of the PFL with different sparsification rates (dataset: CICIDS 2017). (a) Accuracy curves. (b) Sparsification rate curves. (c) Transmission parameter curves.

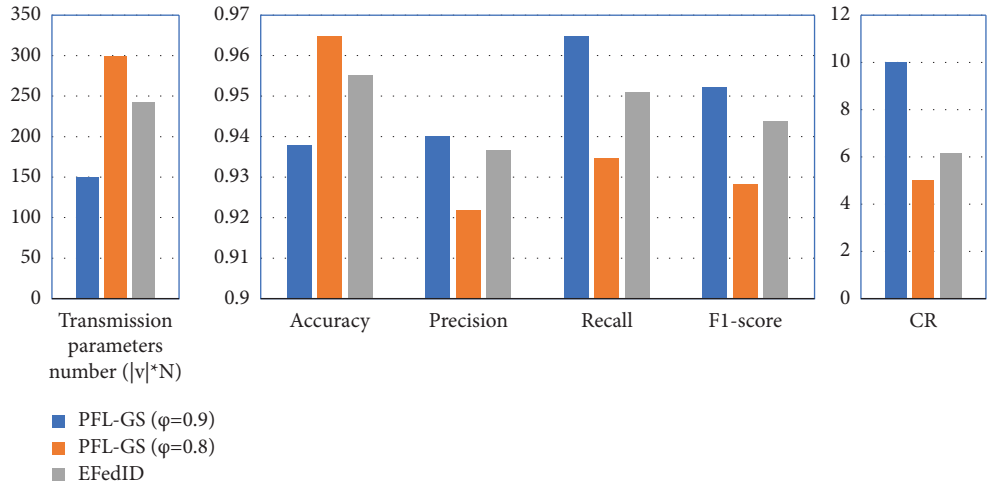

Figure 15: Results of the PFL with different sparsification rates (dataset: CICIDS 2017).

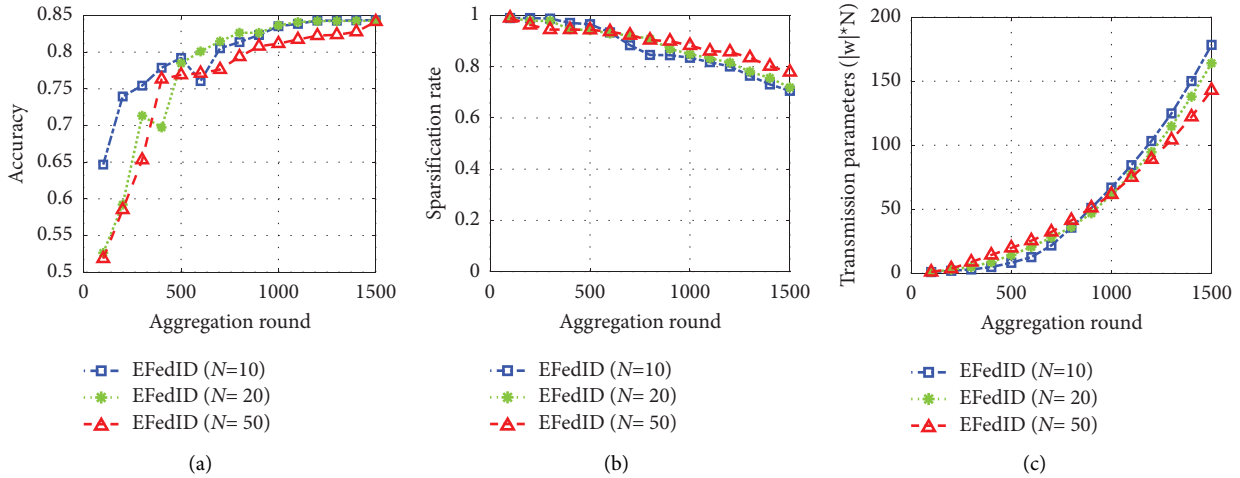

Figure 16: Curves of the EFedID with different N (dataset: NSL-KDD). (a) Accuracy curves. (b) Sparsification rate curves. (c) Transmission parameter curves.

