Li *et al. Cybersecurity* (2026) 9:112
https://doi.org/10.1186/s42400-025-00522-x

Cybersecurity

RESEARCH Open Access

MTDecipher: robust encrypted malicious traffic detection via multi-task graph neural networks

Fan Li[1,2], Xi Luo[3], Weihong Han[2], Binxing Fang[1*] and Lihua Yin[3*]

Abstract

The widespread adoption of encrypted traffic protocols has significantly increased the challenge of detecting malicious traffic. Existing detection methods based on deep learning typically rely on fine-grained features of data packets, such as length sequences and intra-flow interaction graphs. However, these features are highly susceptible to disruption by diverse network environments and traffic obfuscation. This paper proposes MTDecipher, a robust method for detecting encrypted malicious traffic based on multi-task Graph Neural Network (GNN). MTDecipher employs a bidirectional attentive sequence encoder to mitigate the impact of diverse network environments and traffic obfuscation on packet length sequences, along with an edge-block dual sampling method and a multi-task GNN model to mitigate the training bias introduced by the unbalanced distribution of traffic. In the bidirectional attentive sequence encoder, a combination of a Bi-GRU layer and an attention pooling layer is utilized to enhance the bidirectional encoding by generating weights for each element in the sequence, thereby obtaining robust encrypted traffic sequence features. In the edge-block dual sampling method, two rounds of sampling are involved to generate more evenly distributed subgraphs as training data, which reduces the local structural bias resulting from the aggregation of malicious flows. In the multi-task GNN model, the losses for both edge and node classification tasks are simultaneously optimized, thereby minimizing the homogeneity of adjacent edges. Experimental results on two real-world datasets with traffic obfuscation demonstrate that MTDecipher outperforms eight existing methods in terms of effectiveness in detecting encrypted malicious traffic.

Keywords Encrypted traffic detection, Malicious traffic detection, Sequence encoder, Subgraph sampling, Multi-task GNN

Introduction

As awareness of privacy protection among Internet users continues to increase, the adoption of encryption protocols to protect traffic content has become increasingly widespread. Protocols such as Transport Layer Security (TLS), Hypertext Transfer Protocol Secure (HTTPS), Internet Protocol Security (IPsec), and Domain Name System (DNS) over HTTPS (DoH), effectively mitigate privacy risks associated with the exposure of user traffic and personal data on the Internet. In this context, Google, as one of the largest browser providers, announced its commitment to implement full encryption across all its products and services, achieving the

*Correspondence:
Binxing Fang
fangbx@cae.cn
Lihua Yin
yinlh@gzhu.edu.cn
[1] School of Computer Science and Technology, Harbin Institute of Technology (Shenzhen), Shenzhen 518071, Guangdong, China
[2] Department of New Networks, Peng Cheng Laboratory, Shenzhen 518000, Guangdong, China
[3] Cyberspace Institute of Advanced Technology, Guangzhou University, Guangzhou 510006, Guangdong, China

completion of 96% by 2024 (Google Transparency Report 2025). Although such initiatives have garnered user approval for enhancing Internet security, they have also introduced new security challenges. Attackers are exploiting encryption protocols to conceal malicious activities. According to the Zscaler Zero Trust Exchange platform, encrypted traffic accounted for more than 85.9% of malicious traffic in 2023, with the total number of encrypted attacks increasing by 24.3% compared to the previous year (Zscaler ThreatLabz 2023). The payloads of this encrypted malicious traffic are invisible to traditional firewall and rule-based security systems, which are unable to filter them effectively. Moreover, decryption-based approaches are limited due to their high cost and potential violations of user privacy (Orchestrator 2024; Yang et al. 2024; Wang et al. 2025).

