

Article

Encrypted Traffic Detection via a Federated Learning-Based Multi-Scale Feature Fusion Framework

Yichao Fei ¹, Youfeng Zhao ² , Wenrui Liu ^{1,3}, Fei Wu ², Shangdong Liu ⁴ , Xinyu Zhu ¹, Yimu Ji ⁴
and Pingsheng Jia ^{1,*}

¹ China Tower (Nanjing) Science and Technology Innovation Center, Nanjing 210018, China; feiyc3@chinatewercom.cn (Y.F.); liuwr@chinatewercom.cn (W.L.); zhuxy13@chinatewercom.cn (X.Z.)

² College of Artificial Intelligence, Nanjing University of Posts and Telecommunications, Nanjing 210023, China; 1324059218@njupt.edu.cn (Y.Z.); feiw@njupt.edu.cn (F.W.)

³ School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing 100081, China

⁴ School of Computer, Nanjing University of Posts and Telecommunications, Nanjing 210023, China; lsd@njupt.edu.cn (S.L.); jiym@njupt.edu.cn (Y.J.)

* Correspondence: jiaps@chinatewercom.cn

Abstract

With the proliferation of edge computing in IoT and smart security, there is a growing demand for large-scale encrypted traffic anomaly detection. However, the opaque nature of encrypted traffic makes it difficult for traditional detection methods to balance efficiency and accuracy. To address this challenge, this paper proposes FMTF, a Multi-Scale Feature Fusion method based on Federated Learning for encrypted traffic anomaly detection. FMTF constructs graph structures at three scales—spatial, statistical, and content—to comprehensively characterize traffic features. At the spatial scale, communication graphs are constructed based on host-to-host IP interactions, where each node represents the IP address of a host and edges capture the communication relationships between them. The statistical scale builds traffic statistic graphs based on interactions between port numbers, with nodes representing individual ports and edge weights corresponding to the lengths of transmitted packets. At the content scale, byte-level traffic graphs are generated, where nodes represent pairs of bytes extracted from the traffic data, and edges are weighted using pointwise mutual information (PMI) to reflect the statistical association between byte occurrences. To extract and fuse these multi-scale features, FMTF employs the Graph Attention Network (GAT), enhancing the model's traffic representation capability. Furthermore, to reduce raw-data exposure in distributed edge environments, FMTF integrates a federated learning framework. In this framework, edge devices train models locally based on their multi-scale traffic features and periodically share model parameters with a central server for aggregation, thereby optimizing the global model without exposing raw data. Experimental results demonstrate that FMTF maintains efficient and accurate anomaly detection performance even under limited computing resources, offering a practical and effective solution for encrypted traffic identification and network security protection in edge computing environments.

Keywords: anomaly detection; encrypted traffic; multi-scale feature fusion; federated learning; edge computing



Academic Editor: Hung-Yu Chien

Received: 14 December 2025

Revised: 20 March 2026

Accepted: 22 March 2026

Published: 9 April 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and

conditions of the [Creative Commons](https://creativecommons.org/licenses/by/4.0/)

[Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

With the rapid expansion of IoT and edge computing technologies [1,2], massive volumes of data traffic are now generated by distributed devices and must be processed and analyzed in real time [3]. Among these tasks, anomaly detection for encrypted traffic has become a core issue in network security [4]. However, due to the opaque nature of encrypted payloads and the limited computational and storage resources of edge devices, traditional schemes that depend on deep packet inspection (DPI) [5] or centralized traffic analysis are difficult to apply directly in edge computing environments. Moreover, edge scenarios inherently involve distributed data and stringent privacy requirements, which further complicate the design of effective anomaly detection mechanisms [6].

In this context, federated learning (FL) has emerged as a promising distributed machine learning paradigm [7] for edge computing. A number of studies have explored applying FL to encrypted traffic detection. For instance, Lv et al. [8] introduced a DDoS detection scheme combining convolutional neural networks (CNNs) and federated learning, where local devices use CNNs to extract features and only share model parameters via the FL framework, thereby improving detection performance while preserving data privacy. Zhang et al. [9] proposed the FLDDoS model, which also relies on FL but employs long short-term memory networks (LSTMs) to process temporal traffic sequences [10–12], enhancing the ability to model attack dynamics. Tian et al. [13] further combined a lightweight residual network (ResNet) with FL, successfully reducing computation overhead while improving classification accuracy for DDoS attacks.

Despite these advances, existing work on encrypted traffic anomaly detection still has several shortcomings. First, many approaches rely on a single type of traffic feature or a limited set of attributes, which makes it difficult to comprehensively characterize the complex and diverse behaviors present in encrypted traffic. Second, unified detection modeling across multiple attack types is often underexplored, and there is a lack of multi-scale feature representations tailored to different attack patterns. Furthermore, although some methods adopt federated learning frameworks, their privacy-preserving mechanisms and feature representation capabilities remain limited, making it challenging to achieve efficient detection of heterogeneous anomalous traffic while simultaneously safeguarding user privacy.

To address these challenges, this paper presents FMTF, a multi-scale feature fusion method based on federated learning for encrypted traffic anomaly detection. In FMTF, each edge device independently performs multi-scale feature extraction and fusion to build a rich, high-dimensional representation of encrypted traffic. Concretely, the framework operates along three complementary dimensions: at the spatial scale, it constructs interaction graphs based on IP host communications; at the statistical scale, it forms traffic statistic graphs where ports serve as nodes and edges encode packet length and direction information; and at the content scale, it generates byte-level graphs using pointwise mutual information (PMI) to capture fine-grained correlations between bytes.

For these graph-structured features, FMTF employs the graph attention network (GAT) to efficiently learn and integrate multi-scale statistical representations. Under the federated learning framework, each edge device conducts local graph construction, feature extraction, and model training, then uploads only its model parameters to a central server for aggregation. The server updates the global model and redistributes it to all devices, enabling iterative refinement of the detection model without ever exposing raw traffic data.

This decentralized learning paradigm improves anomaly detection performance while reducing raw-data exposure. By confining feature extraction and model training to edge devices and avoiding the transmission of original traffic, FMTF reduces the risk of direct data leakage. Under the FL-based aggregation protocol, only model parameters are ex-

changed during collaboration, enabling joint optimization of the global model without transmitting raw traffic data. These privacy-aware characteristics make FMTF suitable for practical deployment in distributed edge computing environments.

Meanwhile, it should be noted that federated learning alone does not fully guarantee data confidentiality in adversarial settings, as model updates may still leak information through inference attacks such as gradient inversion or membership inference. To further strengthen privacy protection, additional mechanisms commonly adopted in federated learning systems, including secure aggregation and differential privacy, can be incorporated when needed to mitigate update-level leakage risks. These privacy-aware characteristics make FMTF well suited for practical deployment in distributed edge computing environments.

In summary, the main contributions of this paper are as follows:

- This paper proposes FMTF, a Multi-Scale Feature Fusion method based on Federated Learning for encrypted traffic anomaly detection in edge computing environments. FMTF models encrypted traffic from three complementary dimensions—spatial scale, statistical scale, and content scale—by constructing interaction graphs, traffic statistic graphs, and byte-level traffic graphs. Graph Attention Networks (GATs) are employed to extract and fuse these multi-scale features, enabling rich and deep traffic representation.
- To address privacy concerns in distributed edge scenarios, FMTF integrates a federated learning framework that enables edge devices to perform local feature extraction and model training without transmitting raw traffic data. Global model updates are aggregated using FedAvg.
- Extensive experiments across diverse edge computing settings demonstrate that FMTF achieves high detection accuracy with significantly reduced computational and communication costs. Moreover, the model maintains stable performance across various types of network attacks, validating its robustness, generalization capability, and practical effectiveness in real-world edge scenarios.

2. Related Work

With the continuous evolution of cyber-attacks, encrypted traffic anomaly detection has become the focus of cyber-security research. Traditional methods such as deep packet inspection (DPI), statistical feature analysis, and machine learning detection face challenges such as high computational overhead and insufficient privacy protection in encrypted traffic environments. Federated Learning (FL), as a distributed learning framework, provides new ideas for traffic detection under privacy protection. Researchers combine FL with deep learning to improve the detection accuracy, but there are still problems such as high computational cost and underutilization of features.

In this section, we first review the methods of traffic monitoring and encrypted traffic anomaly detection, followed by analyzing the application of FL in edge computing environments, and finally summarizing the research progress of FL in encrypted traffic detection and its limitations, which provides the research background for the multi-scale feature fusion method proposed in this paper.

2.1. Traffic Monitoring and Encrypted Traffic Anomaly Detection

With the continuous evolution of network attacks, traffic monitoring techniques have become an important research direction in network security. Traditional traffic detection methods mainly include deep packet inspection (DPI)-based [4,5], traffic statistical characterization-based [14], and machine learning-based anomaly detection methods [1]. Specifically, DPI detects packets by parsing their load information. However, the popularity

of encrypted traffic makes DPI difficult to apply directly. In addition, DPI requires high computational resources and is difficult to apply to resource-constrained edge devices.

In contrast, statistical feature-based traffic detection methods perform anomalous behavior analysis by extracting features such as packet lengths, time intervals, traffic rates, etc., and representative studies include the traffic pattern-based anomaly detection method proposed by Tran et al. [2]. However, such methods are often difficult to capture the deep behavioral patterns of encrypted traffic.