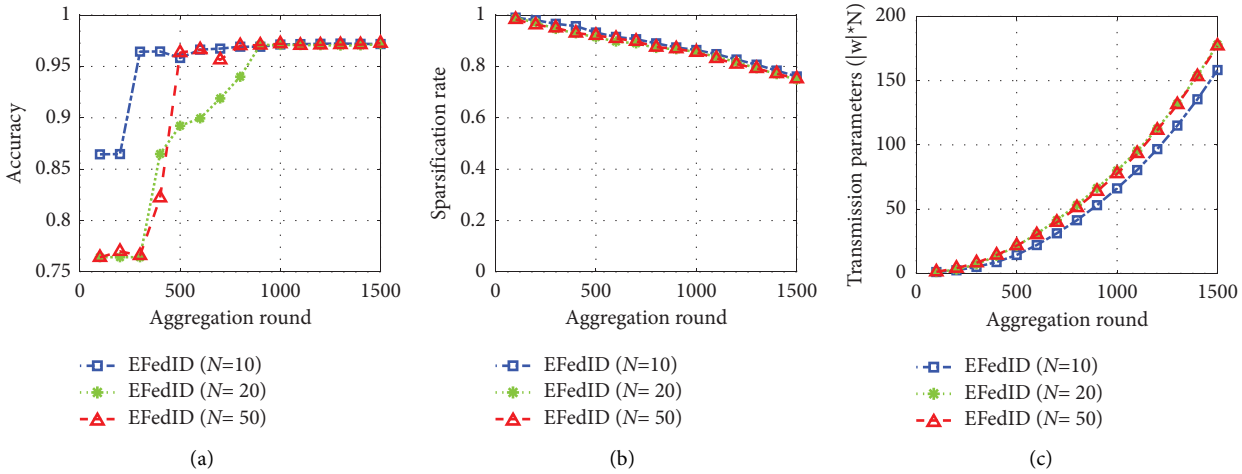

Figure 17: Curves of the EFedID with different N (dataset: KDD CUP 99). (a) Accuracy curves. (b) Sparsification rate curves. (c) Transmission parameter curves.

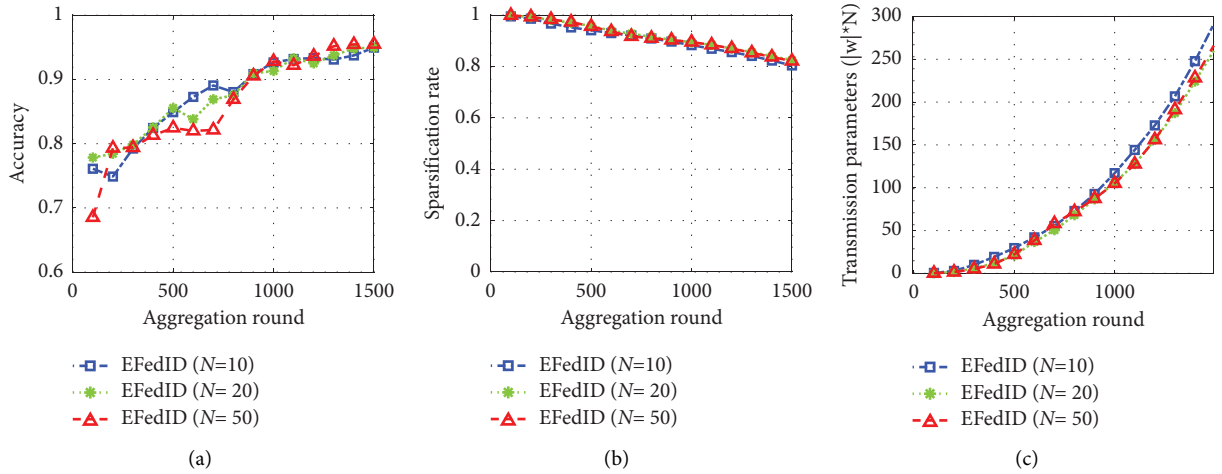

Figure 18: Curves of the EFedID with different N (dataset: CICIDS 2017). (a) Accuracy curves. (b) Sparsification rate curves. (c) Transmission parameter curves.

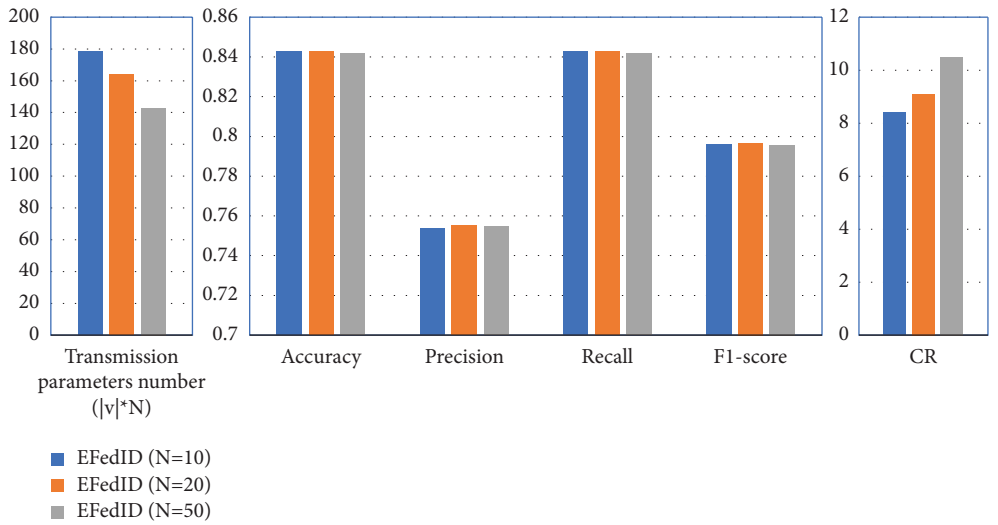

Figure 19: Results of the EFedID with different N (dataset: NSL-KDD).

