Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2025.063308

ARTICLE

# Real-Time Identification Technology for Encrypted DNS Traffic with Privacy Protection

**Zhipeng Qin[1,2,*], Hanbing Yan[3], Biyang Zhang[2], Peng Wang[2] and Yitao Li[3]**

[1]School of Computer Science and Engineering, Beihang University, Beijing, 100191, China
[2]National Computer Network Emergency Response Technical Team/Coordination Center of China Shanxi Branch, Taiyuan, 030001, China
[3]National Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing, 100029, China

*Corresponding Author: Zhipeng Qin. Email: qinzp@buaa.edu.cn

Received: 10 January 2025; Accepted: 19 March 2025; Published: 19 May 2025

**ABSTRACT:** With the widespread adoption of encrypted Domain Name System (DNS) technologies such as DNS over Hyper Text Transfer Protocol Secure (HTTPS), traditional port and protocol-based traffic analysis methods have become ineffective. Although encrypted DNS enhances user privacy protection, it also provides concealed communication channels for malicious software, compelling detection technologies to shift towards statistical feature-based and machine learning approaches. However, these methods still face challenges in real-time performance and privacy protection. This paper proposes a real-time identification technology for encrypted DNS traffic with privacy protection. Firstly, a hierarchical architecture of cloud-edge-end collaboration is designed, incorporating task offloading strategies to balance privacy protection and identification efficiency. Secondly, a privacy-preserving federated learning mechanism based on Federated Robust Aggregation (FedRA) is proposed, utilizing Medoid aggregation and differential privacy techniques to ensure data privacy and enhance identification accuracy. Finally, an edge offloading strategy based on a dynamic priority scheduling algorithm (DPSA) is designed to alleviate terminal burden and reduce latency. Simulation results demonstrate that the proposed technology significantly improves the accuracy and real-time performance of encrypted DNS traffic identification while protecting privacy, making it suitable for various network environments.

**KEYWORDS:** Encrypted DNS; edge computing; federated learning; real-time detection; privacy protection

## 1 Introduction

With the increasing demand for internet privacy protection, encrypted DNS technologies such as DNS over HTTPS (DoH) and DNS over Transport Layer Security (DoT) have become essential means for ensuring DNS traffic security. Encrypted DNS encapsulates DNS requests within HTTPS or Transport Layer Security (TLS) channels, encrypting DNS query content and effectively preventing traffic monitoring and tampering by hiding transmission ports and protocol characteristics [1,2]. The widespread application of this technology enhances user privacy protection but also introduces new security challenges, particularly in the detection and identification of encrypted DNS traffic [3].

The main challenge in encrypted DNS traffic identification is distinguishing this traffic from other types of encrypted or normal traffic, while also detecting potential malicious activities [4]. Encrypted DNS identification has a wide range of applications, including detecting abnormal communications in enterprise networks, identifying attack traffic in public Wi-Fi environments, and uncovering hidden botnet activities

in cloud services [5]. These applications require precise and efficient detection technologies that balance real-time monitoring with privacy protection.

Current methods for identifying encrypted DNS traffic typically rely on analyzing traffic features, such as packet size, latency, and traffic patterns to differentiate encrypted DNS from other encrypted traffic [6,7]. These methods often employ machine learning or statistical models to classify traffic based on these features. However, they face significant challenges. First, real-time detection requires a balance between high system accuracy and efficiency, but the dynamic nature and increasing diversity of encrypted DNS protocols introduce considerable complexity into model training and reduce their generalizability [8,9]. Second, the need for large-scale data collection to train these models can lead to privacy concerns, as the analysis of user traffic may inadvertently expose sensitive information. Moreover, as encrypted DNS protocols evolve rapidly, many existing models struggle to adapt to new and emerging protocols, further diminishing their effectiveness.

Thus, while current approaches have made progress, they still face limitations in both accuracy and adaptability, and they often fail to fully address privacy issues. Achieving efficient, scalable, and privacy-preserving identification of encrypted DNS traffic remains a critical challenge in this field.

Edge computing and federated learning have been proposed to improve both real-time performance and privacy protection. Edge computing reduces detection latency by offloading computational tasks to nodes closer to the data source, easing the load on central servers and supporting real-time detection [10]. Federated learning enables distributed model training, allowing local model updates at nodes and sharing results without exposing sensitive user data [11]. By combining the low-latency benefits of edge computing with the privacy-preserving properties of federated learning, we propose a novel privacy protection layered detection framework for encrypted DNS traffic identification. The framework combines cloud-edge-end collaboration, Federated learning, and differential privacy technologies to enable both real-time identification and privacy protection. By introducing edge computing, the framework achieves edge offloading of training and detection tasks, reducing latency while protecting user data privacy. This combination of technologies improves the accuracy and efficiency of encrypted DNS traffic identification and avoids the privacy risks that may be present in traditional centralized approaches.

The main contributions of this paper are as follows:

1. **Privacy-Preserving Real-Time Encrypted DNS Traffic Identification Framework:** We propose a cloud-edge-end collaborative framework for encrypted DNS traffic identification. The framework, organized into terminal, edge, and cloud layers, resolves the conflict between privacy protection and real-time detection through edge-end collaboration and task offloading mechanisms.
2. **Privacy-Preserving Hierarchical Federated Learning Training Mechanism:** We introduce a hierarchical Federated Learning mechanism based on FedRA. This mechanism ensures that user-sensitive data are not exposed during model training across multiple edge nodes and the cloud.
3. **Encrypted DNS Identification with Dynamic Priority Edge Offloading:** A dynamic priority scheduling algorithm is introduced to intelligently offload computation tasks to edge nodes. By leveraging edge computing, this approach reduces the computational burden on terminal devices, decreases identification latency, and enhances the real-time efficiency of encrypted DNS traffic identification.

The structure of this paper is as follows. Section 2 presents related technologies and research in the field of encrypted DNS traffic identification and privacy protection. Section 3 describes the proposed privacy-preserving real-time encrypted DNS identification framework in detail. In Section 4, we discuss the privacy-preserving FedRA hierarchical federated learning training mechanism. Section 5 presents the

technology of encrypted DNS identification based on dynamic priority edge offloading. Section 6 outlines the simulation setup and results. Finally, Section 7 concludes the paper and discusses future work.

## 2 Related Technologies and Research

### *2.1 Application of Edge Computing Technology in Traffic Identification*

Edge computing involves shifting computational and storage tasks from central servers to edge devices or nodes located closer to the data source. This reduces data transmission latency and enhances system response times [12]. Given the complexity of encrypted DNS traffic and the need for quick adaptation to changing network environments, edge computing enables fast data preprocessing, feature extraction, and traffic identification, significantly enhancing real-time performance [13].

Edge computing has been widely utilized for traffic identification and real-time monitoring in various studies. For example, Kim et al. proposed a network traffic classification scheme based on deep recurrent neural networks (RNNs) implemented within an edge computing framework, which significantly improved classification accuracy and real-time performance [14]. Song et al. explored an edge computing-enhanced online traffic prediction method for autonomous driving and vehicular networks. Their approach achieved precise predictions of vehicle behavior through real-time traffic data analysis, enhancing traffic management efficiency [15]. Additionally, Modupe et al. designed a cloud/edge computing-based system for network traffic monitoring and threat detection. This system processes preliminary data at edge nodes, enabling rapid responses to network security incidents and ensuring stable network operations [16].

### *2.2 Application of Federated Learning Technology in Privacy Protection*

Federated learning is a distributed machine learning approach that allows multiple participants to collaboratively train a global model by sharing model updates instead of raw data [17,18]. The key advantage of federated learning in this context is that it enables local data processing and model training, thereby eliminating the need to upload raw traffic data to central servers. Distributed training not only improves the generalization of the identification model but also ensures privacy, making federated learning well-suited for large-scale traffic analysis involving sensitive information [19].