Malicious traffic detection systems based on deep learning techniques have been extensively studied (Xue and Zhu 2024). These methods typically input high-dimensional traffic features into carefully designed deep learning models, achieving notable success in plaintext malicious traffic detection. This success has inspired several studies to adapt deep learning models for encrypted traffic detection, aiming to decipher and extract meaningful information from encrypted traffic. A key gap between the two domains is that encryption protocols alter the range of features available for traffic classification. For instance, although the content payload is obfuscated after encryption, Barut et al. demonstrated that packet header bytes can still be leveraged for classifying encrypted traffic (Barut et al. 2022). In addition, flow statistics, such as packet length sequences, have been utilized to uncover encrypted traffic patterns (Liu et al. 2019). Many researchers have emphasized packet interactions within a network flow, which are considered as fingerprints for encrypted traffic detection (Yang et al. 2024). However, existing methods tend to overly rely on fine-grained features of captured packets, which are inherently unreliable. In the Transmission Control Protocol (TCP), for example, packet length and sequence are significantly influenced by diverse network environments. Since the sequence numbers of data packets are often encrypted, the packet sequence used in these studies is typically based on packet arrival times. However, network fluctuations, such as TCP packet replay, merging, and reconstruction (Xie et al. 2023), and traffic obfuscation, such as website fingerprinting (WF) defenses (Gong and Wang 2020, Wang and Goldberg 2017), can drastically alter the order and size of packets, leading to changes in flow statistics. Similarly, network fluctuations can induce structural changes in intra-flow interaction graphs, such as traffic interaction graph (TIG) (Shen et al. 2020) or malicious traffic interaction graph (MTIG) (Yang et al. 2024), rendering the methods ineffective.

Despite the fact that network fluctuations and traffic obfuscation diminish the reliability of fine-grained features, communication between attackers and victims remains unavoidable, and such interactions are typically resilient to network fluctuations. This insight reveals the potential for malicious intent leakage through inter-flow interaction features, which helps to detect encrypted malicious traffic. Graph neural networks (GNNs) are extensively utilized for processing inter-flow interaction graphs, where nodes represent IP addresses and edges represent flows. However, strong correlation and unbalanced distribution of malicious traffic can introduce significant bias into edge classification tasks performed by GNNs. On the one hand, existing methods usually overlook the potential influence of the node class on edge classification. The malicious behavior exhibited by attackers is intrinsically associated with specific IP addresses. Prior studies have demonstrated that inter-flow interaction behaviors can assist in identifying command and control (C&C) nodes within botnets, thereby facilitating the detection of malicious traffic originating from these nodes (Zhou et al. 2020). On the other hand, the homogeneity assumption adopted by GNNs does not hold true for inter-flow interaction graphs with unbalanced malicious flows. Traditional GNNs utilize a message-passing mechanism that promotes feature homogeneity among adjacent nodes or edges, operating under the assumption that neighboring nodes and edges are more likely to belong to the same class (Zhu et al. 2020). However, this assumption is inconsistent with empirical observations. As illustrated in Fig. 1, in the MCFP dataset (Stratosphere Laboratory Datasets 2015), IP addresses 10.0.2.110 and 8.8.8.8 are connected to a substantial number of malicious edges. This may lead to the erroneous conclusion that edges connected to specific nodes are necessarily malicious, introducing local structural bias (Dong et al. 2022) in GNNs. Such bias can result in information leakage and misclassification of edges associated with these IP addresses as malicious, consequently inflating the accuracy of the model. Additionally, this bias introduces errors in edge classification near IP address 224.0.0.252, which is associated with benign and malicious edges, with the malicious edges predominating. Network fluctuations further obscure the classification boundaries between benign and malicious edges, exacerbating local structural bias in GNNs and diminishing their effectiveness and robustness.

To address these challenges, we propose MTDecipher, a robust method for detecting encrypted malicious traffic based on a multi-task GNN. First, we select datasets collected from the real world to preserve the