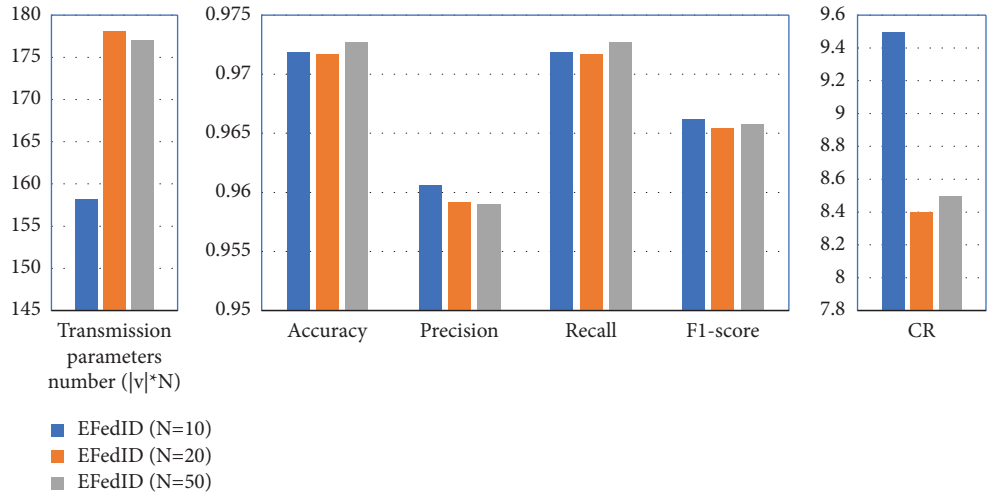

Figure 20: Results of the EFedID with different N (dataset: KDD CUP 99).

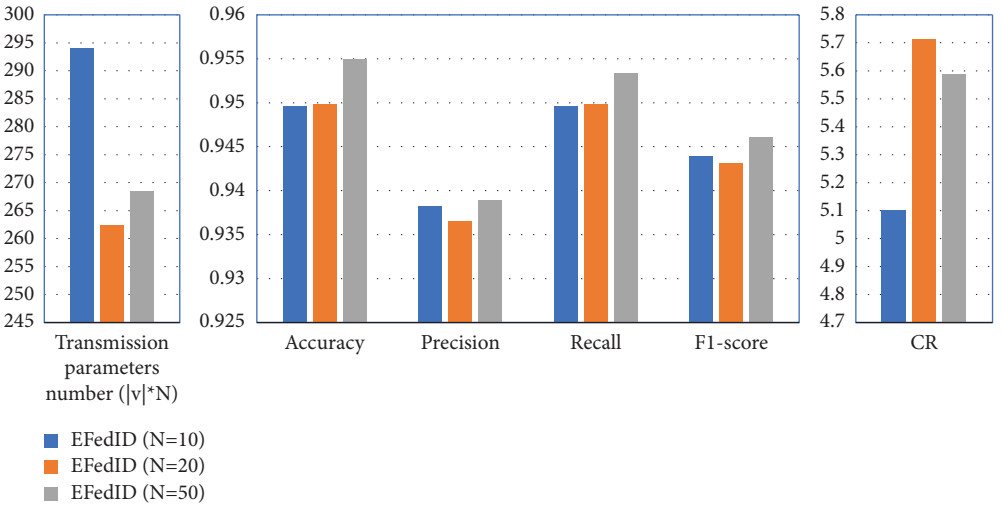

Figure 21: Results of the EFedID with different N (dataset: CICIDS 2017).