Several studies highlight the effectiveness of federated learning in preserving privacy. For example, Mateus et al. proposed a federated learning-based solution for Distributed Denial of Service (DDoS) detection in software-defined networks (SDN), which identified and mitigated DDoS attacks while protecting network data privacy through distributed model training [20]. Doriguzzi-Corin et al. developed FLAD, an adaptive federated learning method for DDoS detection that maintains high accuracy in dynamic network environments while safeguarding user data [21]. de Caldas Filho et al. proposed a federated learning-based botnet detection and mitigation model for IoT networks. This approach enabled effective botnet identification and defense through collaboration among distributed devices, while preserving the privacy of sensitive data on each device [22].

### *2.3 Edge Computing-Assisted Federated Learning Technology*

The integration of edge computing and federated learning offers a powerful solution for efficient encrypted DNS traffic identification while ensuring privacy protection [23]. In this framework, edge computing nodes perform local data processing and traffic analysis, while also contributing to distributed model training. This collaborative approach reduces transmission latency, enhances real-time identification capabilities, and avoids centralized storage and sharing of sensitive data [24].

Research combining edge computing with federated learning has shown promising results across various domains. For instance, Jarwan et al. proposed an edge-based federated deep reinforcement learning approach for IoT traffic management. Their method deployed intelligent agents on edge nodes to achieve efficient traffic scheduling and resource allocation while ensuring data privacy [25]. Liu et al. addressed the issue of privacy-preserving traffic prediction by proposing a federated learning-based model that performs local training and updates on edge devices. This approach achieved high-accuracy predictions while protecting user data privacy [26]. Wang et al. developed a federated semi-supervised learning method for network traffic classification, leveraging edge computing to efficiently classify large-scale network traffic and improve the performance and generalization of classification models, all while preserving data privacy [27].

Compared to traditional centralized processing methods, frameworks that combine edge computing and federated learning demonstrate greater flexibility and adaptability when dealing with large-scale, dynamically changing network environments. This technical architecture can not only cope with the growing volume of encrypted DNS traffic, but also meet the needs of real-time monitoring and rapid response, especially in the case of a high degree of privacy protection, providing a more reliable solution. In addition, references [28,29] have verified the feasibility of edge computing technology and federated learning technology in encrypted DNS traffic identification, and the combination of the two technologies can provide a feasible solution for encrypted DNS traffic identification.

## 3 Privacy-Preserving Real-Time Encrypted DNS Identification Framework

The extensive use of encrypted DNS protocols like DoH and DoT has created major hurdles for traditional port and protocol-based traffic analysis methods. Because communication content is encrypted and encrypted DNS traffic is stealthy, conventional traffic analysis techniques often fail. To tackle these issues, we suggest a Privacy-Preserving Real-Time Encrypted DNS Identification Framework. This framework boosts the accuracy and efficiency of encrypted DNS traffic identification while protecting user privacy. As shown in Fig. 1, it uses a cloud-edge-end collaborative structure and combines federated learning, differential privacy, and Temporal Convolutional Networks (TCN). Through multi-layer task allocation and collaboration, it accomplishes efficient and secure encrypted DNS traffic identification.

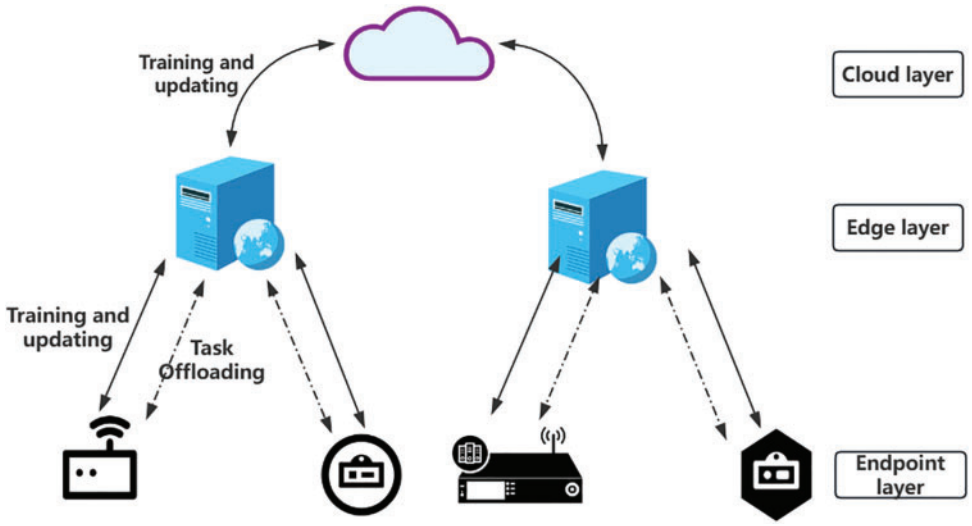


**Figure 1:** Privacy-preserving real-time encrypted DNS identification framework

The framework has three layers: terminal, edge, and cloud, each with specific roles to ensure data privacy and boost identification accuracy.

The terminal layer gathers data, extracts features, and does initial identification. It pulls key features from encrypted DNS traffic, such as IP addresses, port numbers, packet sizes, and DNS queries. Lightweight models like TCN handle preliminary identification. For privacy, sensitive data is encrypted before transmission, and differential privacy is added to model parameters before uploading. Non-sensitive data goes to the edge layer for more processing.

The edge layer manages computation offloading, model aggregation, and training. It dynamically adjusts offloading strategies based on resources and network conditions. Using federated learning, it aggregates encrypted model parameters from terminals to ensure privacy. The edge layer also optimizes resource use and cuts latency.

The cloud layer gets encrypted model parameters from multiple edge nodes and aggregates them into a global model via federated learning. This process only uses encrypted parameters, so raw traffic data isn't exposed. The cloud layer then sends the global model back to edge nodes, improving the model's performance, identification accuracy, and generalization.

In the federated learning setup, data is split using horizontal partitioning. Devices train the model locally and only share model updates with the edge or cloud layers, not raw data. This keeps user data private while allowing collaborative global model training.

As shown in Fig. 2, the training process follows a multi-step procedure. Terminal devices collect and extract features from encrypted DNS traffic, keeping sensitive features local. Non-sensitive features are sent to the edge layer, where federated learning aggregates model parameters. These encrypted parameters are uploaded to the cloud, where a global model is created. This model is then sent back to the terminal and edge layers for improved identification performance.

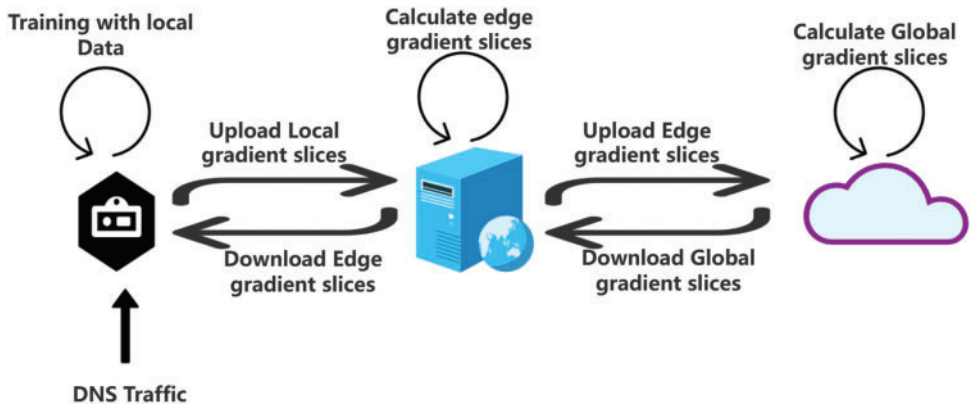


**Figure 2:** The training process of encrypted DNS within the cloud-edge-end collaborative architecture

During real-time encrypted DNS traffic identification, terminal devices extract features from traffic data and perform preliminary identification using local models. For less complex tasks, terminal devices directly use local models for inference. For computationally intensive tasks, some of non-sensitive workload is offloaded to edge nodes.