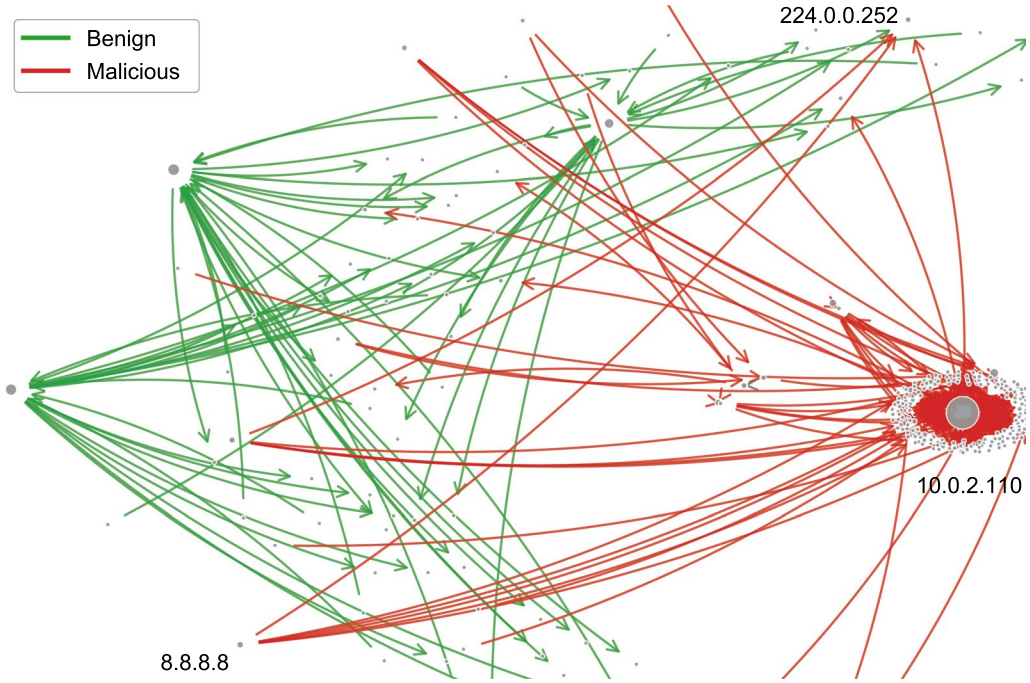

Fig. 1 Inter-flow Interaction Graph in MCFP Dataset

actual network fluctuations. Moreover, we simulate the traffic obfuscation from the attacker by incorporating website fingerprinting defenses into the original pcap file. The raw data packets are then preprocessed to generate robust sequential and non-sequential embeddings, which are aggregated into edge and node features. Balanced subgraphs for each batch are obtained using the edge-block dual sampling method. Finally, we employ a multi-task GNN for training and classification, which enhances the detection of encrypted malicious traffic. Our contributions are as follows:

- We propose a robust flow-to-edge approach based on a bidirectional attentive sequence encoder. This method constructs edge and node features within inter-flow interaction graphs under conditions of unknown network fluctuations.
- We introduce an edge-block dual sampling method that samples balanced subgraphs from inter-flow interaction graphs exhibiting unbalanced edge distributions, thereby mitigating training bias in multi-class GNNs.
- We present a multi-task GNN framework that simultaneously performs high-accuracy node and edge classification within inter-flow interaction graphs of encrypted traffic, enabling multi-class identification of malicious encrypted flows.
- We implement a prototype of MTDecipher and conduct performance tests. The results indicate that MTDecipher outperforms eight existing methods across two public datasets with website fingerprinting defenses.

Background and related work

In this section, we provide an overview of the encrypted malicious traffic and discuss the methods employed to detect malicious activities within such encrypted flows, with a particular focus on techniques that utilize GNNs. Additionally, we highlight their inherent limitations in diverse network environments.

Malicious traffic detection

In recent years, deep learning has emerged as a robust paradigm for malicious traffic detection, overcoming the limitations of traditional rule-based systems and machine learning methods when tackling complex and evolving network threats. For instance, Hassn et al. (2025) proposed a novel one-dimensional (1D) Convolutional Neural Network (CNN) for malicious traffic detection. While this approach exhibits strong capabilities in local feature extraction and generalization, it is inherently limited in capturing long-range temporal dependencies. Recurrent Neural Networks (RNNs) and their variants (e.g., Long

Short-Term Memory, LSTM; Gated Recurrent Unit, GRU) have been extensively utilized to model the temporal dynamics of network traffic. For example, Sasi et al. (2024) developed an IoT attack detection framework titled the self-attention-augmented 1D-CNN-LSTM. This hybrid architecture combines spatial feature extraction (via CNN) with temporal pattern modeling (via LSTM), with a self-attention mechanism to prioritize critical time steps. Although this design enhances the capture of time-series patterns, it still struggles to maintain robustness for long sequences. Furthermore, Transformer-based models have recently attracted increasing attention in malicious traffic detection due to their ability to model global dependencies across entire flow sequences (Du et al. 2025). By emphasizing long-range dependencies, these models can effectively identify subtle patterns in malicious traffic that are often overlooked by traditional methods.