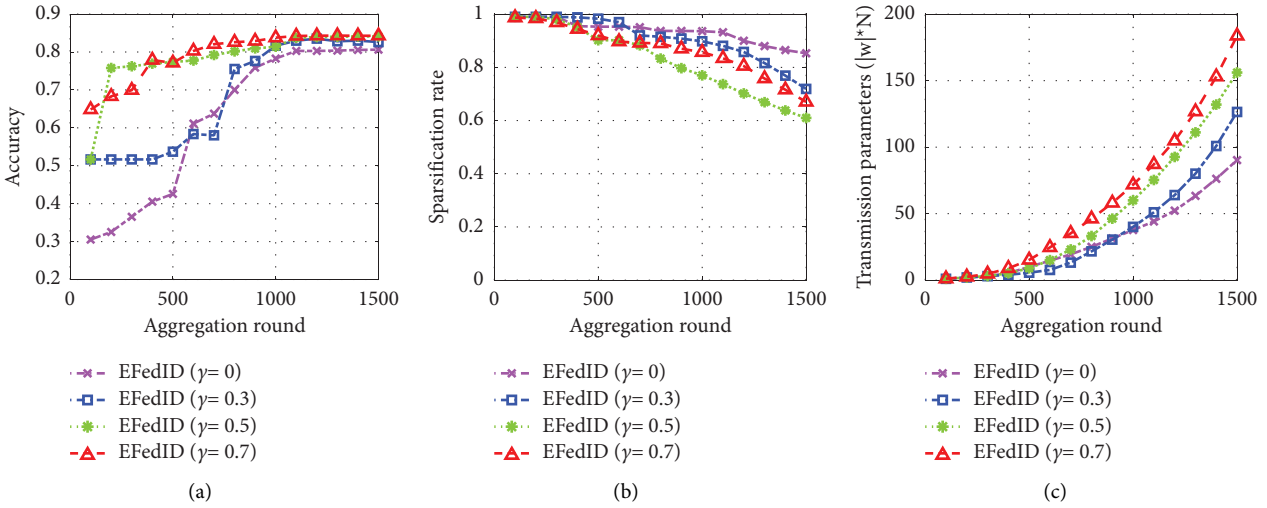

Figure 22: Curves of the EFedID with different γ (dataset: NSL-KDD). (a) Accuracy curves. (b) Sparsification rate curves. (c) Transmission parameter curves.

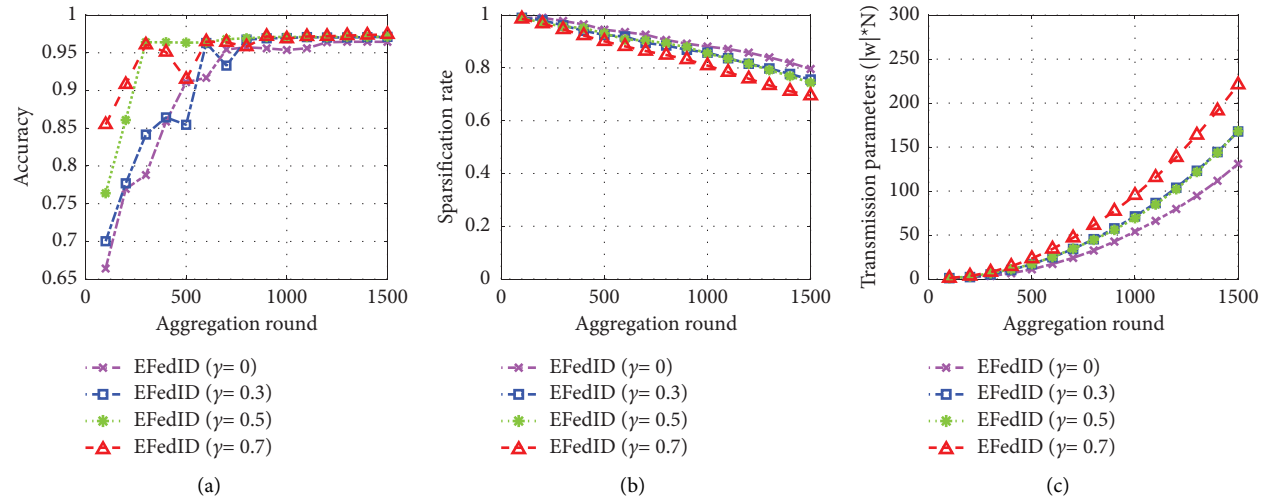

Figure 23: Curves of the EFedID with different γ (dataset: KDD CUP 99). (a) Accuracy curves. (b) Sparsification rate curves. (c) Transmission parameter curves.