In recent years, the application of machine learning and deep learning techniques has greatly promoted the progress of network traffic detection. For example, Ji et al. [15] reviewed AI-based anomaly detection over encrypted traffic, outlining key methods, challenges, and future directions. Lotfollahi et al. [16] proposed the Deep Packet method, which utilizes a deep learning model to classify encrypted traffic and achieves effective identification of VPN and non-VPN traffic. In addition, Barradas et al. [17] proposed FlowLens for efficient flow classification in ML-based network security applications, and Zhao et al. [18] proposed ERNN to improve encrypted traffic detection robustness under network-induced phenomena. Kim [19] proposed a one-class autoencoder using minimal metadata, improving anomaly detection in imbalanced encrypted IoT traffic. However, these methods mainly rely on a centralized training model, which leads to high computational cost, insufficient data privacy protection, and limited generalization capability when dealing with encrypted traffic, and is especially difficult to adapt to resource-constrained edge computing environments.

It is also worth noting that encryption does not eliminate information leakage from observable traffic patterns and side channels. For example, FOAP shows that fine-grained open-world Android app fingerprinting remains possible from encrypted traffic features [20], and Ni et al. further demonstrate that app activities can even be inferred via radio-frequency energy harvesting signals beyond the network layer [21]. While FMTF targets anomaly detection rather than side-channel defense, these studies motivate robust modeling of non-payload cues under encryption, which aligns with our multi-scale representations.

To address these challenges, researchers have proposed a variety of innovative approaches. For example, the HyperVision system developed by Chuanpu Fu et al. [22] detects encrypted malicious traffic in an unsupervised manner by constructing traffic interaction patterns through memory graphs, however, its detection performance depends on specific traffic structures and its adaptability to face complex network environments is still limited. Amardeep Singh and Julian Jang-Jaccard proposed a self-encoder unsupervised intrusion detection method [23] that combining multiscale CNN and LSTM, which performs well in dealing with spatial and temporal features of traffic, but it has a large computational overhead and is not applicable to resource-constrained edge devices.

2.2. Encrypted Traffic Anomaly Detection Based on Federated Learning

In recent years, researchers have begun to try to apply federated learning (FL) to traffic anomaly detection tasks in order to improve detection capabilities while protecting data privacy. For example, Lv et al. [8] proposed a DDoS attack detection method based on CNN and FL, which utilizes CNN to extract features at the local device and shares model parameters through the FL framework, enabling multiple devices to collaboratively train a global model, thus improving detection accuracy while protecting privacy. However, the method relies mainly on CNNs for feature extraction, which may have limitations when dealing with complex time-series traffic data, making it difficult to adequately capture the dynamic features of attack traffic.

On the other hand, several studies rely on the FedAvg algorithm, which aggregates local models through a weighted average strategy. Pourahmadi et al. [24] proposed an anomaly-exposure-based cross-domain federated learning approach for detecting DDoS attacks at the network edge. This method enhances detection performance while preserving data privacy by sharing model parameters instead of raw data. Li et al. [25] developed the FIDS system, which employs federated learning to improve the efficiency and accuracy of DDoS detection while maintaining data confidentiality. Nguyen et al. [26] introduced D²IoT, a federated self-learning anomaly detection system for IoT environments that autonomously learns normal behavior patterns and detects anomalies without centralized data collection, thereby enhancing IoT security. Zhao et al. [27] proposed a semi-supervised federated learning method for intrusion detection, which integrates a small amount of labeled data with a large volume of unlabeled data to improve detection performance while ensuring privacy protection. Zhang et al. [28] introduced GuardianAI, a privacy-preserving federated anomaly detection system that employs adaptive aggregation techniques to enhance detection performance.

Although these approaches have demonstrated effectiveness in terms of detection accuracy and privacy preservation, they still face several limitations. Specifically, they may struggle to capture complex traffic relationships in dynamic network environments. Furthermore, most of these methods rely on a single type of feature modeling, such as statistical or temporal features [29–32], and thus fail to fully exploit the multi-dimensional characteristics of network traffic.

To enhance the ability to capture attack patterns, Zhang et al. [9] proposed the FLDDoS model, which employs a long short-term memory (LSTM) network to process temporal traffic data and enables collaborative multi-device detection under the federated learning (FL) framework. The method effectively models the temporal dependencies of traffic and demonstrates improved detection performance for certain types of attacks. However, FLDDoS mainly focuses on temporal features while ignoring other important dimensions of traffic data, such as spatial and content-related characteristics, leading to insufficient feature extraction. In addition, its privacy protection strategy is relatively limited, which may reduce its effectiveness in practical federated learning scenarios involving sensitive data.

In addition, to reduce computational cost and improve detection efficiency, Tian et al. [13] proposed an efficient DDoS attack detection method by combining ResNet and FL. The method utilizes ResNet for feature extraction at the local device and shares model updates across multiple devices via the FL framework, thus reducing computational and communication overheads. However, the method still relies on deep learning model optimization in practical deployment, and still suffers from unstable training and limited model convergence speed when facing the problem of device resource heterogeneity.

2.3. Federated Learning in Edge Computing Environments

Federated Learning (FL), as an emerging distributed machine learning framework [7], has been widely applied to privacy-sensitive data analysis tasks. Under the FL system, each participating device locally trains the model and uploads the model parameters to the central server for aggregation, which effectively reduces the risk of data leakage.

In edge computing environments, studies have been conducted to explore the optimization of FL on resource-constrained devices. For example, Haripriya et al. [33] proposed TAPFed, a threshold secure aggregation method designed to enhance privacy preservation in federated learning environments. McMahan et al. [7] proposed FedProx to improve model stability and adapt to heterogeneous devices by adding constraint terms during local model training. Li et al. [34] proposed FedBN to optimize the generalization ability of the model for the variability of data distributions across devices. Additionally,

Purohit et al. [35] proposed FL-ADS, a federated learning anomaly detection system for distributed energy resource networks, which employs a hybrid approach combining vertical and horizontal federated learning to facilitate collaborative anomaly detection while preserving data privacy. However, these studies mainly focus on optimizing the FL training strategy rather than improving it for the encrypted traffic detection task.

To better highlight the research gap and position our contribution, Table 1 provides an intuitive comparison of representative encrypted-traffic detection paradigms under centralized training and federated learning. The comparison suggests that existing approaches often focus on a single perspective of traffic evidence and do not provide a unified federated design that can jointly leverage complementary information for privacy-sensitive edge deployment.

Table 1. Comparison of representative encrypted-traffic detection paradigms.

Category	Paradigm	Spatial	Statistical	Content-Related
DPI-based [4,5]	Centralized	×	×	✓
Statistical ML [2]	Centralized	×	✓	×
Deep learning-based traffic detection [16–18]	Centralized	×	✓	×
FL-based deep models [9,13]	FL	×	×	×
FL + statistical/metadata detection [24,26]	FL	×	✓	×
Graph-based interaction modeling [22]	Centralized	✓	✓	×
FMTF	FL	✓	✓	✓

“✓” denotes that the paradigm supports the corresponding feature type, whereas “×” denotes that it does not.

To address the challenges of encrypted traffic anomaly detection in edge computing environments, this paper proposes FMTF, a federated anomaly detection framework for encrypted traffic. FMTF incorporates federated learning to enable decentralized model training and protect data privacy, strengthens feature extraction by integrating multi-dimensional traffic representations, and is designed to ensure stable performance under distributed edge deployment. This approach effectively enhances the accuracy and robustness of encrypted traffic anomaly detection in real-world edge scenarios.

3. Methodology

3.1. General Framework

Figure 1 illustrates the general structure of the proposed framework, FMTF, which consists of several modules designed to achieve efficient encrypted traffic anomaly detection and collaboratively obtain globally optimized parameters under a federated learning setting.

First, the network traffic preprocessing module parses and formats raw traffic data, transforming it into a standardized representation that facilitates subsequent modeling and feature extraction. Second, the multi-scale representation module transforms traffic data into graph-structured inputs from three complementary perspectives—spatial, statistical, and content—each capturing distinct aspects of network behavior. Specifically, global interaction graphs are constructed to reflect communication relationships, statistical graphs are derived from packet-level traffic metrics to characterize statistical properties, and content-level graphs are built using byte-wise pointwise mutual information (PMI) to capture fine-grained content correlations. This multi-perspective representation enables a more comprehensive and expressive modeling of network behaviors, effectively addressing the limitations of single-feature approaches commonly adopted in previous studies. Finally, these graph representations are fed into a Graph Attention Network (GAT), which employs attention mechanisms to adaptively learn the importance of different traffic aspects and generate expressive feature embeddings. The resulting embeddings are used to identify

abnormal patterns in encrypted traffic. During model training, a federated learning framework enables collaborative learning among edge devices and the central server, where only model parameters—not raw data—are exchanged. This decentralized strategy ensures privacy preservation, avoids centralized data transmission, and mitigates the limitations of traditional centralized architectures. Additionally, the system continuously monitors the convergence of local models and optimizes global updates to maintain training stability and improve overall detection performance in distributed edge environments.