Despite these advancements, existing approaches face significant challenges in handling encrypted malicious traffic, mainly attributable to payload obfuscation. The original payload becomes inaccessible, which narrows the range of extractable features. Notably, traditionally reliable statistical features (e.g., packet length distributions, inter-arrival time intervals, byte frequency profiles) become less robust under encryption. Encryption undermines their correlation with underlying traffic intent, rendering them unreliable indicators of malicious behavior. This drastic reduction in usable feature dimensions exacerbates the difficulty of training deep learning models, as fewer informative cues limit pattern learning and heighten the risk of overfitting.

Encrypted malicious traffic

Although the content of encrypted malicious traffic is invisible, it often exhibits analyzable traffic patterns. For example, to rapidly complete the distribution of encrypted keys and the transmission of control commands, Conti ransomware adopts short high-frequency connections and performs quick retries upon connection failures (Alzahrani et al. 2022). The encrypted crypto-mining malware XMRig sends a large number of oversized packets (>1500 bytes) containing mining programs (Kara and Hasgul 2023). The ransomware LockBit employs a fixed sequence of "short packet → long packet → short packet" for encrypted communication (Eliando and Warsito 2023). Taking TCP-based encrypted transmission protocols as an example, this paper introduces the impact of network environment changes on traffic patterns.

TCP is a connection-oriented and reliable transport layer protocol that does not inherently incorporate encryption, resulting in the transmission of data in plaintext. Given the complexities of network environments, data packets frequently arrive out of order, necessitating the synchronization of sequence numbers between the sender and the receiver. Furthermore, TCP implements flow control and congestion control mechanisms, such as sliding windows, to prevent the sender from overwhelming the receiver. It also employs timeout retransmission and fast retransmission mechanisms to ensure reliable data transfer, which are activated in response to packet loss, disorder, or delays within the network.

When transmitting sensitive information over TCP, encryption mechanisms are typically employed to ensure confidentiality and integrity. In the context of detecting encrypted malicious traffic, both the sequence number and packet payloads are encrypted, thereby obscuring the true sequence of packets from the Intrusion Detection System (IDS). Consequently, the IDS is influenced by dynamic variations in traffic patterns due to factors such as network congestion, receiver buffer saturation, and other conditions, even though the content of the messages remains unchanged. These variations manifest as alterations in network flow sequences, packet intervals, packet sizes, and other network characteristics, complicating the prediction of traffic patterns. As a result, conventional methods of traffic pattern analysis often demonstrate poor performance in detecting encrypted traffic.

It should be noted that, in addition to TCP, a range of novel encrypted traffic transmission protocols have been proposed, such as Quick UDP Internet Connections (QUIC) (Langley et al. 2017). These protocols can effectively reduce latency and improve security during encrypted traffic transmission. However, like TCP, the traffic patterns of these encrypted protocols undergo dynamic variations in response to changes in the network environment. This inherent dynamism renders them incompatible with traditional traffic pattern analysis methodologies.

Encrypted malicious traffic detection

Based on statistical features. Statistical features, such as packet length and packet count, serve as coarse-grained indicators of encrypted traffic. These features are relatively easy to collect and are commonly employed in traditional malicious traffic detection. For instance, Tegeler et al. (2012) applied clustering algorithms to analyze encrypted botnet control traffic using statistical features. Anderson and McGrew (2017) extracted 386 statistical features from packet length and TLS handshake metadata, utilizing these features to train a random forest-based detector. In the context of encrypted traffic communication in smart homes, Zhang et al. (2018)

employed a deterministic finite automaton (DFA), constructed with application control logic, to establish normal interaction patterns and detect anomalies based on side-channel information such as packet size and interval. Wang et al. (2022) synthesized frequent features from existing studies and applied machine learning algorithms to achieve promising results. Additionally, Et-BERT (Lin et al. 2022) and MTFlowFormer (Zhao et al. 2022) utilized natural language models to process encrypted traffic packet sequences, transferring knowledge from large unlabeled datasets to perform encrypted network traffic classification with labeled data.