## 4 Privacy-Preserving FedRA Hierarchical Federated Learning Training Mechanism

Within the cloud-edge-end architecture, achieving high identification accuracy for encrypted DNS traffic recognition while ensuring user privacy through a distributed training framework is a pressing issue. To address this, this chapter introduces a Privacy-Preserving FedRA Hierarchical Federated Learning Training Mechanism. This method enhances model updates and improves identification accuracy through a hierarchical aggregation mechanism. By incorporating differential privacy techniques, it enables efficient encrypted DNS traffic identification without sharing raw data, effectively safeguarding user privacy.

### *4.1 Data Input and Standardization Processing*

Terminal devices generate raw data by capturing DNS requests and responses, which include multiple features such as timestamps, request types, response types, request sizes, response sizes, and Time to Live (TTL). Since different features have varying units and ranges, directly inputting raw data into the model may lead to instability during training and slow convergence. Therefore, data standardization becomes a necessary preprocessing step.

The goal of standardization is to adjust each feature to the same scale, achieving zero mean and unit variance. The specific standardization formula is as follows:

$$x'_{i,j} = \frac{x_{i,j} - \mu_j}{\sigma_j} \tag{1}$$

where $x_{i,j}$ represents the value of the $j$-th feature for the $i$-th sample, $\mu_j$ and $\sigma_j$ are the mean and standard deviation of the $j$-th feature across all samples, respectively, and $x'_{i,j}$ is the standardized feature value. The formulas for calculating the mean and standard deviation are:

$$\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{i,j}, \quad \sigma_j = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(x_{i,j} - \mu_j\right)^2} \tag{2}$$

where $N$ is the number of samples. Through standardization, the feature matrix $\mathbf{X} \in \mathbb{R}^{T\times D}$ (where $T$ is the time step and $D$ is the number of features) generated by terminal devices eliminates dimensional discrepancies, better accommodating subsequent model training processes. This ensures that the model can effectively capture the temporal characteristics and patterns within encrypted DNS traffic.

### *4.2 Training Method for Encrypted DNS Identification Model Based on FedRA Federated Learning Algorithm*

Encrypted DNS traffic varies widely due to diverse network environments, device types, and user behaviors. This variability challenges traditional single-model training methods in effectively capturing its features. Although TCN can model traffic data's temporal characteristics to improve identification accuracy and robustness, single terminal devices often have limited DNS traffic data, struggling to support traditional machine learning model training, especially with significant data distribution differences.

To solve this, federated learning enables multiple devices to jointly train models without raw data sharing, fully utilizing the heterogeneous data across devices. The proposed FedRA algorithm uses the Medoid aggregation strategy to select the most representative local model updates, reducing the impact of abnormal or outlier updates on the global model. Unlike FedAvg, FedRA enhances robustness by lessening the effect of malicious or faulty updates, which is crucial in highly variable data environments. Moreover, FedRA, combined with differential privacy, ensures data privacy without raw data sharing. Below is the specific application of the FedRA algorithm in encrypted DNS recognition.

In addition to FedRA, we also conduct comparative experiments with other robust aggregation methods, such as Krum and Trimmed Mean, to validate the advantages of our approach in terms of model robustness and accuracy. These experiments demonstrate that FedRA consistently outperforms other methods, particularly in scenarios with noisy or incomplete data.

FedRA is employed at both the edge layer and the cloud layer in federated learning to ensure the accuracy and privacy of the global model. The specific process is as follows.

*4.2.1 Terminal Layer Training*

At the Terminal Layer, we use incremental learning to process the local data of the terminal device. Different from traditional batch learning, incremental learning does not need to load the entire data set at once, but processes the data step by step to reduce calculation and storage costs. Each terminal device performs incremental training locally, using only its local data and uploading updated model parameters, which both protects privacy and improves training efficiency. The incremental training process includes the following steps:

1. **Local Data Standardization:** Terminal devices standardize the received encrypted DNS traffic data, generating the standardized feature matrix $\mathbf{X}'$.
2. **Model Forward Propagation and Loss Calculation:** Based on the standardized data, terminal devices perform forward propagation using the current model parameters $\theta_t$, calculating the predicted values $\hat{Y}$ and the loss function $\mathscr{L}$:

$$\mathscr{L}(\theta_t) = \frac{1}{M}\sum_{i=1}^{M}\mathscr{L}(\hat{y}_i, y_i) \tag{3}$$

   where $M$ is the number of samples in the local dataset, $\hat{y}_i$ is the model's prediction for the $i$-th sample, and $y_i$ is the actual label.
3. **Gradient Calculation and Privacy Protection:** Compute the gradient of the loss function with respect to the model parameters $\nabla L_{\text{local}}(\theta_t)$, and add Gaussian noise to the gradient to achieve differential privacy protection:

$$\widehat{\nabla L_{\text{local}}}(\theta_t) = \nabla L_{\text{local}}(\theta_t) + \mathscr{N}(0, \sigma^2 I) \tag{4}$$

   where $\mathscr{N}(0, \sigma^2 I)$ represents Gaussian noise with mean 0 and variance $\sigma^2$, and $I$ is the identity matrix. The standard deviation $\sigma$ controls the strength of privacy protection.
4. **Uploading Noisy Gradient:** Terminal devices upload the noisy gradient $\widehat{\nabla L_{\text{local}}}(\theta_t)$ to the edge server, ensuring participation in global model updates without exposing raw data.

*4.2.2 Edge Layer Aggregation*

The edge layer is responsible for receiving noisy gradients from multiple terminal devices and aggregating them using the FedRA algorithm. The specific steps are as follows:

1. **Receiving Local Updates:** The edge server receives noisy local gradient updates from $N$ terminal devices:

$$\left\{\widehat{\nabla L_{\text{local},i}}(\theta)\right\}_{i=1}^{N} \tag{5}$$

   where $\widehat{\nabla L_{\text{local},i}}(\theta)$ represents the noisy gradient uploaded by the $i$-th terminal device.

2. **Calculating Gradient Distances:** The edge server calculates the Euclidean distance between each pair of local updates:

$$d\left(\widehat{\nabla L_{\text{local},i}}(\theta), \widehat{\nabla L_{\text{local},j}}(\theta)\right) = \left|\widehat{\nabla L_{\text{local},i}}(\theta) - \widehat{\nabla L_{\text{local},j}}(\theta)\right| \tag{6}$$

where $d$ represents the distance metric function.

3. **Selecting Medoid Updates:** Using the Medoid aggregation strategy, the edge server selects the most representative local update as the aggregation result. Specifically, the edge server selects a local update $\nabla L_{\text{medoid}}(\theta)$ that minimizes the total distance to all other local updates:

$$\nabla L_{\text{medoid}}(\theta) = \arg\min_{i} \sum_{\substack{j=1 \\ j\neq i}}^{N} d\left(\widehat{\nabla L_{\text{local},i}}(\theta), \widehat{\nabla L_{\text{local},j}}(\theta)\right) \tag{7}$$

This strategy effectively filters out anomalous or deviating gradient updates, ensuring that the aggregation result is more representative and robust.

4. **Adding Differential Privacy Noise:** Further add Gaussian noise to the Medoid aggregation result to enhance privacy protection:

$$\nabla L'_{\text{medoid}}(\theta) = \nabla L_{\text{medoid}}(\theta) + \mathcal{N}(0, \sigma^2_{\text{edge}}) \tag{8}$$

where $\sigma_{\text{edge}}$ is the standard deviation of the noise added at the edge layer, controlling the strength of privacy protection.

5. **Uploading Aggregated Results:** The edge server uploads the noisy Medoid update $\nabla L'_{\text{medoid}}(\theta)$ to the cloud layer, serving as the basis for global model updates.