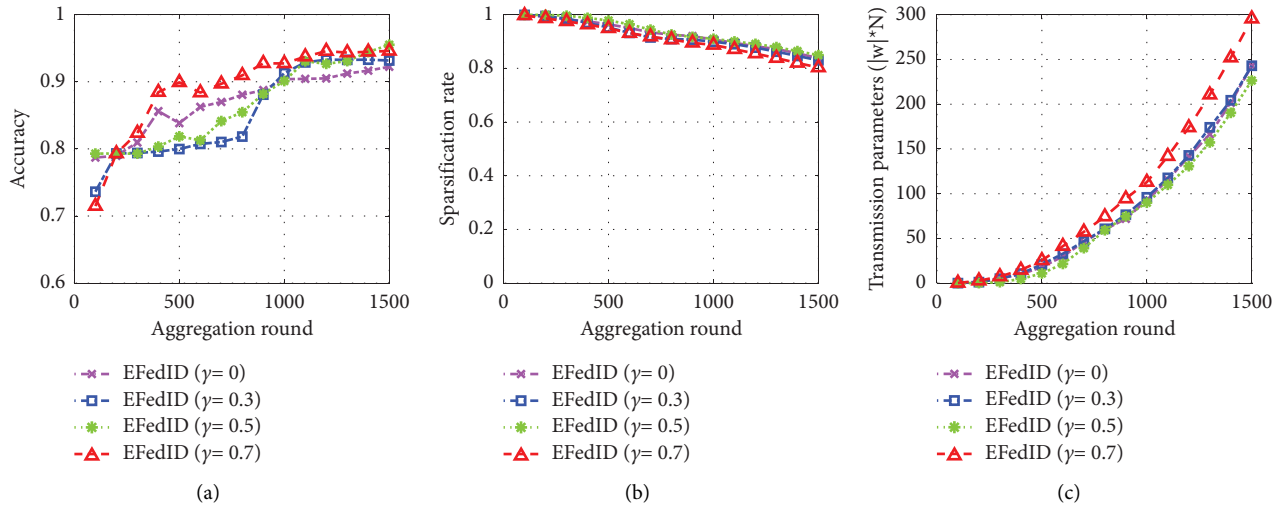

Figure 24: Curves of the EFedID with different γ (dataset: CICIDS 2017). (a) Accuracy curves. (b) Sparsification rate curves. (c) Transmission parameter curves.

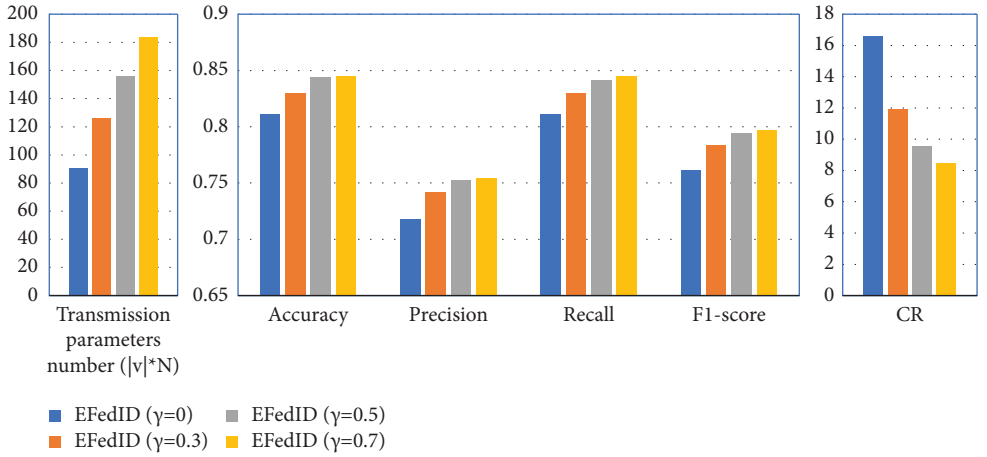

Figure 25: Results of the EFedID with different γ (dataset: NSL-KDD).

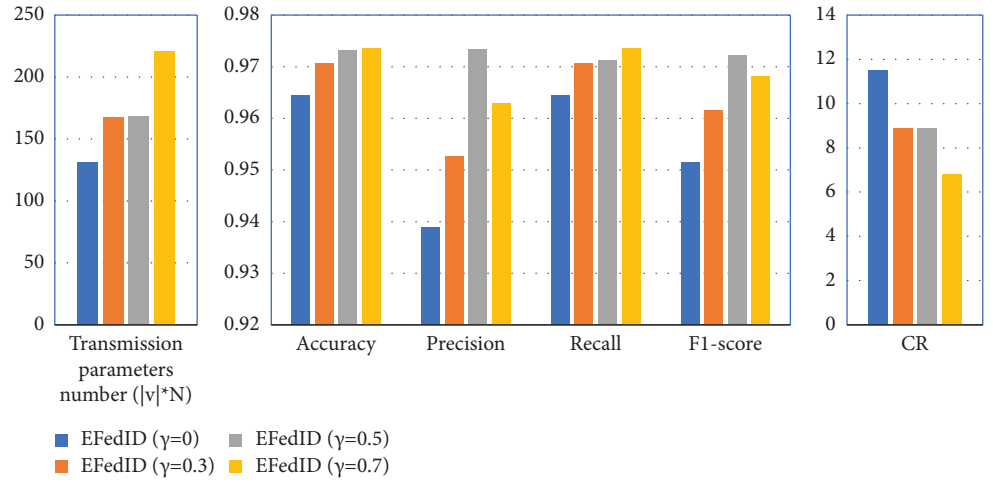

Figure 26: Results of the EFedID with different γ (dataset: KDD CUP 99).