Although these methods can achieve high accuracy on fixed datasets, their generalization performance in real-world environments is often limited. As discussed in Section II, the features employed by these methods are highly susceptible to network fluctuations, which can substantially degrade model performance (Xie et al. 2023).

Based on traffic fingerprints. The interactions of encrypted malicious traffic exhibit more discernible patterns. Traffic fingerprints are commonly used to identify such patterns. For example, Korczyński and Duda (2014) developed a Markov chain-based traffic fingerprint to identify encrypted application interactions, classifying encrypted traffic by comparing message sequences with pre-generated fingerprints. Liu et al. (2019) transformed encrypted traffic into a sequence of packet lengths and employed a multi-layer encoder-decoder to extract potential flow sequence embeddings.

In recent studies, researchers have increasingly employed flow interaction graphs, as opposed to traditional flow sequences, to generate fingerprints for encrypted traffic. These methods construct graphs based on packet interactions between senders and receivers, converting the encrypted flow classification task into a graph classification task. For instance, Shen et al. (2021) employed Traffic Interaction Graphs (TIG) as a comprehensive representation of encrypted flows, emphasizing multidimensional features in bidirectional client–server interactions. Fu et al. (2022) introduced ST-Graph to identify infected hosts within encrypted communications by constructing a heterogeneous graph that includes hosts and external servers. This graph generated edge embeddings from TLS handshake features, external domain name features, and side-channel statistics, which were subsequently aggregated into node embeddings for classification. Han et al. (2024) presented the Dual Embedding with Graph Neural Networks (DE-GNN) model, which encodes raw packet header and payload bytes, constructs TIGs, processes them with GNNs, and performs classification following feature fusion. Yang et al. (2024) developed MTSecurity, which constructs Malicious Traffic Interaction Graphs (MTIG) based on client–server interactions and employs Transformer and GNN techniques to extract both raw byte and graph-based interaction features, significantly improving the classification accuracy of encrypted malicious traffic. TGPrint (Wang et al. 2023) proposed an attack fingerprint recognition method utilizing time window graphs, where ports are treated as nodes, communication between attackers and victim hosts as edges, and communication features as edge attributes to generate attack graphs. Huoh et al. (2022) introduced an end-to-end GNN architecture that considers each data packet as a node, the timing relationships between packets as edges, and the time duration as edge weights, thereby constructing graphs and employing graph encoder-decoder networks for training and classification.

However, generating specific identifiers as fingerprints for each class of encrypted malicious traffic behaviors presents significant challenges in practice. As network environments evolve, each attack necessitates multiple fingerprints, which constrains the efficacy of fingerprint-based methods in detecting higher-order encrypted interactions, such as springboard attacks and command-and-control (C&C) channels.

Malicious traffic detection based on GNN

Graph Neural Networks (GNNs) have emerged as a powerful tool for learning on graph-structured data (Wu et al. 2020). GNNs aim to learn embeddings that incorporate both node and edge features, as well as the topological structure of the graph. This characteristic renders GNNs particularly effective for tasks such as node classification, link prediction, and graph classification.

A fundamental assumption underlying many successful GNN models is the Homophily Assumption (Ma et al. 2022), which posits that nodes connected by an edge are likely to share similar attributes or features. This assumption is central to the manner in which GNNs propagate information between nodes. Specifically, each node aggregates information from its neighbors in a way that emphasizes the similarity between connected nodes. This process facilitates the learning of rich node embeddings that capture not only the intrinsic features of the node but also the structure of its local neighborhood.

GNNs have been extensively employed for malicious traffic detection, particularly in the context of high-level malicious behaviors. For example, Altaf et al. (2024) developed a time-stamped multi-edge graph structure to uncover temporal patterns and hidden relationships in network flows, which are critical for recognizing botnet behaviors. Xu et al. (2024) designed an encoder to obtain graph embeddings that integrate the graph attention mechanism, and proposed a self-supervised method

based on graph contrastive learning. Zhao et al. (2020) utilized heterogeneous GNNs to construct weighted similarity graphs to characterize botnet features. Lo et al. proposed E-GraphSAGE (Lo et al. 2022) to detect botnet flows by integrating statistical flow features with edge embeddings. Zhu et al. (2022) evaluated the interpretability of GNN-based botnet detection models by calculating relevance scores between the model-reliant data and informative data.