*4.2.3 Cloud Layer Aggregation and Update*

The cloud layer is responsible for receiving aggregated updates from multiple edge layers and further integrating them using the FedRA algorithm to maintain the global model. The specific steps are as follows:

1. **Receiving Aggregated Updates:** The cloud server receives aggregated updates from $M$ edge servers:

$$\left\{\nabla L'_{\text{medoid}}(\theta)\right\}_{m=1}^{M} \tag{9}$$

2. **Calculating Global Update:** The cloud server computes the weighted average of all received aggregated updates to generate the global model update:

$$\nabla L_{\text{global}}(\theta) = \frac{1}{M} \sum_{m=1}^{M} \nabla L'_{\text{medoid}}(\theta)_m \tag{10}$$

3. **Model Update:** The cloud server updates the global model parameters $\theta$ using the global update $\nabla L_{\text{global}}(\theta)$ and distributes the new model parameters to all edge servers and terminal devices, completing one federated learning training cycle.

The FedRA mechanism addresses this by using the Medoid aggregation strategy, which selects the most representative local updates. This approach better accommodates variations in data distribution and effectively filters out anomalous or malicious gradient updates. As a result, it enhances the stability and accuracy of the global model, which is crucial for detecting abnormal traffic and malicious requests within encrypted DNS traffic.

At each level of aggregation, FedRA integrates differential privacy techniques by adding noise to local updates and aggregation results. This ensures that the privacy of individual terminal device data is preserved.

### *4.3 Privacy Protection Mechanism for Encrypted DNS Identification Based on Differential Privacy*

In the FedRA algorithm, Differential Privacy (DP) techniques are integrated into every aggregation step at the terminal layer, edge layer, and cloud layer, ensuring the privacy of data throughout the model training process. Differential privacy aims to make the model's output independent of any single user's data by adding noise, thereby preventing the leakage of sensitive information. Differential privacy is quantified using a privacy budget $(\varepsilon, \delta)$, which measures the strength of privacy protection:

$$\Pr\left[\mathscr{M}(D) \in S\right] \leq e^{\varepsilon} \Pr\left[\mathscr{M}(D') \in S\right] + \delta \tag{11}$$

where $\mathscr{M}$ represents the model, $D$ and $D'$ are two adjacent datasets, $S$ is the set of possible outputs of the model, $\varepsilon$ controls the risk of privacy leakage, and $\delta$ is a small probability that allows the model output to violate the inequality in extreme cases.

Terminal devices add Gaussian noise to the gradient updates after local training to achieve differential privacy protection:

$$\widehat{\nabla L_{\text{local}}}(\theta_t) = \nabla L_{\text{local}}(\theta_t) + \mathscr{N}(0, \sigma^2 I) \tag{12}$$

where $\mathscr{N}(0, \sigma^2 I)$ represents Gaussian noise with mean 0 and variance $\sigma^2$, and $I$ is the identity matrix. The standard deviation $\sigma$ is adjusted based on the required privacy budget $\varepsilon$ and $\delta$ to ensure the desired level of privacy protection.

At the edge layer, after selecting the Medoid aggregation result, additional Gaussian noise is added to further enhance privacy protection:

$$\nabla L'_{\text{medoid}}(\theta) = \nabla L_{\text{medoid}}(\theta) + \mathscr{N}(0, \sigma^2_{\text{edge}}) \tag{13}$$

where $\sigma_{\text{edge}}$ is the standard deviation of the noise added at the edge layer, controlling the strength of privacy protection. This approach ensures that even if an edge server is compromised during the aggregation process, it cannot recover individual terminal devices' private data from the aggregated results.

Upon receiving the aggregated updates from the edge layer, the cloud server continues to apply differential privacy techniques to further protect the global model's privacy:

$$\nabla L'_{\text{global}}(\theta) = \nabla L_{\text{global}}(\theta) + \mathscr{N}(0, \sigma^2_{\text{cloud}}) \tag{14}$$

where $\sigma_{\text{cloud}}$ is the standard deviation of the noise added at the cloud layer, ensuring that the global model update process does not leak any sensitive information from terminal devices.

The choice of privacy budget $(\varepsilon, \delta)$ directly affects the overall performance of encrypted DNS traffic identification. Specifically, $\varepsilon$ will increase the noise added to the gradient, and a smaller $\varepsilon$ will affect the training speed and accuracy of the model; Larger ones may reduce the ability to protect privacy. Therefore, the choice of privacy budget should consider the trade-off between utility and privacy protection. In this study, we propose a hierarchical privacy budget allocation method to reasonably manage the privacy budget of each layer.

$$\varepsilon_{\text{total}} = \varepsilon_{\text{terminal}} + \varepsilon_{\text{edge}} + \varepsilon_{\text{cloud}} \tag{15}$$

where $\varepsilon_{\text{total}}$ is the overall privacy budget, and $\varepsilon_{\text{terminal}}$, $\varepsilon_{\text{edge}}$, and $\varepsilon_{\text{cloud}}$ are the privacy budgets allocated to the terminal layer, edge layer, and cloud layer. A higher $\varepsilon_{\text{terminal}}$ ensures that the terminal layer retains higher privacy protection while improving the accuracy of local model updates. However, excessive $\varepsilon_{\text{edge}}$ and $\varepsilon_{\text{cloud}}$ can lead to the risk of privacy breaches in the aggregation step, so different levels of privacy need to be

balanced. The terminal layer typically handles the most sensitive user data and therefore allocates a higher privacy budget; The privacy budget of the edge layer and cloud layer is appropriately reduced to balance privacy protection and computing efficiency. In practice, the privacy budget can be adjusted according to specific scenarios.

## 5 Encrypted DNS Identification Technology Based on Dynamic Priority Edge Offloading

The challenge of encrypted DNS traffic identification lies in its high computational load and low latency requirements. When terminals employ complex algorithms such as machine learning or operate in high-frequency identification scenarios, the computational capabilities of terminal devices are often insufficient to meet real-time demands. Therefore, this section proposes an edge offloading strategy based on the Dynamic Priority Scheduling Algorithm (DPSA). The goal is to optimize task offloading decisions to reduce the computational burden on terminal devices, decrease latency, and ensure identification accuracy.

### *5.1 Edge Offloading Task Model*

The overall latency $D_{\text{total}}$ of encrypted DNS identification tasks consists of data transmission latency $D_{\text{trans}}$, edge computing latency $D_{\text{edge}}$, and local computation latency $D_{\text{local}}$:

$$D_{\text{total}} = D_{\text{trans}} + D_{\text{edge}} + D_{\text{local}} \tag{16}$$

where $D_{\text{trans}}$ refers to the time required for terminal devices to upload feature data to edge nodes. Specifically, $D_{\text{trans}}$ is calculated as:

$$D_{\text{trans}} = \frac{D_{\text{task}}}{B_{\text{net}}} + D_{\text{net}} \tag{17}$$

Here, $D_{\text{task}}$ represents the size of the task data, $B_{\text{net}}$ is the network bandwidth, and $D_{\text{net}}$ is the network latency. $D_{\text{edge}}$ is the time required for edge nodes to perform model inference on the uploaded feature data. It is given by:

$$D_{\text{edge}} = \frac{T_{\text{task}}}{C_{\text{edge}}} \tag{18}$$

where $T_{\text{task}}$ denotes the computational complexity of the task, and $C_{\text{edge}}$ is the computational capability of the edge node. $D_{\text{local}}$ is the time required for terminal devices to perform feature extraction and model inference locally:

$$D_{\text{local}} = \frac{T_{\text{task}}}{C_{\text{local}}} \tag{19}$$

where $C_{\text{local}}$ represents the computational capability of the terminal device. In encrypted DNS traffic identification, the key to edge offloading strategies is the dynamic decision of whether to offload feature extraction or model inference tasks to edge nodes. The objective of this strategy is to minimize the total task latency while balancing the trade-off between latency and identification accuracy. To optimize offloading decisions, we model it as an optimization problem with the following mathematical formulation:

$$\min \sum_{i=1}^{N} \sum_{j=1}^{M_i} \left[ x_{ij} \cdot \left( D_{\text{trans},ij} + D_{\text{edge},ij} \right) + \left( 1 - x_{ij} \right) \cdot D_{\text{local},ij} \right] \tag{20}$$

where $x_{ij}$ is the decision variable, with $x_{ij} = 1$ indicating that task $j$ of terminal $i$ is offloaded to the edge node. The model constraints are as follows:

**Edge Node Load Constraint:** The computational load on edge nodes must not exceed their maximum load $L_{\max}$:

$$\sum_{i=1}^{N}\sum_{j=1}^{M_i} x_{ij} \cdot \frac{T_{\text{task},ij}}{C_{\text{edge}}} \le L_{\max} \tag{21}$$

**Latency Requirement Constraint:** The total latency of each task must be less than its maximum tolerated latency $D_{\max,ij}$:

$$x_{ij} \cdot \left(D_{\text{trans},ij} + D_{\text{edge},ij}\right) + \left(1 - x_{ij}\right) \cdot D_{\text{local},ij} \le D_{\max,ij}, \quad \forall i, j \tag{22}$$

### *5.2 Edge Offloading Strategy Based on Dynamic Priority Scheduling Algorithm*

To optimize task offloading decisions in encrypted DNS traffic identification, this paper proposes an edge offloading strategy based on the Dynamic Priority Scheduling Algorithm (DPSA). This strategy dynamically schedules and optimizes task offloading schemes to ensure minimal system latency. The specific strategy is as follows:

1. **Assigning Priority to Each Task:** Based on the task's latency tolerance $D_{\max,ij}$ and computational complexity $T_{\text{task},ij}$, assign a priority to each task:

$$\text{priority}_{ij} = \left(\frac{D_{\max,ij}}{T_{\text{task},ij}}\right) \times \alpha + \left(\frac{1}{D_{\text{network},ij}}\right) \times \beta \tag{23}$$

   where $\alpha$ and $\beta$ are weight coefficients used to balance the impact of latency tolerance and network conditions on priority. The values of the two parameters can be adjusted based on different scenarios and task requirements. For example, applications with high real-time requirements can increase the weight of $\alpha$, and in environments with unstable network conditions, tasks with better network conditions can be prioritized by increasing the $\beta$ value.

2. **Calculating Local and Edge Latencies:**

$$D_{\text{local},ij} = \frac{T_{\text{task},ij}}{C_{\text{local}}} \tag{24}$$

$$D_{\text{edge},ij} = \frac{D_{\text{task},ij}}{B_{\text{net}}} + D_{\text{net}} + \frac{T_{\text{task},ij}}{C_{\text{edge}}} \tag{25}$$

3. **Offloading Condition Judgment:** Let $\theta$ be the priority threshold. If the latency of offloading the task to the edge node $D_{\text{edge},ij}$ is less than the local processing latency $D_{\text{local},ij}$, the edge node's load $L_{\text{edge}}$ does not exceed its maximum load $L_{\max}$, and the task's priority $\text{priority}_{ij}$ exceeds a preset threshold $\theta$, then the task is offloaded to the edge node:

$$x_{ij} = \begin{cases} 1, & \text{if } D_{\text{edge},ij} < D_{\text{local},ij} \text{ and } L_{\text{edge}} + \frac{T_{\text{task},ij}}{C_{\text{edge}}} \le L_{\max} \text{ and } \text{priority}_{ij} > \theta \\ 0, & \text{otherwise} \end{cases} \tag{26}$$

4. **Updating Edge Node Load:**

$$L_{\text{edge}} \leftarrow L_{\text{edge}} + \frac{T_{\text{task},ij}}{C_{\text{edge}}} \quad \text{if } x_{ij} = 1 \tag{27}$$

5. **Dynamically Adjusting Load Threshold:** Adjust the load threshold $L_{\max}(t)$ based on the current edge node load and a load ratio coefficient $\gamma$:

$$L_{\max}(t) = \gamma \cdot C_{\text{edge}} \tag{28}$$

Through the above design and optimization, DPSA can dynamically adjust task offloading decisions based on the task's latency tolerance, computational complexity, and network conditions. This enables a low-latency and low-load offloading strategy for encrypted DNS traffic identification tasks, meeting the requirements for real-time performance and stability.

### *5.3 Edge-Endpoint Collaborative Real-Time Encrypted DNS Identification Strategy*

After implementing the edge offloading strategy, the feature extraction process for tasks can be divided into two parts: terminal devices are responsible for extracting privacy-sensitive features, while edge nodes handle non-sensitive intermediate features. Through this collaborative mechanism, sensitive data are retained on terminal devices to prevent leakage, while low-privacy-risk features are uploaded to edge nodes for further processing.

$$f_{\text{total}} = f_{\text{local}} + f_{\text{edge}} \tag{29}$$

where $f_{\text{local}}$ represents the local sensitive features extracted by terminal devices, and $f_{\text{edge}}$ represents the non-sensitive intermediate features extracted and uploaded to edge nodes by terminal devices. Terminal devices use the DPSA offloading strategy to offload $f_{\text{edge}}$ to edge nodes. Edge nodes employ Principal Component Analysis (PCA) to process the uploaded features, reducing redundancy and enhancing feature representation capabilities. PCA is applied to the feature set received from terminal devices in order to identify the most informative dimensions and discard those that contribute little to the model's predictive power:

$$f_{\text{enhance}} = \text{PCA}(f_{\text{edge}}) \tag{30}$$

Terminal devices receive the processed results $f_{\text{enhance}}$ from edge nodes and combine them with locally extracted sensitive features $f_{\text{local}}$ for model inference:

$$f_{\text{final}} = f_{\text{enhance}} \oplus f_{\text{local}} \tag{31}$$

In the edge computing architecture, model inference tasks are divided into two parts: the first $k$ layers are processed by terminal devices, and the subsequent layers are handled by edge nodes. This effectively reduces the computational burden on terminal devices while fully leveraging the computational capabilities of edge nodes. The specific hierarchical processing procedure is as follows: Edge nodes first perform convolutional computations on the input features for the first $k$ layers, obtaining intermediate results $h_{\text{edge}}^{(k)}$. The output of each layer undergoes a nonlinear transformation through the activation function $\sigma(\cdot)$. The computation formulas for the first $k$ layers are:

$$h_{\text{local}}^{(k)} = \sigma\left(W_{\text{local}}^{(k)} * f_{\text{final}} + b_{\text{local}}^{(k)}\right) \tag{32}$$

where $W_{\text{local}}^{(k)}$ and $b_{\text{local}}^{(k)}$ are the convolutional kernels and biases of the $k$-th layer, $*$ denotes the convolution operation, and $\sigma$ is the ReLU activation function.

After processing the first $k$ layers, terminal devices upload the intermediate results $h_{\text{edge}}^{(k)}$ to edge nodes and continue processing the remaining $L - k$ layers. The terminal devices combine the intermediate

results returned by edge nodes with their local features to perform inference in the subsequent layers. The computation formulas for the remaining layers are as follows:

$$h_{\text{edge}}^{(k+1)} = \sigma\left(W_{\text{edge}}^{(k+1)} * h_{\text{local}}^{(k)} + b_{\text{edge}}^{(k+1)}\right) \quad \text{my} = \text{softmax}\left(W_{\text{output}} * h_{\text{final}} + b_{\text{output}}\right) \tag{33}$$

where $W_{\text{output}}$ and $b_{\text{output}}$ are the weights and biases of the output layer, and softmax$(\cdot)$ denotes the softmax function. The output $y$ represents the classification label for encrypted DNS traffic.