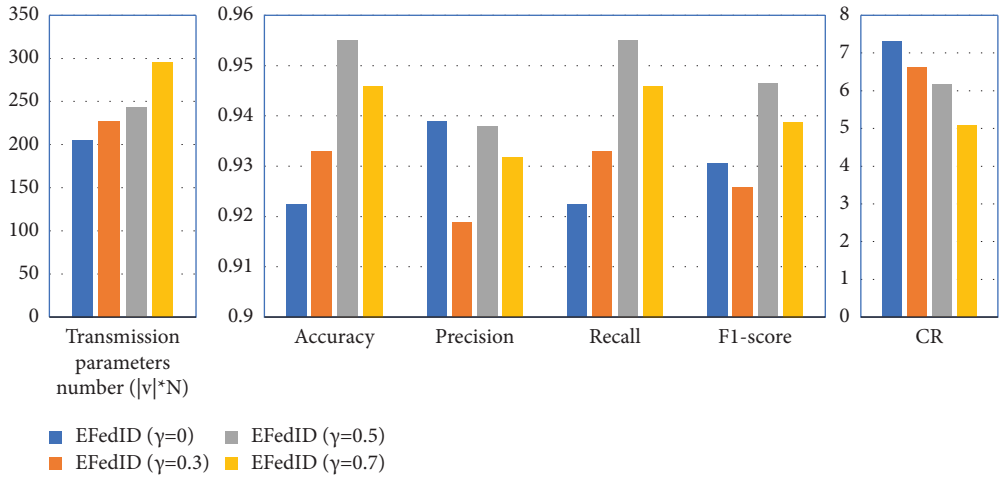

Figure 27: Results of the EFedID with different γ (dataset: CICIDS 2017).

Table 3: The accuracy of the EFedID and the other latest models.

Dataset	Model	Accuracy (%)
KDD CUP 99	CNN [37]	92.14
	Three-stage SMOTE-GAN-VAE [39]	94.0
	NIDS-CNNLSTM [38]	97.05
	EFedID	**97.3**
NSL-KDD	Muli-CNN [40]	81.33
	AE-R with SMOTE [44]	82.09
	SCAE + SVM [41]	84.16
	EFedID	**84.31**
CICIDS 2017	Three-stage SMOTE-GAN-VAE [39]	94.5
	FL [42]	93
	CPIO [43]	93.64
	EFedID	**95.51**

Bold values represent the best results of the several compared models.

outperform the $\gamma = 0$ baseline, underscoring the benefits of utilizing the momentum term. Figures 24 and 27 illustrate the experiment curves and detailed results on the CICIDS 2017 dataset, where a similar pattern emerges, further reinforcing the positive impact of adjusting γ. In conclusion, our analysis of different γ values indicates that EFedID's convergence rate can be accelerated by incorporating the momentum term.

5.5. Performance Comparison with Other Latest Methods. In the same way, we compare the performance of EFedID with some of the latest other notable state-of-the-art methods. Models CNN [37] and NIDS-CNNLSTM [38] were both trained using the KDDTrain$^+$ dataset and tested using the KDDTest$^+$ dataset. The model SMOTE-GAN-VAE [39] was evaluated on the NSL-KDD and CICIDS 2017 datasets, respectively. Muli-CNN [40] and SCAE + SVM [41] showed their performance using the NSL-KDD dataset. FL [42] and CPIO [43] showed their performance using the CICIDS 2017 dataset. Table 3 presents the experiment results of the above compared methods. It can be seen that the proposed EFedID model performs better than other state-of-the-art methods in multivariate classification.

6. Conclusion

In this article, we proposed EFedID, which allows distributed industrial agents to collaboratively train a network intrusion detection model. The resource overheads of the FL system are reduced by using the AGS algorithm, and the performance of the model is guaranteed. Moreover, we demonstrated the preservation of data privacy for industrial agents through a secure communication protocol based on the CKKS cryptosystem. The experiment results show that our proposed method can efficiently organize multiple industrial agents to collaboratively train a network intrusion detection model and protect the industrial agents' data.

Since the training data are collected by each client in its own local environment and follows its usage patterns, the size and distribution of the local datasets often different. Non-IID (not identically and independently distributed) data for privacy-preserving federated learning models can reflect practical scenarios. In the future, we will also focus on investigating how to adjust the model aggregation interval to further improve training efficiency for privacy-preserving federated learning on non-IID data. This will allow us to continue pushing the boundaries of secure and efficient collaborative learning in diverse industrial settings.

Data Availability

Previously reported KDD CUP 99, NSL-KDD, and CICIDS 2017 datasets were used to support this study. These prior studies (and datasets) are cited at relevant places within the text as references [34–36].

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Authors' Contributions

Ningxin He and Zehui Zhang contributed equally to this work.

Acknowledgments

The authors gratefully acknowledge the financial support provided by National Science and Technology Major Project of China (2021YFB0300104), Tianjin Research Innovation Project for Postgraduate Students (2022BKY013), and National Key R&D Projects (2022YFE0210700).

References

[1] F. Liang, W. Yu, X. Liu, D. Griffith, and N. Golmie, "Towards deep Q-network based resource allocation in industrial internet of Things," *IEEE Internet of Things Journal*, vol. 9, p. 1, 2021.