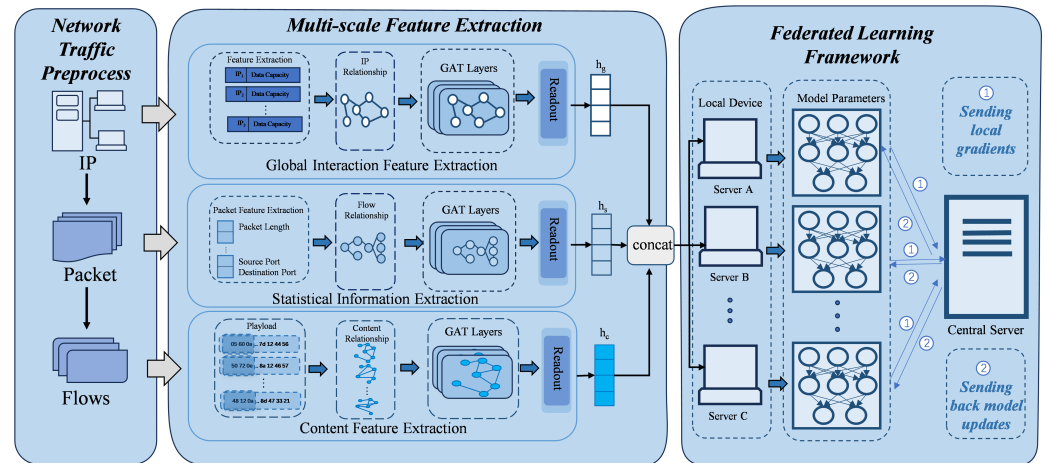


Figure 1. Overall architecture of the proposed FMTF framework.

3.2. Network Traffic Preprocessing

The primary task of this module is to process and convert raw network traffic data from edge devices into a format suitable for analysis. It operates in three main steps: traffic extraction, which obtains basic features such as IP addresses and data size from network packets; packet processing, which further extracts detailed information including packet length, timestamp, and interaction port number; and traffic formatting, which transforms the processed data into a structured traffic stream suitable for feature extraction and subsequent analysis. This module ensures that the input to the feature extraction stage is clean, consistent, and reliable.

3.3. Multi-Scale Feature Fusion

Feature extraction is a critical step in anomaly detection for encrypted traffic. However, in edge computing environments, traditional methods often struggle to capture deep-level traffic patterns, resulting in limited detection performance. To address this issue, this paper introduces a multi-scale feature extraction approach that captures behavioral characteristics from different scales. By integrating multiple feature representations, the method effectively mines pattern information in encrypted traffic and enhances the model's ability to distinguish anomalous behavior, thereby improving detection accuracy.

3.3.1. Spatial Scale Feature Extraction

For spatial-scale feature extraction, we adopt the global interaction feature extraction framework, which consists of four main steps: feature extraction, IP interaction relationship modeling, graph attention network (GAT) processing, and feature aggregation (Readout). First, the basic features of IP hosts are extracted from the raw traffic data. Key features of each IP host IP_i acts as a node in the network include traffic volume, i.e., the total amount of data sent or received per unit of time, as well as other network behavioral features, such as the number of connections and the frequency of packet exchanges, etc., which constitute an initial feature representation of IP hosts and provide the basis for subsequent

interaction modeling. Subsequently, we construct an IP interaction graph to characterize traffic interaction patterns between hosts. An edge e_{ij} is created if there is a data interaction between two hosts (IP_i, IP_j) , and the edge weight W_{ij} is determined by the communication frequency (or traffic volume) over a time period:

$$W_{ij} = \sum_{t=1}^T f_{ij}(t). \quad (1)$$

where $f_{ij}(t)$ denotes the amount of data exchanged between hosts i and j at time moment t , and T is the total duration of the period. This weighting scheme quantifies the interaction strength between hosts and thus helps capture their communication patterns.

This paper presents the details of the construction of the IP interaction graph in Algorithm 1. The algorithm builds a graph that captures the communication relationships between IP addresses by extracting the source IP address (src_ip) and destination IP address (des_ip) from each packet and assigning a unique integer ID to each IP. First, an empty list ip_edges is initialized to store the edges, and a dictionary ip_to_int is used to map IP addresses to integer IDs. Then, the algorithm iterates through each packet to extract src_ip and des_ip , assigns new IDs if the IP_s are previously unseen, and appends the corresponding edge to ip_edges . As illustrated in Figure 2, the nodes represent different IP addresses and the edges denote their communication relationships.

Algorithm 1 Construction of IP Interaction Graph

Require: A list of data packets, each with fields src_ip , des_ip

Ensure: A weighted graph representing communication between IP_s

```

1: Procedure Build IP Graph(data)
2:   Initialize  $ip\_edges$  as an empty list
3:   Initialize  $W$  as an empty dictionary  $\{W_{ij}$  in Equation (1)}
4:   for each entry in data do
5:     Extract  $src\_ip$  and  $des\_ip$  from entry
6:     if  $src\_ip$  is not in  $ip\_to\_int$  then
7:       Assign  $src\_ip$  a new ID and increment  $ip\_counter$ 
8:     end if
9:     if  $des\_ip$  is not in  $ip\_to\_int$  then
10:      Assign  $des\_ip$  a new ID and increment  $ip\_counter$ 
11:    end if
12:    Set  $i = ip\_to\_int[src\_ip]$  and  $j = ip\_to\_int[des\_ip]$ 
13:    if  $(i, j)$  not in  $W$  then
14:      Set  $W[(i, j)] = 0$ 
15:    end if
16:    Set  $W[(i, j)] = W[(i, j)] + 1$ 
17:  end for
18:  for each key  $(i, j)$  in  $W$  do
19:    Append the tuple  $(i, j, W[(i, j)])$  to  $ip\_edges$ 
20:  end for
21:  Create NetworkX graph  $G\_ip$ 
22:  Add weighted edges from  $ip\_edges$  to  $G\_ip$ 
23:  return  $G\_ip$ 

```

3.3.2. Statistical Scale Feature Extraction

In statistical scale feature extraction, we first extract packet-level attributes, including the source/destination ports and the packet length. We then construct a traffic relationship graph based on these attributes, where each port p is represented as a node V_p , $p \in (1, 2, \dots, M)$, and M is the total number of involved ports. For each packet transmitted from a source port to a destination port, an edge is created between the corresponding port

nodes. To incorporate packet-size information in a reproducible manner, we associate each port-pair edge with a weight that aggregates packet lengths over the observation period. Specifically, for a port pair (p, q) , the edge weight W_{pq} is computed as the sum of packet lengths of all packets observed on (p, q) during the period.

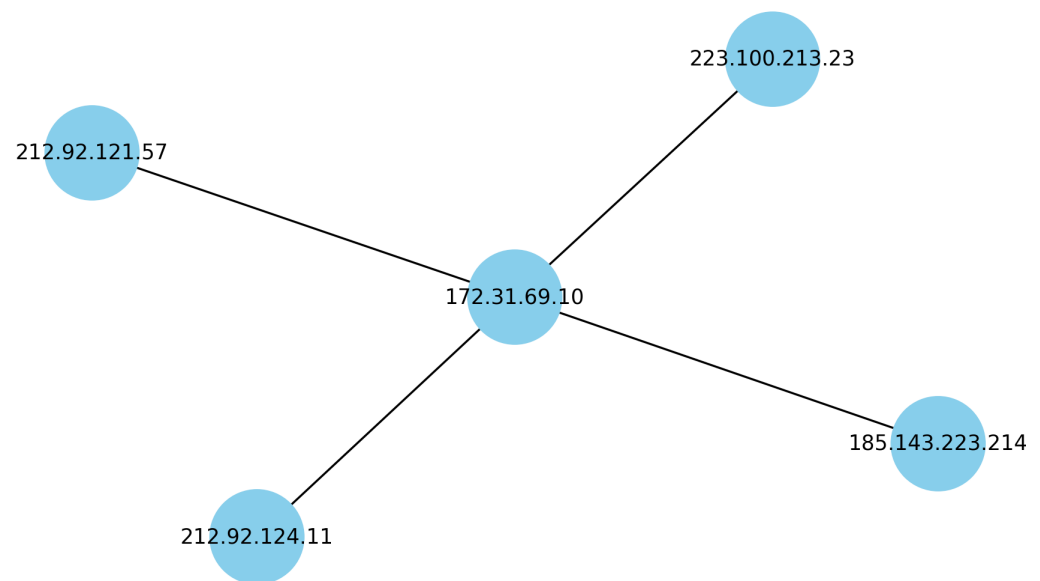


Figure 2. IP host interaction diagram. This figure illustrates the structure of the IP Interaction Graph, where the blue nodes represent IP addresses, and the edges indicate communication links between them.

Algorithm 2 presents the detailed process for constructing the statistical graph (G_{stats}). The graph models port-to-port communication by extracting the source port (src_port) and destination port (des_port) from each packet. To encode packet-size information, the algorithm accumulates packet lengths for each port pair and assigns the aggregated value as the edge weight. Specifically, for each (src_port, des_port) , the corresponding weight is updated by adding pkt_len . After processing all packets, the algorithm creates a weighted edge list and constructs the *NetworkX* graph G_{stats} using these weighted edges. As illustrated in Figure 3, nodes represent distinct ports and edges denote their communication interactions with packet-length-based weights.