Under different load conditions, the effect of edge unloading strategy will vary. When the load is light, tasks are evenly distributed, computing resources of terminal devices and edge nodes are fully utilized, and edge nodes can quickly respond to tasks, improving system delay performance. However, when the load is high, fluctuation in network bandwidth and a shortage of edge node computing resources can cause an increased delay in uninstallation tasks. In this case, DPSA adjusts the task uninstallation decision to optimize the network status and load of the node, avoid the overload of the edge nodes, and improve system stability. At the same time, the separation strategy of sensitive and nonsensitive data still plays an important role in the case of high load. Terminal devices process simple local feature extraction, and complex non-sensitive features are unloaded to edge nodes for processing, reducing the burden on terminal devices and improving the overall system processing efficiency.

By combining the Dynamic Priority Edge Offloading Algorithm with a collaborative edge-end phased identification strategy, this chapter effectively optimizes task offloading decisions, reduces the computational burden on terminal devices, and improves overall system latency. Additionally, the collaborative privacy protection mechanism ensures the efficiency and security of encrypted DNS traffic identification, providing reliable technical support for real-time traffic analysis and security protection.

## 6 Simulation

### *6.1 Dataset and Simulation Environment*

The DNS traffic data set published by the Canadian Internet Registration Authority (CIRA), CIRA-CIC-DohBRW-2020, was used as the basis of the experiment. The dataset comprises 269,643 DoH traffic samples and 897,493 non-DoH traffic samples. Among the DoH traffic, 18,807 samples are benign DoH traffic, and 249,836 samples are malicious DoH traffic. The original dataset contains 34 feature dimensions, primarily including source IP, destination IP, port numbers, packet sizes.

In the data preprocessing phase, we divide the data set into a training set and a test set. Specifically, the training set contains about 200,000 data samples, while the test set contains 50,000 data samples. Considering that the number of samples of malicious DoH traffic in the data set is much larger than that of benign DoH traffic, we adopt oversampling to deal with class imbalance in the training process. The over-sampling strategy increases the number of samples of benign DoH traffic, which improves the model's ability to identify minority classes.

It should be noted that the CIRA-CIC-DoHBrw-2020 dataset is derived from encrypted DNS traffic in real network environments and is mainly used to study the detection of DoH traffic, but because its samples are mainly from specific network environments and attack modes, it may not fully represent all possible encrypted DNS traffic scenarios. Therefore, in practical applications, the results of this dataset may be somewhat limited, and future studies can further verify the effectiveness of our method by extending more diverse traffic scenarios and attack modes.

The experimental environment is based on Python 3.8 programming language and leverages the TensorFlow 2.0 deep learning framework for model training and inference. During the data preprocessing stage, the Pandas library is used for data cleaning and feature selection, while Matplotlib and Seaborn

libraries are employed for result visualization and model performance evaluation. The FedRA federated learning algorithm, based on the Medoid aggregation strategy, is combined with TCN for feature learning. The model parameters are configured as follows: the number of TCN convolutional kernels is set to 128, the convolutional kernel size is 3, the number of residual layers is 2, the number of stacked residual layers per layer is 4, and the dropout rate is set to 0.2. As a counterpart to the FedAvg aggregation algorithm, FedRA utilizes the Medoid strategy to select the most representative local model updates, enhancing the robustness and accuracy of the global model.

### *6.2 Experimental Comparison*

To comprehensively evaluate the performance of the proposed Privacy-Preserving Real-Time Encrypted DNS Identification Framework, this section conducts several comparative experiments covering model identification effectiveness, federated learning training modes, and recognition speed under edge computing assistance.

The experiments compare FedRA-TCN with several other encrypted DNS identification algorithms, including FedRA-LSTM, Extreme Gradient Boosting (XGBoost), and Support Vector Machine (SVM). Notably, FedRA-LSTM differs from the proposed algorithm in that it employs Long Short-Term Memory (LSTM) networks instead of TCN within the federated learning framework. The evaluation metrics for each algorithm include Accuracy, Precision, Recall, F1-Score, ROC-AUC, and PR-AUC.

Fig. 3 presents the confusion matrices for different algorithms, including FedRA-TCN, FedRA-LSTM, SVM, and XGBoost, in the task of classifying DoH and non-DoH traffic. The confusion matrix highlights the True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN) for each algorithm. FedRA-TCN shows the highest TP, meaning it accurately identifies the most DoH traffic, and the lowest FN, indicating it misclassifies the fewest DoH packets as non-DoH. This demonstrates that FedRA-TCN outperforms other models in recognizing DoH traffic with high accuracy and strong differentiation capability. In contrast, SVM and XGBoost have fewer TP and TN, reflecting their limited ability to classify encrypted traffic. These traditional machine learning models lack the deep feature learning capability needed for effective classification of encrypted DNS traffic, resulting in lower performance compared to deep learning-based models like FedRA-TCN.

From the metrics shown in Fig. 4, FedRA-TCN consistently outperforms all other algorithms across key evaluation metrics. It achieves the highest accuracy (93.13%), precision (93.96%), recall (93.26%), F1-score (93.61%), ROC-AUC (93.72%), and PR-AUC (93.57%), highlighting its superior ability to correctly identify DoH and non-DoH traffic with a well-balanced trade-off between precision and recall. FedRA-LSTM follows closely in performance, with strong scores in all metrics, though slightly behind FedRA-TCN in accuracy and other key indicators. This indicates that FedRA-TCN is the most reliable model in terms of classification accuracy, minimizing both false positives and false negatives.

In comparison, traditional models like SVM and XGBoost demonstrate lower performance across all metrics, with accuracy scores of 91.09% and 91.01%, respectively. Their precision and recall are also lower than those of the deep learning models, showing their struggle to effectively distinguish between DoH and non-DoH traffic. These models are limited by their inability to capture the complex patterns in encrypted traffic, resulting in higher misclassification rates and less reliable performance overall. Therefore, FedRA-TCN emerges as the most robust solution for encrypted DNS traffic classification, with FedRA-LSTM also showing strong but slightly less competitive results.

Additionally, to validate the acceleration effect of edge computing on federated learning, this paper designs a FedRA cloud-edge-end collaborative learning mode and compares it with traditional cloud-edge

federated learning. The simulation configuration for cloud-edge federated learning is consistent with the parameters of FedRA's cloud and terminal layers. The number of training rounds $R$ is set to 50.

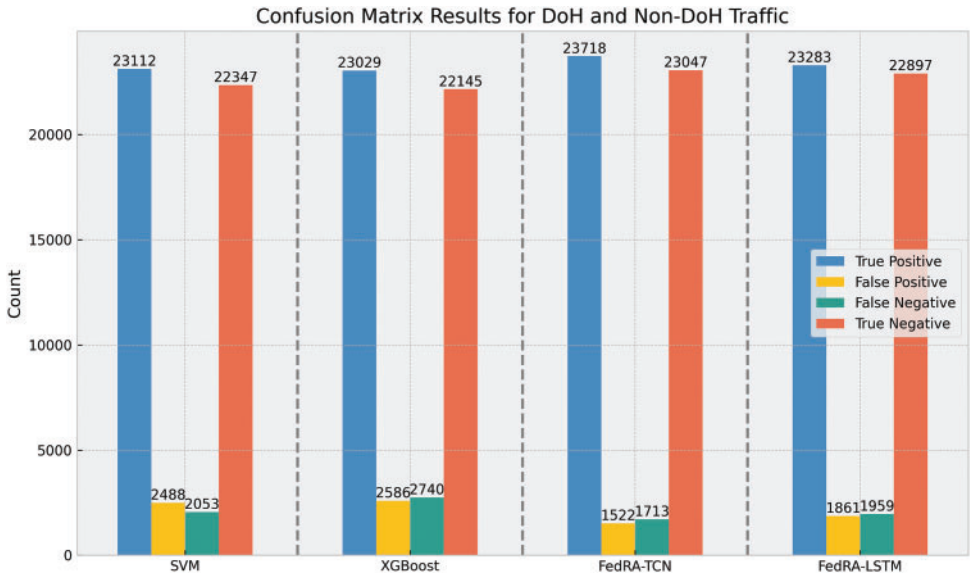


**Figure 3:** Confusion matrices of different algorithms for DoH and non-DoH traffic classification tasks, including the number of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN)