[2] J. Franco, A. Aris, B. Canberk, and A. S. Uluagac, "A survey of honeypots and honeynets for internet of Things, industrial internet of Things, and cyber-physical systems," *IEEE Communications Surveys and Tutorials*, vol. 23, no. 4, pp. 2351–2383, 2021.

[3] J. Yu, X. Ye, and H. Li, "A high precision intrusion detection system for network security communication based on multi-scale convolutional neural network," *Future Generation Computer Systems*, vol. 129, 2021.

[4] S. Roy, J. Li, B. J. Choi, and Y. Bai, "A lightweight supervised intrusion detection mechanism for IoT networks," *Future Generation Computer Systems*, vol. 127, pp. 276–285, 2022.

[5] E. U. H. Qazi, M. Imran, N. Haider, M. Shoaib, and I. Razzak, "An intelligent and efficient network intrusion detection system using deep learning," *Computers and Electrical Engineering*, vol. 99, Article ID 107764, 2022.

[6] I. A. Khan, N. Moustafa, D. Pi, K. M. Sallam, A. Y. Zomaya, and B. Li, "A new explainable deep learning framework for cyber threat discovery in industrial IoT networks," *IEEE Internet of Things Journal*, vol. 9, no. 13, pp. 11604–11613, 2022.

[7] M. Ismail, M. F. Shaaban, M. Naidu, and E. Serpedin, "Deep learning detection of electricity theft cyber-attacks in renewable distributed generation," *IEEE Transactions on Smart Grid*, vol. 11, no. 4, pp. 3428–3437, 2020.

[8] M. Wazzeh, H. Ould-Slimane, C. Talhi, A. Mourad, and M. Guizani, "Privacy-preserving continuous authentication for mobile and iot systems using warmup-based federated learning," *IEEE Network*, vol. 37, no. 3, pp. 224–230, 2023.

[9] M. N. H. Nguyen, N. H. Tran, Y. K. Tun, Z. Han, and C. S. Hong, "Toward multiple federated learning services resource sharing in mobile edge networks," *IEEE Transactions on Mobile Computing*, vol. 22, p. 1, 2021.

[10] A. Li, L. Zhang, J. Wang, F. Han, and X. Y. Li, "Privacy-preserving efficient federated-learning model debugging," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, p. 1, 2021.

[11] S. Agrawal, S. Sarkar, O. Aouedi et al., "Federated learning for intrusion detection system: concepts, challenges and future directions," *Computer Communications*, vol. 195, pp. 346–361, 2022.

[12] M. Cao, L. Zhang, and B. Cao, "Toward on-device federated learning: a direct acyclic graph-based blockchain approach," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, pp. 1–15, 2021.

[13] M. Aloqaily, I. Al Ridhawi, and M. Guizani, "Energy-aware blockchain and federated learning-supported vehicular networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, pp. 1–12, 2021.

[14] Y. Zheng, S. Lai, Y. Liu, X. Yuan, X. Yi, and C. Wang, "Aggregation service for federated learning: an efficient, secure, and more resilient realization," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, 2022.

[15] U. Ahmed, G. Srivastava, and J. C. W. Lin, "Reliable customer analysis using federated learning and exploring deep-attention edge intelligence," *Future Generation Computer Systems*, vol. 127, pp. 70–79, 2022.

[16] S. Banabilah, M. Aloqaily, E. Alsayed, N. Malik, and Y. Jararweh, "Federated learning review: fundamentals, enabling technologies, and future applications," *Information Processing and Management*, vol. 59, no. 6, Article ID 103061, 2022.

[17] Z. Tang, H. Hu, and C. Xu, "A federated learning method for network intrusion detection," *Concurrency and Computation: Practice and Experience*, vol. 34, no. 10, 2022.

[18] I. A. Khan, D. Pi, M. Z. Abbas, U. Zia, Y. Hussain, and H. Soliman, "Federated-SRUs: a federated simple recurrent units-based IDS for accurate detection of cyber attacks against IoT-augmented industrial control systems," *IEEE Internet of Things Journal*, vol. 10, 2022.

[19] H. Wang, J. Gu, and S. Wang, "An effective intrusion detection framework based on SVM with feature augmentation," *Knowledge-Based Systems*, vol. 136, pp. 130–139, 2017.

[20] Y. Li, Y. Xu, Z. Liu et al., "Robust detection for network intrusion of industrial IoT based on multi-CNN fusion," *Measurement*, vol. 154, 2020.

[21] C. Wu and W. Li, "Enhancing intrusion detection with feature selection and neural network," *International Journal of Intelligent Systems*, vol. 36, no. 7, pp. 3087–3105, 2021.

[22] I. A. Khan, M. Keshk, D. Pi, N. Khan, Y. Hussain, and H. Soliman, "Enhancing IIoT networks protection: a robust security model for attack detection in Internet Industrial Control Systems," *Ad Hoc Networks*, vol. 134, Article ID 102930, 2022.