Algorithm 2 Construction of Statistics Graph

Require: A list of data packets, each with fields src_port , des_port , pkt_len

Ensure: A weighted graph representing communication between ports

- 1: Procedure Build Stats Graph(data)
 - 2: Initialize $stats_edges$ as an empty list
 - 3: Initialize W as an empty dictionary
 - 4: **for** each entry in data **do**
 - 5: Extract src_port , des_port and pkt_len from entry
 - 6: **if** (src_port, des_port) is not in W **then**
 - 7: Set $W[(src_port, des_port)] = 0$
 - 8: **end if**
 - 9: Set $W[(src_port, des_port)] = W[(src_port, des_port)] + pkt_len$
 - 10: **end for**
 - 11: **for** each key (p, q) in W **do**
 - 12: Append the tuple $(p, q, W[(p, q)])$ to $stats_edges$
 - 13: **end for**
 - 14: Create *NetworkX* graph G_{stats}
 - 15: Add weighted edges from $stats_edges$ to G_{stats}
 - 16: **return** G_{stats}
-

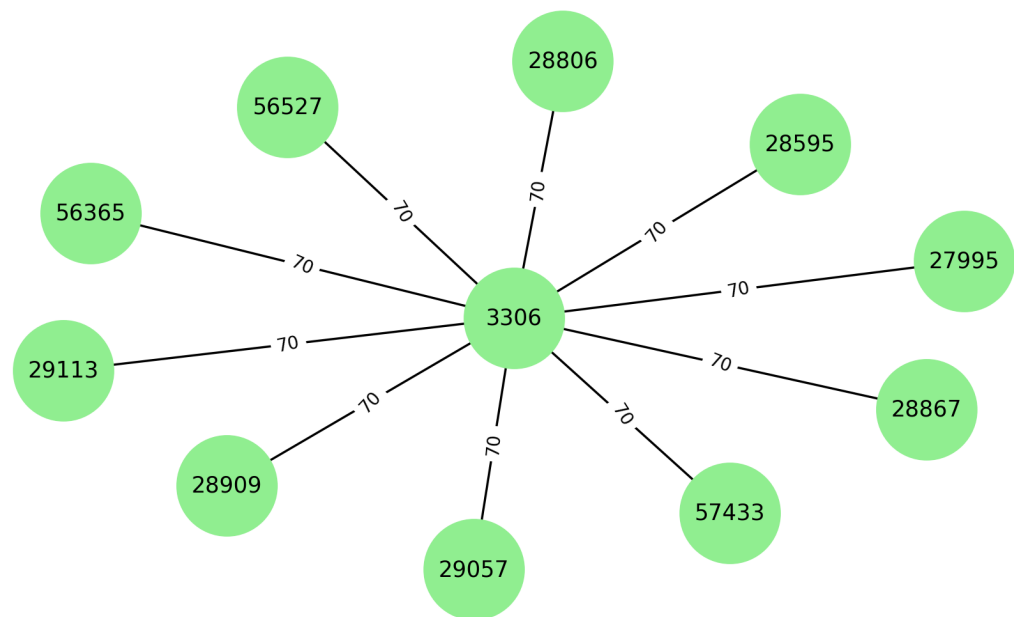


Figure 3. Traffic statistics diagram. This figure presents a demo of a Statistics Graph, where the green nodes represent communicating ports, the numbers on the nodes indicate specific port numbers, and the edge labels denote packet lengths.

3.3.3. Content Scale Feature Extraction

In content-scale feature extraction, we portray the correlation between byte pairs by computing the pointwise mutual information (PMI). PMI is used to measure the dependency between two bytes in a given context. Consider two bytes b_1 and b_2 , their joint probability $P(b_1, b_2)$ and marginal probability, $P(b_1)$ and $P(b_2)$ can be used to compute the PMI value:

$$PMI(b_1, b_2) = \log \left(\frac{P(b_1, b_2) + \epsilon}{(P(b_1) + \epsilon)(P(b_2) + \epsilon)} \right) \quad (2)$$

where a small smoothing parameter $\epsilon = 10^{-8}$ is introduced to avoid numerical singularities (logarithm of zero or division by zero) when marginal or joint probabilities are zero. its negligible magnitude preserves the original PMI value characteristics. In this context, $P(b_1, b_2)$ represents the joint probability of bytes b_1 and b_2 occurring together, while $P(b_1)$ and $P(b_2)$ denote the marginal probabilities of byte b_1 and byte b_2 , respectively. Based on the computed correlation between byte pairs, a byte-level traffic graph is constructed, where each byte b_i is represented as a node V_i , and the edge weight w_{ij} is determined by the PMI value between byte b_i and byte b_j , defined as:

$$w_{ij} = PMI(b_i, b_j) \quad (3)$$

After computing PMI scores for adjacent byte pairs, we filter weak co-occurrences using a PMI threshold to prevent the payload graph from becoming overly dense and noisy. Since $PMI = 0$ indicates statistical independence under $\log_2(\cdot)$, we set the cutoff to $\tau_{PMI} = 2$ and construct $G_{payload}$ by retaining only byte-pair edges with $PMI > 2$, so that only sufficiently strong byte associations are preserved while weak correlations are removed.

In this way, the byte-level traffic graph effectively captures the dependencies between bytes, providing essential content-level features for anomaly detection.

The details of constructing a byte-level traffic graph ($G_{payload}$) are shown in Algorithm 3. The algorithm constructs a graph representing the relationships between

byte pairs by extracting the byte sequences in a packet and computing the PMI between byte pairs. First, an empty *byte_data* list is initialized for storing byte data, and a *pmi_edges* list is initialized for storing byte pair edges that satisfy the PMI threshold. Subsequently, the packet is traversed to extract byte-level information, and the PMI is computed for each byte pair. If the value of the PMI exceeds the set threshold, the corresponding byte pair is added as an edge to *pmi_edges*; otherwise, it is discarded and not considered in the graph construction. Finally, a *NetworkX* graph *G_payload* is created using these edges. As shown in Figure 4, the nodes in the graph represent different bytes, the edges represent the relationships between byte pairs, and the green numbers represent specific PMI values.

Algorithm 3 Construction of Payload Graph

Require: A list of data packets, each with field *all_info* (byte-level data)

Ensure: A weighted graph representing the byte-level traffic graph based on PMI

```

1: Procedure Build Payload Graph(data)
2:   Initialize byte_data as an empty list
3:   Initialize pmi_edges as an empty list
4:   Initialize byte_pair_count and byte_count as Counter objects
5:   Set pmi_threshold = 2
6:   for each entry in data do
7:     Extract all_info from entry
8:     Append all_info to byte_data
9:   end for
10:  for each packet in byte_data do
11:    for each consecutive byte pair (packet[i], packet[i+1]) do
12:      Update byte_pair_count for (packet[i], packet[i+1])
13:      Update byte_count for packet[i] and packet[i+1]
14:    end for
15:  end for
16:  for each byte pair (b1, b2) in byte_pair_count do
17:    Calculate PMI using the formula defined in Equation (2)
18:    Set  $w_{ij} = PMI(b_1, b_2)$ 
19:    if  $w_{ij} > pmi\_threshold$  then
20:      Append the weighted edge (b1, b2, wij) to pmi_edges
21:    end if
22:  end for
23:  Create NetworkX graph G_payload
24:  Add weighted edges from pmi_edges to G_payload
25:  return G_payload

```

3.3.4. Graph Attention Network Model

For the constructed multi-view traffic graph, this paper adopts Graph Attention Network (GAT) for feature learning of encrypted traffic. The model consists of a two-layer GAT structure, which can effectively capture the node association information in graph-structured data to enhance the feature representation capability.

GAT adopts a multi-head attention (multi-head attention) mechanism to dynamically allocate attention weights among neighboring nodes to make full use of the local and global information of the input features. GAT can adaptively allocate the attention weights at the node level to capture the key information in the graph-structured data. The model adopts a two-layer GAT structure and aggregates the neighborhood information through the multi-head attention mechanism to enhance the feature representation capability.

In the first layer of GAT computation, the model performs a weighted aggregation of input node features using the multi-head attention mechanism to capture the importance and interaction patterns of neighboring nodes. Specifically, let the input graph structure be $G = (V, E)$, where V is the set of nodes, E is the set of edges, and the node feature matrix

In order to enhance the feature extraction capability and improve the stability of the model, the first layer of GAT adopts the multi-head attention mechanism (multi-head attention), it uses K independent attention heads to perform the computation and splice or average the results:

$$\vec{h}'_i = \begin{bmatrix} \sigma \left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^{(1)} W^{(1)} \vec{h}_j \right) \\ \sigma \left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^{(2)} W^{(2)} \vec{h}_j \right) \\ \vdots \\ \sigma \left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^{(K)} W^{(K)} \vec{h}_j \right) \end{bmatrix} \quad (7)$$

Among them, K denotes the number of attention heads, α_{ij}^k and W^k represent the attention coefficient and the learnable weight matrix associated with the k -th attention head, respectively. The output of each attention head is processed through a non-linear activation function and then vertically stacked in order to form a unified feature representation. As a result, the output dimension of the first-layer GAT becomes $d' \times K$, i.e., $\vec{h}'_i \in \mathbb{R}^{N \times (d'K)}$, providing higher-order node representations for subsequent layers.

Based on the hidden features extracted from the first layer GAT, the second layer GAT further transforms the features to generate the final node representation. Since the output dimension of the first layer GAT is $d' \times K$, it needs to be flattened to match the input format of single-head attention. Subsequently, the single-head attention mechanism is utilized to compute the final node representation:

$$h''_i = \sigma \left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij} W h_j \right) \quad (8)$$

Here, $h_j \in \mathbb{R}^{d'K}$ denotes the hidden feature vector of node j obtained from the first-layer GAT, which serves as the input to the second-layer GAT. W is a learnable transformation matrix, and α_{ij} represents the attention weight between node i and its neighbor j . This operation aggregates the information from neighboring nodes and generates the final node representation h''_i with a dimension of d' , which serves as the input for subsequent anomaly detection tasks.

This GAT model is able to effectively learn the multi-scale features of encrypted traffic interaction graphs, thus improving the performance of anomalous traffic detection.

After obtaining three graph views at different scales, namely the spatial interaction graph (G_{ip}), the statistical graph (G_{stats}), and the content graph ($G_{payload}$), we fuse their representations using an adaptive view-level attention mechanism.