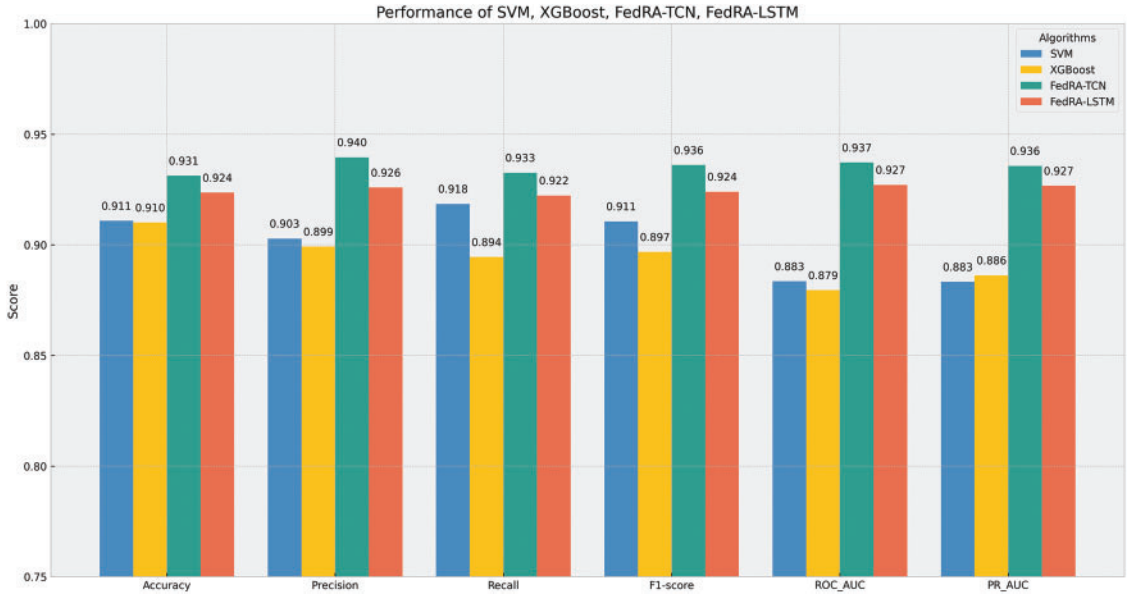


**Figure 4:** Performance evaluation metrics of different algorithms, including accuracy, precision, recall, and F1-score

In Fig. 5, the overall latency of the cloud-edge-terminal collaborative training mode is 183 s, compared to 247 s for the cloud-terminal training mode, reducing latency by 64 s. This demonstrates that introducing the terminal layer for collaborative training significantly enhances overall training efficiency. In terms of communication latency, the cloud-edge-end mode has a communication latency of 77 s, much lower than the

cloud-edge mode's 151 s, reducing latency by 74 s. This is primarily due to terminal devices only needing to upload local model updates rather than raw data or complete model parameters, reducing data transmission volume and effectively alleviating network bandwidth pressure, especially in bandwidth-constrained environments, thereby significantly improving communication efficiency. Although the cloud-edge-end mode's computation latency is slightly higher at 113 s compared to 104 s for the cloud-edge mode, this increase is acceptable because the terminal devices undertake more computational tasks, and the overall latency reduction compensates for this increment.

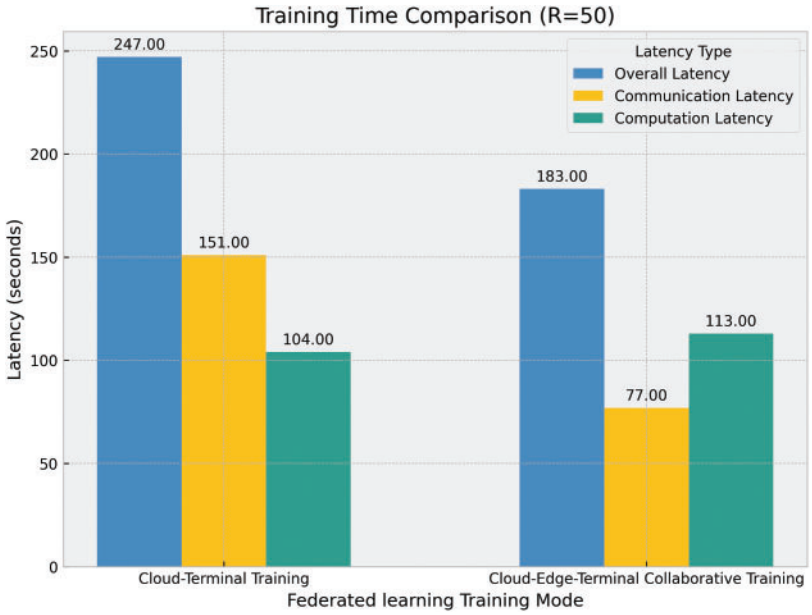


**Figure 5:** Training time comparison

Furthermore, to further evaluate the acceleration effect of edge computing in real-time encrypted DNS traffic identification, this experiment compares the latency of edge computing-assisted recognition with pure local recognition under different load conditions. The evaluation metric is recognition latency (unit: milliseconds), i.e., the time from receiving an encrypted DNS packet to completing classification.

In Fig. 6, latency comparison between edge computing-assisted recognition and pure local recognition under varying CPU load conditions. When CPU load is below 44%, local recognition latency is lower than edge computing-assisted recognition, indicating that edge offloading strategies are unnecessary. However, as terminal load increases (higher CPU utilization), the latency growth of edge computing-assisted recognition is minimal, whereas pure local recognition latency increases significantly. In high-load scenarios, edge computing-assisted encrypted DNS recognition significantly outperforms local recognition in terms of latency, demonstrating that edge offloading effectively reduces recognition latency on terminal devices and enhances the overall system's real-time performance.

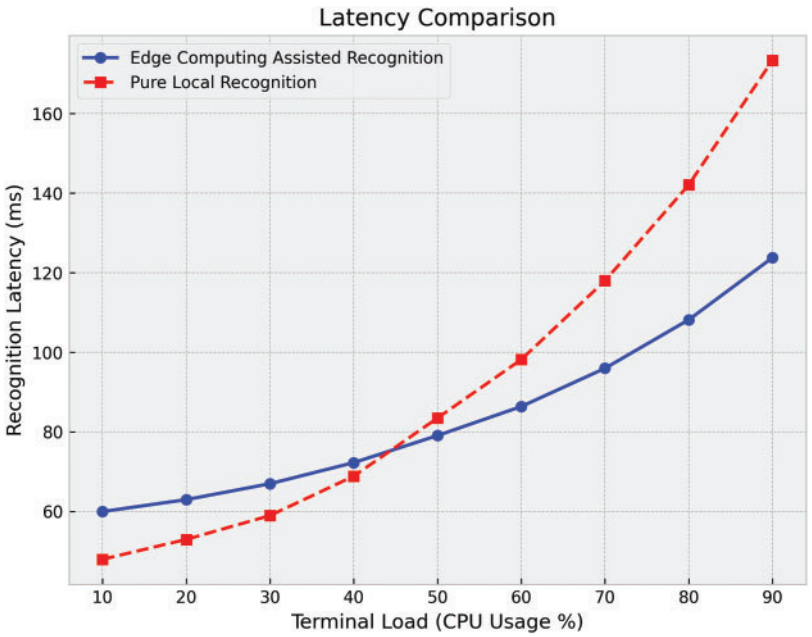


**Figure 6:** Latency comparison

## 7 Conclusion

This paper proposes a Privacy-Preserving Real-Time Encrypted DNS Identification Framework that significantly improves the accuracy and efficiency of encrypted DNS traffic identification through multi-layer collaboration among terminal, edge, and cloud layers. By integrating federated learning and differential privacy technologies, the framework effectively handles encrypted DNS traffic while avoiding the privacy leakage issues inherent in traditional methods. Simulation results demonstrate that the proposed FedRA-TCN model outperforms traditional algorithms in multiple performance metrics, and the intelligent offloading strategy significantly enhances the real-time identification capability of encrypted DNS traffic. Additionally, the application of edge computing effectively reduces system latency and improves model update efficiency, providing a more efficient and flexible solution for encrypted DNS traffic monitoring.