Although these methods have demonstrated promising detection results, they fail to consider the strong correlation among malicious traffic behaviors and the inaccuracies introduced by unbalanced data distributions. The local graph structure bias generated by these features violates the homophily assumption of GNNs, resulting in shifts in node embeddings toward the mainstream class. This shift increases the likelihood that heterogeneous edges will be misclassified as belonging to mainstream classes. Furthermore, we note that due to the limitations of network simulation, most well-known public datasets for encrypted malicious traffic identification utilize fixed IP addresses to implement malicious behaviors and simulate attacks within small-scale host clusters. Notable examples include Darknet (2020), IDS17 (2017), IDS18 (2018), N-BaIoT (2018), BotIoT (2018), and UNSW-NB15 (Moustafa and Slay 2016). A common characteristic of these datasets is their abundance of flows coupled with a limited number of IP addresses. Consequently, these datasets frequently associate malicious behaviors with a small subset of IPs. Additionally, although most studies partition the flow set into training, validation, and test sets when utilizing these datasets, the global graph structure is employed for both model training and testing, which introduces potential data leakage issues.

The MTDecipher method proposed in this paper captures the interaction patterns between flows by constructing an inter-flow interaction graph and designing an edge encoder to mitigate the impact of network fluctuations on the packet sequence. Furthermore, the proposed edge-block dual sampling method and the multi-task GNN model address the training bias resulting from local graph structure bias, thereby achieving robust detection of encrypted malicious traffic.

Methodology

In this section, we introduce our proposed method, MTDecipher. MTDecipher is divided into four components, as illustrated in Fig. 2: preprocessing, graph generating, block sampling, and multi-task GNN.

In the preprocessing phase, we implement website fingerprinting defenses on the original pcap file, inserting virtual packets to simulate the traffic under real network fluctuations. During the graph generating phase, MTDecipher employs a bidirectional attentive sequence encoder to generate edge and node features, thus creating the inter-flow interaction graph. In the block sampling phase, MTDecipher generates balanced subgraphs for each batch using the edge-block dual sampling method. In the multi-task GNN phase, MTDecipher simultaneously optimizes the loss for both edge and node classification tasks while