The set of graph views is denoted as $\mathcal{V} = \{\text{ip}, \text{stats}, \text{payload}\}$. For each view $v \in \mathcal{V}$, a GAT encoder produces node embeddings $H^{(v)} = \text{GAT}(G_v, X^{(v)})$, and a readout operator summarizes them into a graph-level embedding:

$$h^{(v)} = \text{Readout}(H^{(v)}). \quad (9)$$

A lightweight scoring function $g(\cdot)$ assigns an importance score to each view, and the fusion weights are obtained by softmax normalization:

$$\beta^{(v)} = \frac{\exp(g(h^{(v)}))}{\sum_{u \in \mathcal{V}} \exp(g(h^{(u)}))}, \quad h = \sum_{v \in \mathcal{V}} \beta^{(v)} h^{(v)}. \quad (10)$$

The parameters of $g(\cdot)$ are randomly initialized and optimized jointly with the GAT encoders via backpropagation under the federated learning protocol, enabling the model to adaptively balance the contributions of spatial, statistical, and content cues.

3.4. Federated Learning Framework

To cope with the limited computational resources of devices and the need for data privacy protection in edge computing environments, this paper adopts a federated learning framework to realize collaborative training among multiple devices. Under this framework, edge devices do not need to upload raw data, but only need to upload locally trained model parameters, thus realizing effective data privacy protection. The following sections elaborate on the framework design and implementation will be elaborated in detail from three aspects: local training, model aggregation and model update.

3.4.1. Parameter Aggregation and Global Model Generation

After completing the local training, the edge device uploads the model parameters obtained from its local training to the central server, without transmitting any raw data. In the federated learning framework, this process effectively protects the privacy of the device by ensuring that sensitive user data remains stored locally.

After receiving the model parameters uploaded by each device, the central server aggregates these parameters using appropriate aggregation algorithms. Commonly used aggregation methods include Weighted Average and FedAvg, which adjust the aggregation results based on the device weights, the amount of local data, and the quality of training. In this work, the FedAvg algorithm is also adopted to aggregate the local models and generate a global model that integrates knowledge from different devices, thereby improving the overall detection performance.

3.4.2. Model Update and Optimization Iterations

After generating the global model, the central server optimizes the model to improve its performance on each device. This process can include parameter tuning, application of regularization techniques, etc. The optimized model is delivered to each edge device for the next round of local training.

After each device receives the optimized global model, it interacts with the local data to continue local training. In this way, the devices are able to use the local data for further optimization and tuning of the model. After each round of local training, the device will upload the updated model parameters to the central server again to start a new round of aggregation process.

The local training and model aggregation process will be carried out in multiple rounds of iterations. In each round of iteration, the local model of the edge device will be continuously improved, which ultimately makes the global model continuously optimized and the detection capability gradually improved. Multiple rounds of iteration not only improve the accuracy of the model, but also further strengthen the generalization ability of the model, making its performance more stable under different devices and different scenarios.

4. Experiments

4.1. Dataset

4.1.1. CICIDS2018 Dataset

The CICIDS2018 dataset, created by the Canadian Institute for Cybersecurity (CIC) [36], simulates network traffic in an enterprise environment and contains multiple types of attacks such as Bot networks, DDoS attacks, Zeus attacks, and port scans. The dataset

contains traffic data generated by approximately 500 machines and records more than 6,000,000 network traffic records, each containing up to 80 features, including protocol type, IP address, port, packet size, etc., which are able to provide detailed labeling of attack and normal traffic. These features make the dataset an ideal benchmark for the research and development of Intrusion Detection Systems (IDSs) and traffic classification models, which are widely used in cybersecurity.

4.1.2. CIC-DDoS2019 Dataset

The CIC-DDoS2019 dataset was released by the Canadian Cybersecurity Research Centre (CIC) and is dedicated to the study and detection of Distributed Denial of Service (DDoS) attacks [37]. The dataset contains both normal traffic and multiple types of DDoS attacks (e.g., DNS Flood, SYN Flood, HTTP Flood, etc.) and can help researchers develop and validate DDoS attack detection models. The dataset is collected in a real network environment, covering traffic in different time periods, with high-quality labels and rich features, including source IP, destination IP, port, protocol type, packet length, etc. The CIC-DDoS2019 dataset is divided into a training set and a test set. It is suitable for research on DDoS attack detection, traffic classification, and other studies, and is designed to help improve the accuracy of the existing protection mechanisms.

4.1.3. Experimental Data Partitioning and Processing

In order to accurately realize multi-scale feature fusion, we need specific features to be used to construct spatial scale features, statistical scale features, and content scale features. Based on the dataset characteristics, including both the CICIDS2018 and CIC-DDoS2019 datasets, several key features are selected to support feature extraction for multi-scale graph construction. The feature selection and division strategy for different scale feature construction is summarized in Table 2. For example, connection features include Flow ID, Source IP, Destination IP, Source Port, and Destination Port, which help to construct spatial scale graphs to characterize the communication relationship of flows. Flow statistics features such as Flow Duration, Total Fwd Packets, Total Bwd Packets, and Flow Bytes are mainly used to construct statistical scale graphs to reflect the overall behavior of flows. In contrast, content scale graphs utilize the full flow specifics to capture the changing relationships of byte-level flows. These selected features provide essential information for our graph structure construction and facilitate the effective extraction of multi-scale features.

Table 2. Categorization of network traffic features.

Traffic Statistics	Packet-Level Features	Connection Features
Flow Duration	Packet Length Mean	Source IP
Total Fwd Packets	Packet Length Std	Destination IP
Total Bwd Packets	Max Packet Length	Source Port
Total Length of Fwd Packets	Min Packet Length	Destination Port
Total Length of Bwd Packets	Average Packet Size	Flow ID
Flow Bytes/s	Avg Fwd Segment Size	—
Flow Packets/s	Avg Bwd Segment Size	—
Bwd Packet Length Mean	—	—

“—” indicates that no corresponding feature exists in this category.

The CICIDS2018 dataset has been categorized into seven subsets according to specific attack types. In the conducted experiments, the data division for each subset approximately adheres to a ratio of 60–70% for the training set, 10–20% for the validation set, and 20–30% for the testing set, as detailed in Table 3. This dataset comprises a broad spectrum of attack types, including Distributed Denial of Service (DDoS), Denial of Service (DoS), SQL injection attacks (MySQL) [38], web-based attacks [39], malware (e.g., Zeus), botnet activities (Bot), brute-force attacks, and infiltration attempts, thereby ensuring a comprehensive and diverse representation of malicious traffic scenarios. Such a division enables the model to be adequately trained on different types of attacks and its ability to detect various attacks is verified in the testing phase, thus proving that the model is able to effectively identify different classes of anomalous traffic.

Table 3. CICIDS 2018 dataset splits.

Attack Type	Total Set	Training Set	Validation Set	Test Set
DDoS	6065	4030	335	1700
MySQL	3045	2040	503	502
Web	5508	3508	500	1500
Zeus	7700	4500	850	2350
Bot	600	450	50	100
BruteForce	2300	1030	70	1200
Infiltration	6380	4325	425	1630

The CIC-DDoS2019 dataset contains only DDoS attack traffic, so there is no need to perform attack type partitioning. Instead, we organize the dataset into two different types for experimental purposes. One form is the balanced dataset, where the original data is divided into many smaller subsets. Each subset contains only one specific attack type along with its corresponding benign traffic samples. To introduce data imbalance, we gradually increase the number of attack samples in each subset, starting from the smallest subset with approximately 200 attack samples, and doubling the sample size progressively until the largest subset, where the number of attack samples is limited to around 50,000. In addition, we carefully adjusted the data distribution during the partitioning process to ensure that the number of benign traffic samples and DDoS attack samples in each subset is approximately balanced.

The other form is the unpartitioned dataset, where the original CIC-DDoS2019 dataset is used directly without any subset partitioning or sample adjustment, thereby preserving the original data distribution characteristics. The final dataset splits for both types are summarized in Table 4, where approximately 80% is used for training, 15% for testing, and 5% from the training set as the validation set.

Table 4. CIC-DDoS2019 dataset splits.

Dataset Type	Training Set	Validation Set	Test Set
Balanced	38,000	2000	10,000
Imbalanced	40,000	2500	7500

4.2. Comparative Methods

We compare and analyze the proposed method with several existing federated learning approaches, including FedAvg [7], FLAD [40], and FLDDoS [9]. FedAvg is a classical federated learning algorithm originally designed for distributed optimization and model aggregation tasks, rather than for network attack detection. FLAD is a federated learning-based anomaly detection method that utilizes a DDoS traffic dataset for evaluation.

FLDDoS is a recent federated learning framework specifically designed for DDoS attack detection scenarios.

We include two non-federated encrypted-traffic detection baselines for reference, namely FlowLens [17] and ERNN [18]. FlowLens calculates statistical histograms of packet-size distribution and adopts machine learning models to perform classification. ERNN integrates a finite-state automaton into an RNN for encrypted-traffic detection.

To ensure fair and reproducible comparisons, all methods are evaluated using the same train/validation/test splits and the same metrics. The federated baselines (FedAvg, FLAD, and FLDDoS) are executed under the same federated setting and training schedule as FMTF. FlowLens and ERNN are non-federated methods and are therefore evaluated in a centralized setting following their original feature pipelines on the same dataset splits. We report them as reference baselines rather than protocol-matched federated comparisons.

4.3. Experimental Details

We used an NVIDIA Tesla P40 GPU and configured TensorFlow with multithreading to improve computational efficiency. In the federated learning configuration, we deployed a distributed framework consisting of 4 edge clients and 1 central aggregation server for all experiments. Unless otherwise stated, all reported results use a fixed training protocol: 15 global communication rounds, 1 local epoch per round, batch size 64, and the Adam optimizer with an initial learning rate of 0.001, with the learning rate decayed by 10% every 5 rounds. Before each training round, all clients cloned and synchronized the global model weights.