**Acknowledgement:** Not applicable.

**Funding Statement:** This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

**Author Contributions:** Zhipeng Qin and Hanbing Yan conceived and designed the research. Biyang Zhang and Peng Wang performed the experiments and analyzed the data. Yitao Li contributed to writing and revising the paper. All authors reviewed the results and approved the final version of the manuscript.

**Availability of Data and Materials:** The datasets generated and/or analyzed during the current study are not publicly available due to personal privacy reasons but are available from the corresponding author on reasonable request.

**Ethics Approval:** This study complies with the ethical standards of the institution and the National Research Council.

**Conflicts of Interest:** The authors declare no conflicts of interest to report regarding the present study.

## References


1. Csikor L, Singh H, Kang MS, Divakaran DM. Privacy of DNS-over-HTTPS: requiem for a dream?. In: 2021 IEEE European Symposium on Security and Privacy (EuroS&P); 2021; IEEE. p. 252–71.
2. Siby S, Juarez M, Diaz C, Vallina-Rodriguez N, Troncoso C. Encrypted DNS–> privacy? A traffic analysis perspective. arXiv:1906.09682. 2019.
3. Kambourakis G, Karopoulos G. DNS: the good, the bad and the moot. Comput Fraud Secur. 2022;2022(5):2–20.
4. Hoffman PE. DNS security extensions (DNSSEC). RFC 9364. 2023 [cited 2025 Feb 10]. Available from: https://docs.microsoft.com/zh-cn/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/ee683904(v=ws.10).
5. Lyu M, Gharakheili HH, Sivaraman V. A survey on DNS encryption: current development, malware misuse, and inference techniques. ACM Comput Surv. 2022;55(8):1–28.
6. Ding S, Zhang D, Ge J, Yuan X, Du X. Encrypt DNS traffic: automated feature learning method for detecting DNS tunnels. In: 2021 IEEE International Conference on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom); 2021; IEEE. p. 352–9.
7. MontazeriShatoori M, Davidson L, Kaur G, Lashkari AH. Detection of doh tunnels using time-series classification of encrypted traffic. In: 2020 IEEE International Conference on Dependable, Autonomic and Secure Computing, International Conference on Pervasive Intelligence and Computing, International Conference on Cloud and Big Data Computing, International Conference on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech); 2020; IEEE. p. 63–70.
8. Singh SK, Roy PK. Detecting malicious dns over https traffic using machine learning. In: 2020 International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT); 2020; IEEE. p. 1–6.
9. Alzighaibi AR. Detection of DoH traffic tunnels using deep learning for encrypted traffic classification. Computers. 2023;12(3):47. doi:10.3390/computers12030047.
10. Sharma M, Tomar A, Hazra A. Edge computing for Industry 5.0: Fundamental, applications and research challenges. IEEE Internet Things J. 2024;11(11):19070–93
11. Nguyen DC, Ding M, Pathirana PN, Seneviratne A, Li J, Poor HV. Federated learning for internet of things: a comprehensive survey. IEEE Commun Surv Tutor. 2021;23(3):1622–58. doi:10.1109/COMST.2021.3075439.
12. Zhao X, Huang G, Jiang J, Gao L, Li M. Research on lightweight anomaly detection of multimedia traffic in edge computing. Comput Secur. 2021;111(2):102463. doi:10.1016/j.cose.2021.102463.
13. Wan S, Ding S, Chen C. Edge computing enabled video segmentation for real-time traffic monitoring in internet of vehicles. Pattern Recognit. 2022;121(11):108146. doi:10.1016/j.patcog.2021.108146.
14. Kim K, Lee JH, Lim HK, Oh SW, Han YH. Deep RNN-based network traffic classification scheme in edge computing system. Comput Sci Inf Syst. 2022;19(1):165–84. doi:10.2298/CSIS200424038K.
15. Song X, Guo Y, Li N, Zhang L. Online traffic flow prediction for edge computing-enhanced autonomous and connected vehicles. IEEE Trans Vehicular Technol. 2021;70(3):2101–11. doi:10.1109/TVT.2021.3057109.
16. Modupe OT, Otitoola AA, Oladapo OJ, Abiona OO, Oyeniran OC, Adewusi AO, et al. Reviewing the transformational impact of edge computing on real-time data processing and analytics. Comput Sci IT Res J. 2024;5(3):603–702.
17. Chen J, Yan H, Liu Z, Zhang M, Xiong H, Yu S. When federated learning meets privacy-preserving computation. ACM Comput Surv. 2024;56(12):1–36. doi:10.1145/3679013.
18. Wen J, Zhang Z, Lan Y, Cui Z, Cai J, Zhang W. A survey on federated learning: challenges and applications. Int J Mach Learn Cybern. 2023;14(2):513–35. doi:10.1007/s13042-022-01647-y.
19. Huang K. Federated learning for network traffic analysis. Italy: Politecnico di Torino; 2023.
20. Mateus J, Zodi GAL, Bagula A. Federated learning-based solution for DDoS detection in SDN. In: 2024 International Conference on Computing, Networking and Communications (ICNC); 2024; IEEE. p. 875–80.
21. Doriguzzi-Corin R, Siracusa D. FLAD: adaptive federated learning for DDoS attack detection. Comput Secur. 2024;137(4):103597. doi:10.1016/j.cose.2023.103597.

22. de Caldas Filho FL, Soares SCM, Oroski E, de Oliveira Albuquerque R, da Mata RZA, de Mendonça FLL, et al. Botnet detection and mitigation model for IoT networks using federated learning. Sensors. 2023;23(14):6305. doi:10.3390/s23146305.
23. He Z, Yin J, Wang Y, Gui G, Adebisi B, Ohtsuki T, et al. Edge device identification based on federated learning and network traffic feature engineering. IEEE Trans Cogn Commun Netw. 2021;8(4):1898–909. doi:10.1109/TCCN.2021.3101239.
24. Liu L, Zhang J, Song S, Letaief KB. Client-edge-cloud hierarchical federated learning. In: ICC 2020–2020 IEEE International Conference on Communications (ICC); 2020; IEEE. p. 1–6.
25. Jarwan A, Ibnkahla M. Edge-based federated deep reinforcement learning for IoT traffic management. IEEE Internet Things J. 2022;10(5):3799–3813. doi:10.1109/JIOT.2022.3174469.
26. Liu Y, James J, Kang J, Niyato D, Zhang S. Privacy-preserving traffic flow prediction: a federated learning approach. IEEE Internet Things J. 2020;7(8):7751–63. doi:10.1109/JIOT.2020.2991401.
27. Wang Z, Li Z, Fu M, Ye Y, Wang P. Network traffic classification based on federated semi-supervised learning. J Syst Archit. 2024;149(2):103091. doi:10.1016/j.sysarc.2024.103091.
28. Fotse YSN, Tchendji VK, Velempini M. Federated learning based DDoS attacks detection in large scale software-defined network. IEEE Trans Comput. 2024;74(1):101–15. doi:10.1109/TC.2024.3474180.
29. Lai WP, Wang JC. Editorial for the special issue on learning, security, AIoT for emerging communication/networking systems. APSIPA Trans Signal Inf Process. 2023;12(2):e13. doi:10.1561/116.00000102.