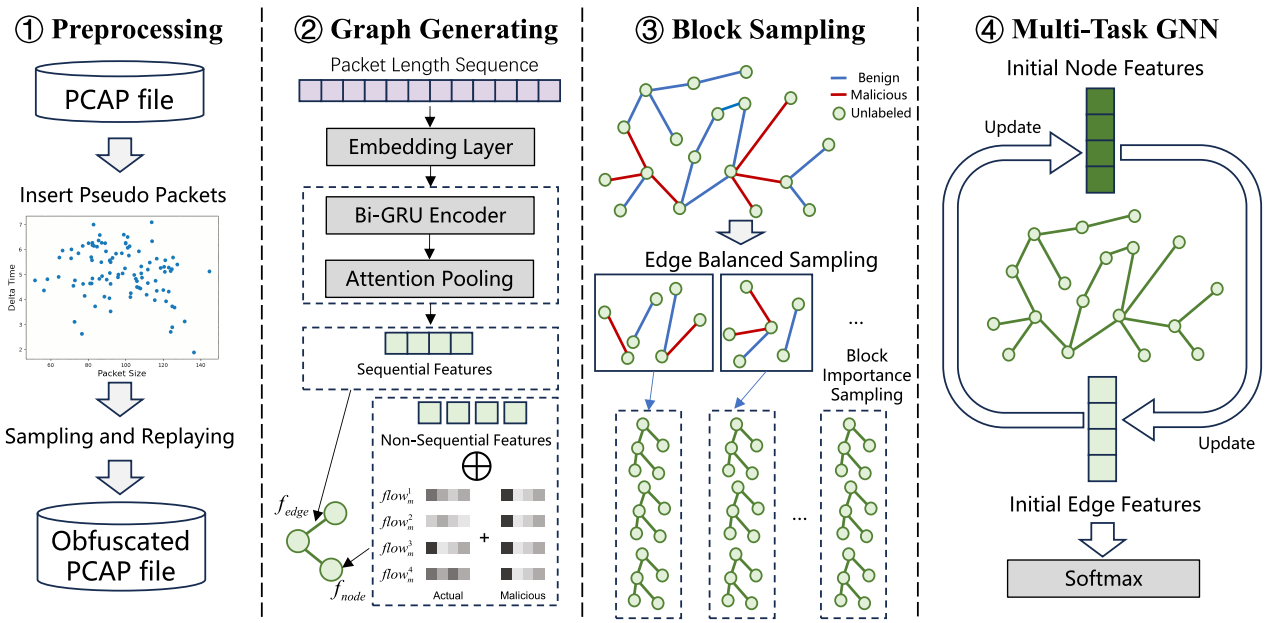

Fig. 2 Workflow of MTDecipher

training GNN model, thereby achieving accurate encrypted malicious traffic detection.

Preprocessing module

Website fingerprinting attacks (Cherubin et al. 2022) are techniques that exploit metadata from encrypted traffic to infer the website a user is visiting without decrypting the traffic. In contrast, the methods employed to counteract this attack are referred to as website fingerprinting defenses, which can also be used in encrypted traffic obfuscation by attackers (Shen et al. 2023). Similar to the approaches taken by Front (Gong and Wang 2020) and Walkie-Talkie (Wang and Goldberg 2017), we apply random delays to the original data packets and insert virtual data packets into the network flow to simulate fluctuations in the network environment and resist encrypted traffic classification. Specifically, based on the analysis results shown by Front (Gong and Wang 2020), we randomly select data packets and introduce random delays that conform to the Rayleigh distribution (Kundu and Raqab 2005). The purpose of this step is to enhance the background delays present in public datasets, thereby simulating a relatively harsh network transmission environment, rather than serving as the primary obfuscation method. It is important to note that the addition of random delays alters the sequence of data packets and modifies certain statistical features observed by the receiver. Subsequently, a specified number of virtual data packets are inserted at random positions within the flow. The size of these virtual data packets does not exceed that of the largest data packet in the flow and adheres to a Gaussian distribution (Folks and Chhikara 1978). Following the insertion of data packets, multiple features, such as the sequence of data packets, the number of data packets, and the average size of the data packets, are altered, thereby achieving a sufficiently strong obfuscation effect on the network flow.

Given a set of raw network flows $\mathcal{F} = \{f_i\}_{i=1}^{N}$, where each flow $f_i = \{p_k\}_{k=1}^{M_i}$ consists of M_i data packets, we define the website fingerprinting defenses Γ as a transformation that produces a modified flow $\tilde{f}_i$. The process is described by:

$$\tilde{f}_i = \Gamma(f_i) = \bigcup_{k=1}^{M_i} \left(\pi_{\text{delay}}(p_k)\right) \oplus \bigcup_{j=1}^{\Delta_i} v_j \tag{1}$$

where $\oplus$ denotes random interleaving of packets at arbitrary positions. Here, $p_k = (t_k, s_k)$ represents the original data packet, with t_k and s_k denoting the original timestamp and length of packet k, respectively. Additionally, v_j signifies the j-th virtual data packet, while Δ_i indicates the predetermined random number of virtual data packets. The delay function π_{delay} is defined as follows:

$$\pi_{\text{delay}}(p_k) = (t_k + \delta_k, s_k), \quad \delta_k \sim \text{Rayleigh}(\lambda) \tag{2}$$

where $\delta_k > 0$ represents the Rayleigh-distributed random delay, and λ is the scale parameter of the Rayleigh distribution.

This process yields an obfuscated flow $\tilde{f}_i$, which comprises both shuffled packets and newly inserted virtual packets. The effectiveness of the website fingerprinting defense can be modulated by adjusting the delay function and the number of virtual packets.

Graph generation module

After preprocessing the raw pcap file, we first convert network traffic into a graph structure to facilitate processing by GNNs. Specifically, we define source and destination IP addresses in the traffic as nodes, and model the corresponding network flows as edges connecting these nodes, thereby constructing the inter-flow interaction graph. Notably, since most datasets only provide features and labels for network flows, the initial graph only contains edge features and labels. To address this gap, we aggregate edge features and labels to derive node-level features and labels. These nodes, edges, and their respective features and labels are then input to the GNN model for training. In this way, network traffic analysis is transformed into a joint node and edge classification task for the GNN.