For the Graph Attention Network (GAT) model adopted in FMTF, we used a 2-layer architecture: the first layer included 4 attention heads with 32 hidden units per head, and the second layer used a single head with 64 hidden units; both layers employed LeakyReLU activation with a negative slope of 0.2. We used a Multi-Layer Perceptron (MLP) classifier consisting of two fully connected layers, each with 32 neurons and ReLU activation.

During preliminary tuning, we explored a small range of local training settings to balance efficiency and accuracy; however, the results reported in this paper are obtained using the fixed protocol described above for full reproducibility.

For performance evaluation, we report Precision, Recall, and F1-score. F1-score is particularly suitable for imbalanced datasets because it jointly reflects precision and recall. Additionally, we report training time as an efficiency metric to assess the suitability of the proposed framework for deployment in resource-constrained edge environments.

4.4. Experimental Results and Discussion

We first discuss the detection performance of our model and existing methods on both imbalanced and balanced datasets using the CIC-DDoS2019 dataset. The experimental results (Table 5) show that FMTF demonstrates strong detection capability and stable performance in the federated learning environment.

Table 5. FL configuration metrics for detection on the CIC-DDoS2019 dataset.

Metric	FLDDoS	FLAD	FedAvg	FlowLens *	ERNN *	FMTF
F1 (Imb.)	0.9091	0.9667	0.9157	0.9109	0.9719	0.9910
F1 (Bal.)	0.7376	0.9699	0.8148	0.9215	0.9811	0.9915
F1 Std	0.1605	0.0369	0.1597	0.1552	0.1508	0.0098

"Imb." and "Bal." denote imbalanced and balanced datasets. "Std" denotes the standard deviation of F1 scores across different training rounds. * denotes non-federated encrypted-traffic detection methods.

Specifically, in terms of overall detection performance on the imbalanced dataset (original CIC-DDoS2019 dataset without any preprocessing), FMTF achieves an F1 score of 0.9910, which is higher than that of FLDDos, FLAD, and FedAvg, and also higher than the non-federated encrypted-traffic detection baselines FlowLens and ERNN. Moreover, in the DDoS attack detection task based on the balanced dataset, FMTF achieves an F1-score of 0.9915, which significantly outperforms FLDDos (0.7376) and FedAvg (0.8148), and also remains higher than FlowLens (0.9215) and ERNN (0.9811), indicating better robustness in malicious traffic identification.

Meanwhile, the standard deviation of FMTF's F1 score across different training rounds is only 0.0098, which is much lower than that of FLDDos (0.1605) and FedAvg (0.1597), and better than that of FLAD (0.0369), demonstrating that FMTF has more stable performance and is less susceptible to data heterogeneity and training fluctuations.

The CICIDS2018 dataset covers a wide range of cyber attack types with complex traffic characteristics, which puts higher requirements on the model's attack recognition capability. The experimental results on this dataset (Table 6) show that FMTF exhibits excellent detection performance in both federated learning and local training modes, which is significantly better than that of FLDDos. To clarify the experimental conditions of the Local mode and improve result interpretability, we set the number of clients in Local mode to be consistent with the Fed mode. Specifically, Local mode also involves four edge clients and no central aggregation server. Each client uses only its own local traffic dataset, and the data partition is identical to that in the Fed mode to ensure a fair comparison. Each client performs training and testing independently on its local data. The metrics reported in Table 6 are averaged over the four clients. In contrast, the Fed mode evaluates the globally aggregated model on the combined test sets of all clients.

Table 6. Comparison of performance between federated and local models on the CICIDS2018 dataset.

Metric	Mode	Precision	Recall	F1 Score
FMTF	Fed	0.9965	0.9877	0.9920
FLDDos	Fed	0.9326	0.9279	0.9262
FMTF	Local	0.9945	0.9875	0.9910
FLDDos	Local	0.9837	0.9839	0.9833

FMTF improves the detection capability by taking full account of different levels of traffic information through the multi-scale feature fusion method, which enables the model to portray the attack behavioral patterns in a more fine-grained way. In the federated learning mode, FMTF's F1 Score reaches 0.9920 and Precision is 0.9965, which far exceeds that of FLDDos (F1 Score 0.9262 and Precision 0.9326), indicating that it is able to extract the complex traffic features more efficiently and improve the accuracy of attack detection. Meanwhile, in the local training mode, FMTF's detection performance remains stable with an F1 Score of 0.9910, while the performance of FLDDos decreases significantly in the federated learning environment, indicating the limitations of its feature expression capability. The overall results validate that FMTF is able to identify different types of attack traffic more comprehensively by integrating multi-scale information, improving detection capability and robustness, and is suitable for the anomaly detection task of complex encrypted traffic.

4.5. Performance Evaluation Against Various Attack Types

Since the CICIDS2018 dataset covers a variety of attack types (e.g., DoS, DDoS, Brute Force, Botnet, etc.), and the attack traffic is interwoven with normal traffic while containing encrypted communication, time-dependent, and multi-dimensional features, it poses

greater challenges for anomaly detection. To this end, this study evaluates the detection performance of the proposed model on different attack types to verify its effectiveness in complex traffic environments.

4.5.1. Convergence Analysis of Training F1 Scores

For multiple attack types, the convergence of the global F1 score and the F1 scores on each server is shown in Figure 5. As illustrated, different attack types exhibit varying convergence trends during federated training. However, overall, both the global F1 score and the local F1 scores on each server increase rapidly within the first 10 rounds and tend to stabilize around the 15th round.

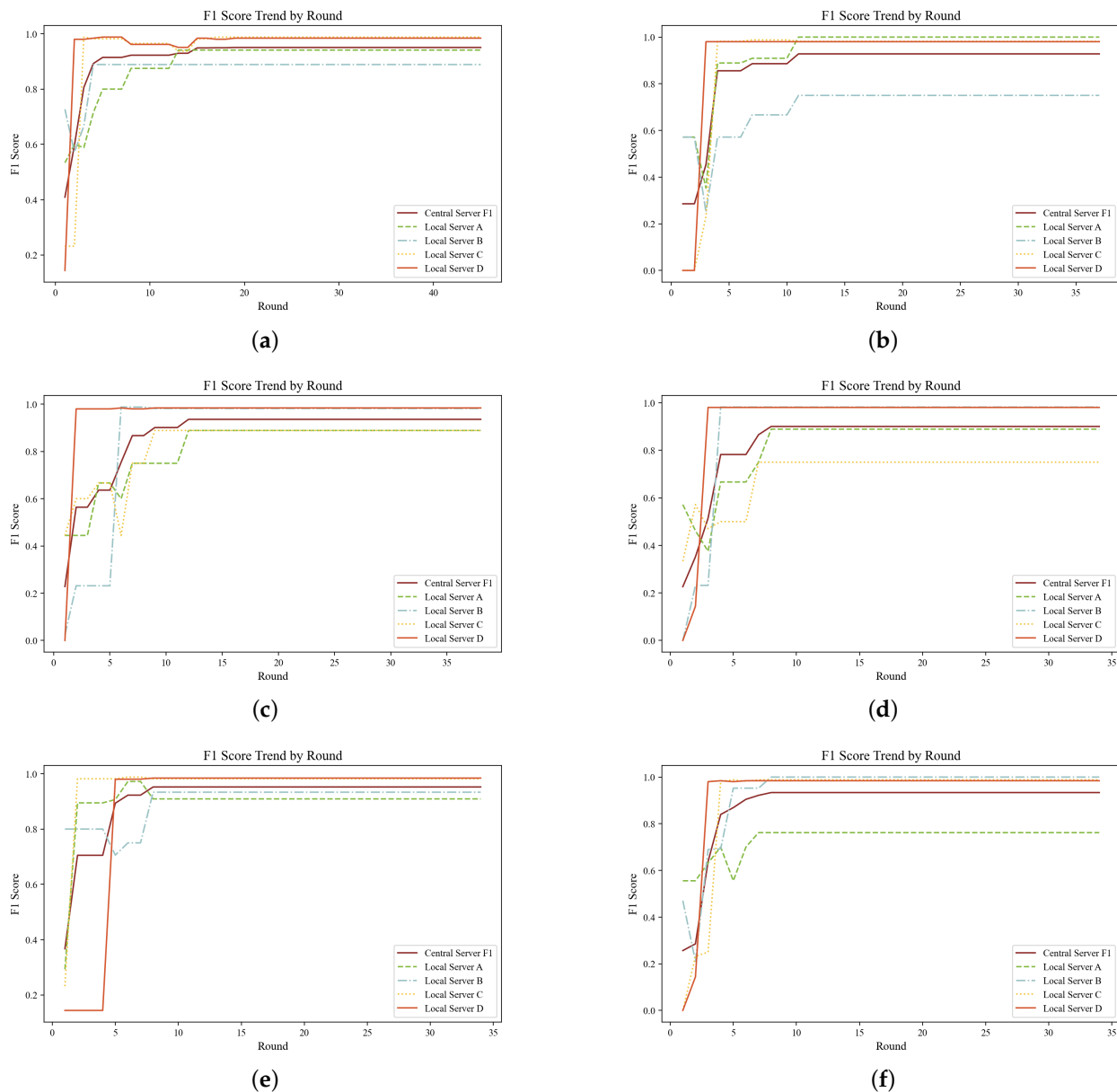


Figure 5. F1 score trends for different attack types. (a) F1 score trend for Bot attack; (b) F1 score trend for BruteForce attack; (c) F1 score trend for Infiltration attack; (d) F1 score trend for MySQL attack; (e) F1 score trend for Web attack; (f) F1 score trend for Zeus attack.