[23] J. Zhang, Y. Wang, K. Zhu, Y. Zhang, and Y. Li, "Diagnosis of interturn short-circuit faults in permanent magnet synchronous motors based on few-shot learning under a federated learning framework," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 12, pp. 8495–8504, 2021.

[24] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, "Communication-efficient federated learning for digital twin edge networks in industrial IoT," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5709–5718, 2021.

[25] W. Zhang, T. Zhou, Q. Lu et al., "Dynamic-fusion-based federated learning for COVID-19 detection," *IEEE Internet of Things Journal*, vol. 8, no. 21, pp. 15884–15891, 2021.

[26] M. Aloqaily, I. Al Ridhawi, and F. Karray, "Towards blockchain-based hierarchical federated learning for cyber-physical systems," in *Proceedings of the 2022 International Balkan Conference on Communications and Networking (BalkanCom)*, pp. 46–50, Sarajevo, Bosnia and Herzegovina, August, 2022.

[27] W. Liu, L. Chen, Y. Chen, and W. Zhang, "Accelerating federated learning via momentum gradient descent," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 8, pp. 1754–1766, 2020.

[28] P. Han, S. Wang, and K. K. Leung, "Adaptive gradient sparsification for efficient federated learning: an online learning approach," *IEEE ICDCS*, July 2020, Hong Kong, China, pp. 300–310.

[29] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security*, vol. 10624, pp. 409–437, Springer, Berlin, Germany, May, 2017.

[30] M. Babenko and E. Golimblevskaia, "Euclidean division method for the homomorphic scheme ckks," in *Proceedings of the 2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus)*, pp. 217–220, Moscow, Russia, January, 2021.

[31] E. M. Shiriaev, A. S. Nazarov, N. N. Kycherov, and N. A. Sotikova, "Efficient implementation of the CKKS scheme using a quadratic residue number system," in *Proceedings of the 2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus)*, pp. 665–669, IEEE, Moscow, Russia, January, 2021.

[32] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and federated learning for privacy-preserved data sharing in industrial IoT," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4177–4186, 2020.

[33] Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Transactions on Information Forensics and Security*, vol. 13, pp. 1333–1345, 2017.

[34] S. K. Sahu, S. Sarangi, and S. K. Jena, "A detail analysis on intrusion detection datasets," in *Proceedings of the 2014 IEEE international advance computing conference (IACC)*, pp. 1348–1353, IEEE, New Delhi, India, February, 2014.

[35] S. A. Rahman, H. Tout, C. Talhi, and A. Mourad, "Internet of Things intrusion detection: centralized, on-device, or federated learning?" *IEEE Network*, vol. 34, no. 6, pp. 310–317, 2020.

[36] G. Caminero, M. Lopez-Martin, and B. Carro, "Adversarial environment reinforcement learning algorithm for intrusion detection," *Computer Networks*, vol. 159, pp. 96–109, 2019.

[37] Y. Wang, J. Wang, and H. Jin, "Network intrusion detection method based on improved CNN in internet of Things environment," *Mobile Information Systems*, vol. 2022, Article ID 3850582, 10 pages, 2022.

[38] J. Du, K. Yang, Y. Hu, and L. Jiang, "Nids-cnnlstm: network intrusion detection classification model based on deep learning," *IEEE Access*, vol. 11, pp. 24808–24821, 2023.

[39] K. T. Chui, B. B. Gupta, P. Chaurasia, V. Arya, A. Almomani, and W. Alhalabi, "Three-stage data generation algorithm for multiclass network intrusion detection with highly imbalanced dataset," *International Journal of Intelligent Networks*, vol. 4, pp. 202–210, 2023.

[40] Y. Li, Y. Xu, Z. Liu et al., "Robust detection for network intrusion of industrial IoT based on multi-CNN fusion," *Measurement*, vol. 154, Article ID 107450, 2020.

[41] W. Wang, X. Du, D. Shan, R. Qin, and N. Wang, "Cloud intrusion detection method based on stacked contractive auto-encoder and support vector machine," *IEEE Transactions on Cloud Computing*, vol. 10, no. 3, pp. 1634–1646, 2022.

[42] M. A. Ayed and C. Talhi, "Federated learning for anomaly-based intrusion detection," in *Proceedings of the 2021 International Symposium on Networks, Computers and*

Communications (ISNCC), pp. 1–8, IEEE, Dubai, UAE, October, 2021.

[43] H. Alazzam, A. Sharieh, and K. E. Sabri, "A feature selection algorithm for intrusion detection system based on pigeon inspired optimizer," *Expert Systems with Applications*, vol. 148, Article ID 113249, 2020.

[44] X. Ma and W. Shi, "Aesmote: adversarial reinforcement learning with smote for anomaly detection," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 2, pp. 943–956, 2021.