From the convergence trend of the F1 score, the model demonstrates stable learning capabilities when handling different attack types. In the early stages of training, the F1 scores for all attack types show significant improvement, particularly within the first five rounds, where the detection performance improves rapidly to a high level. As training progresses, the global F1 score gradually stabilizes and maintains excellent performance, indicating that the model is capable of effectively learning and extracting key features for various attack types. Although there are some fluctuations in the early stages, especially in terms of convergence speed, the F1 scores for all attack types stabilize after approximately 15 rounds. This suggests that the model not only effectively captures the characteristics of diverse attack types but also maintains consistent and robust detection performance across all clients. These results further validate the stability and effectiveness of the multi-scale feature fusion approach in encrypted traffic anomaly detection tasks, demonstrating its potential for practical applications.

4.5.2. Trend Analysis of Training Losses

We further analyzed the training loss trends for different attack types across training rounds, as illustrated in Figure 6. The loss values on each server exhibit a decreasing trend and eventually stabilize, indicating that the model can effectively learn the features of various attack types and achieve convergence.

Overall, the training loss for all attack types shows significant fluctuations in the early stages. Still, as training progresses, the loss gradually decreases and stabilizes, indicating that the model is able to progressively extract key features across different attack types, ultimately achieving consistent learning results. Although the convergence speed and fluctuations vary across attack types, the overall trend reflects the model's robustness when handling different attack scenarios.

Specifically, the training loss for the Web attack shows an initial rapid decline during the early training rounds. However, unlike other attack types, the loss exhibits noticeable fluctuations until around the 20th round. This suggests that the model needs longer convergence time when learning the characteristics of Web attacks, possibly due to the variability or complexity of their traffic patterns.

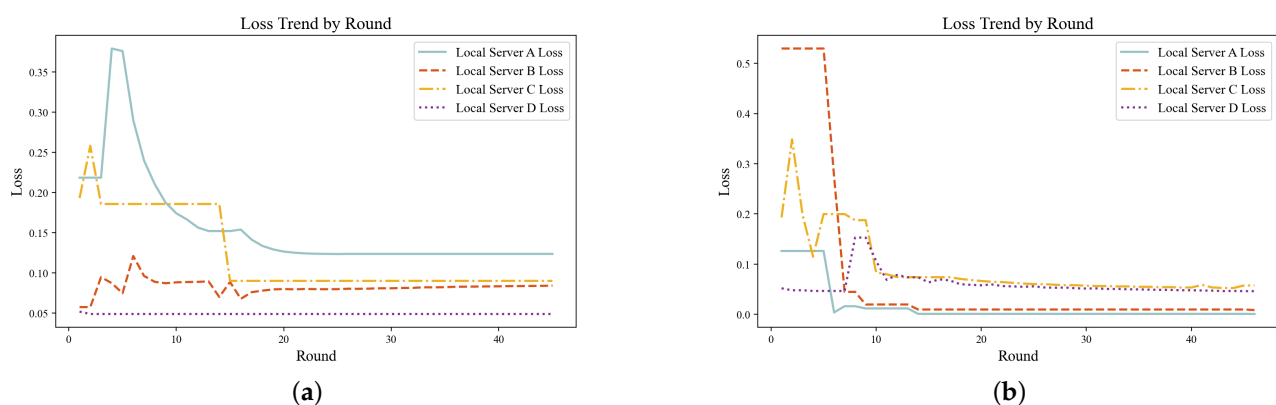


Figure 6. Cont.

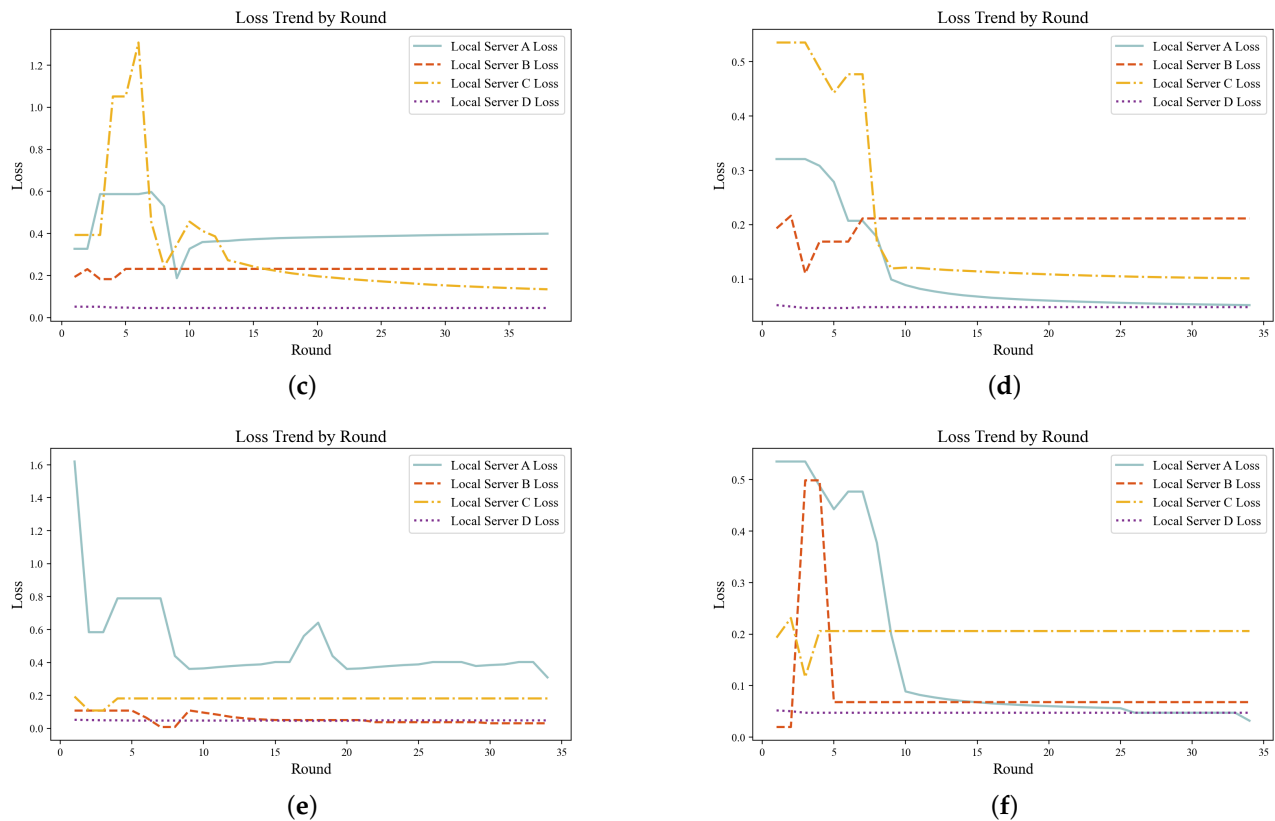


Figure 6. Training loss trends for different attack types. (a) Training loss trend for Bot attack; (b) Training loss trend for BruteForce attack; (c) Training loss trend for Infiltration attack; (d) Training loss trend for MySQL attack; (e) Training loss trend for Web attack; (f) Training loss trend for Zeus attack.

5. Conclusions

In this paper, we propose FMTF, an encrypted traffic anomaly detection method based on federated learning and multi-scale feature fusion. To address the challenges of distributed data, limited computational resources, and privacy requirements in edge computing environments, FMTF establishes an efficient detection framework. It models traffic features from three perspectives—spatial, statistical, and content—and leverages Graph Attention Networks (GATs) to extract multi-scale features. Federated learning is then employed to enable decentralized training and parameter aggregation. Experimental results show that FMTF outperforms FLAD, FedAvg, and FLDDoS in terms of detection performance on the CICIDS2018 and CIC-DDoS2019 datasets. In summary, FMTF enhances the accuracy and efficiency of encrypted traffic anomaly detection while ensuring data privacy, making it well-suited for resource-constrained edge computing environments and scalable to more complex network security detection tasks.

Author Contributions: Conceptualization, Y.F. and P.J.; methodology, Y.F., Y.Z. and P.J.; software, Y.Z.; formal analysis, Y.Z.; investigation, Y.Z., W.L., F.W., S.L., X.Z., Y.J. and P.J.; resources, Y.F., W.L., F.W., S.L., X.Z., Y.J. and P.J.; data curation, Y.F., W.L., F.W., S.L., X.Z. and P.J.; writing—original draft preparation, Y.F.; writing—review and editing, Y.F., Y.Z. and P.J.; visualization, Y.F., W.L. and F.W.; supervision, S.L. and Y.J.; project administration, F.W., S.L., Y.J. and P.J.; funding acquisition, Y.F., W.L., X.Z. and P.J. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the 2024 Sci-Tech Innovation Project of China Tower (Nanjing) Science and Technology Innovation Center (CX-JS-20241114-0-16704), and the National Key R&D Program of China (No. 2023YFB2904000).

Data Availability Statement: The datasets used in this study are publicly available. The CIC-DDoS2019 dataset is available from the Canadian Institute for Cybersecurity: <https://www.unb.ca/cic/datasets/ddos-2019.html>, accessed on 21 March 2026. The CICIDS2018 dataset is available from the Canadian Institute for Cybersecurity: <https://www.unb.ca/cic/datasets/ids-2018.html>, accessed on 21 March 2026.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. M’Kaila, J.C.; Rajabion, L. A strategic approach to IoT security by working towards a secure IoT future. *IEEE Internet Things J.* **2023**, *7*, 1–18.
2. Alotaibi, A.; Barnawi, A. Securing massive IoT in 6G: Recent solutions, architectures, and future directions. *IEEE Internet Things J.* **2023**, *22*, 100715. [CrossRef]
3. Moustafa, N. A new distributed architecture for evaluating AI-based security systems at the edge network TON_IoT datasets. *Society* **2021**, *72*, 102994. [CrossRef]
4. Kumar, S.; Dharmapurikar, S.; Yu, F.; Crowley, P.; Turner, J. Algorithms to accelerate multiple regular expressions matching for deep packet inspection. *ACM SIGCOMM Comput. Commun. Rev.* **2006**, *36*, 339–350. [CrossRef]
5. Nan, H.; Song, H.Y.; Lakshman, T.V. Variable stride multi-pattern matching for scalable deep packet inspection. In *IEEE International Conference on Computer Communications*; IEEE: Piscataway, NJ, USA, 2009; pp. 415–423.
6. Yaacoub, J.P.; Noura, H.; Salman, O.; Chehab, A. Security analysis of drone systems: Attacks, limitations and recommendations. *Internet Things J.* **2020**, *11*, 100218. [CrossRef]
7. McMahan, H.B.; Moore, E.; Ramage, D.; Hampson, S.; Arcas, B.A.y. Communication-efficient learning of deep networks from decentralized data. *Artif. Intell. Stat.* **2017**, *54*, 1273–1282.
8. Lv, D.; Cheng, X.; Zhang, J.; Zhang, W.; Zhao, W.; Xu, H. DDoS attack detection based on CNN and federated learning. In *2022 10th International Conference on Advanced Cloud and Big Data (CBD)*; IEEE: Piscataway, NJ, USA, 2022; pp. 236–241.
9. Zhang, J.; Yu, P.; Qi, L.; Liu, S.; Zhang, H.; Zhang, J. FLDDoS: DDoS attack detection model based on federated learning. In *IEEE International Conference on Trust, Security and Privacy in Computing and Communications*; IEEE: Piscataway, NJ, USA, 2021; pp. 635–642.
10. Hochreiter, S.; Schmidhuber, J. Long short term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]
11. Rezaei, S.; Liu, X. Deep learning for encrypted traffic classification: An overview. *IEEE Commun. Mag.* **2019**, *57*, 76–81. [CrossRef]
12. Rimmer, V.; Preuveneers, D.; Juarez, M.; Goethem, T.V.; Joosen, W. Automated website fingerprinting through deep learning. In *Network and Distributed System Security Symposium*; Internet Society: Reston, VA, USA, 2018; pp. 1–15.
13. Tian, Q.; Guang, C.; Wenchao, C.; Si, W. A lightweight residual networks framework for DDoS attack classification based on federated learning. In *IEEE International Conference on Computer Communications Workshops*; IEEE: Piscataway, NJ, USA, 2021; pp. 1–6.
14. Wang, J.; Liu, Y.; Feng, H.J.M.B. IFACNN efficient DDoS attack detection based on improved firefly algorithm to optimize convolutional neural networks. *Engineering* **2022**, *19*, 1280–1303. [CrossRef]
15. Ji, I.H.; Lee, J.H.; Kang, M.J.; Park, W.J.; Jeon, S.H.; Seo, J.T. Artificial intelligence-based anomaly detection technology over encrypted traffic: A systematic literature review. *Sensors* **2024**, *24*, 898. [CrossRef] [PubMed]
16. Lotfollahi, M.; Shirali Hossein Zade, R.; Jafari Siavoshani, M.; Saberian, M. Deep packet: A novel approach for encrypted traffic classification using deep learning. *Soft Comput.* **2020**, *24*, 1999–2012. [CrossRef]
17. Barradas, D.; Santos, N.; Rodrigues, L.; Madeira, A. FlowLens: Enabling efficient flow classification for ML-based network security applications. In *Network and Distributed System Security Symposium*; Internet Society: Reston, VA, USA, 2021; pp. 1437–1454.
18. Zhao, Z.; Li, Z.; Jiang, J.; Yu, F. ERNN: Error-resilient RNN for encrypted traffic detection towards network-induced phenomena. *IEEE Trans. Dependable Secur. Comput.* **2023**, *20*, 1–18.
19. Kim, M.; Kim, H. Anomaly detection in imbalanced encrypted traffic with few packet metadata-based feature extraction. *Comput. Model. Eng. Sci.* **2024**, *141*, 585–607. [CrossRef]
20. Li, J.; Zhou, H.; Wu, S.; Luo, X.; Wang, T.; Zhan, X.; Ma, X. FOAP: Fine-Grained Open-World Android App Fingerprinting. In *31st USENIX Security Symposium*; USENIX Association: Boston, MA, USA, 2022; pp. 1579–1596.
21. Ni, T.; Lan, G.; Wang, J.; Zhao, Q.; Xu, W. Eavesdropping Mobile App Activity via Radio-Frequency Energy Harvesting. In *32nd USENIX Security Symposium*; USENIX Association: Boston, MA, USA, 2023; pp. 3511–3528.
22. Fu, C.; Li, Q.; Xu, K. Detecting unknown encrypted malicious traffic in real time via flow interaction graph analysis. In *Network and Distributed System Security Symposium*; Internet Society: Reston, VA, USA, 2023; pp. 1–18.
23. Singh, A.; Jang-Jaccard, J. Autoencoder-based unsupervised intrusion detection using multi-scale convolutional recurrent networks. *arXiv* **2022**, arXiv:2204.03779.

24. Pourahmadi, V.; Alameddine, H.A.; Salahuddin, M.A.; Boutaba, R. Spotting anomalies at the edge: Outlier exposure-based cross-silo federated learning for DDoS detection. *IEEE Trans. Dependable Secur. Comput.* **2023**, *20*, 1–14. [[CrossRef](#)]
25. Li, J.; Zhang, Z.; Li, Y.; Guo, X.; Li, H. FIDS: Detecting DDoS through federated learning-based method. In *International Conference on Trust, Security and Privacy in Computing and Communications*; IEEE: Piscataway, NJ, USA, 2021; pp. 635–642.
26. Nguyen, T.D.; Marchal, S.; Miettinen, M.; Fereidooni, H.; Asokan, N.; Sadeghi, A.R. DIoT: A federated self-learning anomaly detection system for IoT. In *39th International Conference on Distributed Computing Systems*; IEEE: Piscataway, NJ, USA, 2019; pp. 756–767.
27. Zhao, R.; Wang, Y.; Xue, Z.; Ohtsuki, T.; Adebisi, B.; Gui, G. Semi-supervised federated learning-based intrusion detection method for internet of things. *IEEE Internet Things J.* **2023**, *10*, 8645–8657. [[CrossRef](#)]
28. Alabdulatif, A. GuardianAI: Privacy-preserving federated anomaly detection with differential privacy. *Array* **2025**, *26*, 100381. [[CrossRef](#)]
29. Korczyński, M.; Duda, A. Markov chain fingerprinting to classify encrypted traffic. In *IEEE Conference on Computer Communications*; IEEE: Piscataway, NJ, USA, 2014; pp. 781–789.
30. Shen, M.; Wei, M.; Zhu, L.; Wang, M. Classification of encrypted traffic with second order Markov chains and application attribute bigrams. *IEEE Trans. Inf. Forensics Secur.* **2017**, *12*, 1830–1843. [[CrossRef](#)]
31. Taylor, V.F.; Spolaor, R.; Conti, M.; Martinovic, I. AppScanner: Automatic fingerprinting of smartphone apps from encrypted network traffic. In *IEEE European Symposium on Security and Privacy*; IEEE: Piscataway, NJ, USA, 2016; pp. 439–454.
32. Taylor, V.F.; Spolaor, R.; Conti, M.; Martinovic, I. Robust smartphone app identification via encrypted network traffic analysis. *IEEE Trans. Inf. Forensics Secur.* **2017**, *13*, 63–78. [[CrossRef](#)]
33. Haripriya, R.; Khare, N.; Pandey, M. Privacy-preserving federated learning for collaborative medical data mining in multi-institutional settings. *Sci. Rep.* **2025**, *15*, 12482. [[CrossRef](#)] [[PubMed](#)]
34. Li, X.; Jiang, M.; Zhang, X.; Kamp, M.; Dou, Q. FedBN: Federated learning on non-IID features via local batch normalization. In *9th International Conference on Learning Representations*; OpenReview.net: Amherst, MA, USA, 2021.
35. Purohit, S.; Govindarasu, M.; Blakely, B. FL-ADS: Federated learning anomaly detection system for distributed energy resource networks. *IET Cyber-Phys. Syst. Theory Appl.* **2025**, *10*, e70001. [[CrossRef](#)]
36. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *4th International Conference on Information Systems Security and Privacy*; SCITEPRESS: Setubal, Portugal, 2018; pp. 108–116.
37. Sharafaldin, I.; Lashkari, A.H.; Hakak, S.; Ghorbani, A.A. Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy. In *2019 IEEE 53rd International Carnahan Conference on Security Technology*; IEEE: Chennai, India, 2019; pp. 1–8.
38. Khari, M.; Kumar, N. User authentication method against SQL injection attack. *Int. J. Sci. Eng. Res.* **2013**, *4*, 1649–1653.
39. Nanda, S.; Lam, L.; Chiueh, T. Web application attack prevention for tiered internet services. In *2008 Fourth Information Assurance and Security*; IEEE: Piscataway, NJ, USA, 2008; pp. 186–191.
40. Doriguzzi-Corin, R.; Siracusa, D. FLAD: Adaptive federated learning for DDoS attack detection. *Comput. Secur.* **2024**, *137*, 103597. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.