Unlike existing works that directly use binary network traffic as edge features, we extract the packet length sequence of each flow to generate sequential features. We then combine these sequential features with network flow meta-features [Proto,Ext_Proto,Sport,Dport] (treated as non-sequential features) to form node and edge features. These features are protocol-independent and can be readily extracted from pcap files generated using encrypted transmission protocols, such as HTTPS and QUIC. This ensures the generalizability of the proposed method to modern encrypted threats. We propose a bidirectional attentive sequence encoder designed to map an input sequence of packet lengths to an embedding vector. The structure of the encoder is illustrated in Fig. 3, which includes an embedding layer, a Bi-GRU encoder layer, an attention pooling layer, and a classification layer. The goal is to efficiently capture the temporal dependencies and structural patterns within the sequence, thus generating a meaningful representation. In this section, we present a detailed overview of each component and describe how features of nodes and edges in the initial inter-flow interaction graph are aggregated.

Embedding layer

For each network flow, we extract its raw packet length sequence $S_{\text{raw}} \in \mathbb{Z}^N$, where both forward and reverse

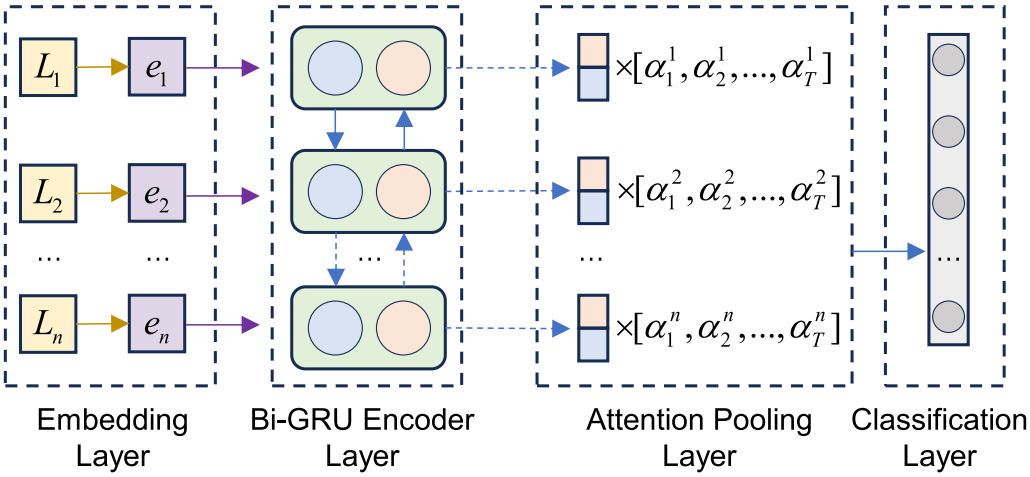

Fig. 3 The Bidirectional Attentive Sequence Encoder of MTDecipher

packet lengths are represented by positive values. For a packet $p_i \in \mathcal{P}$, its signed length $\ell(p_i) \in \mathbb{Z}$ is defined as $\ell_i = |p_i|$, where $|p_i|$ denotes the byte length of packet p_i. The packet length sequence of flow f_k is $L_k = \{\ell_1, \ell_2, \ldots, \ell_N\}$.

We set the maximum sequence length $|L| = 200$, applying zero-padding to sequences shorter than L and truncating those that exceed this length. Consequently, we obtain a normalized sequence $S_{\text{pad}} \in \mathbb{Z}^{|L|}$. This sequence is then projected into a dense vector representation through an embedding layer:

$$\mathbf{E} = \text{Embedding}(S_{\text{pad}}) = [\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_N]^\top \in \mathbb{R}^{N \times d} \tag{3}$$

where $\mathbf{e}_k$ denotes the embedding of L_k, $\mathbf{E}$ represents the collection of all vectors $\mathbf{e}_k$ formed by the sequence of packet lengths, and d is the predetermined embedding dimension. This representation serves as the initial feature input for the Bi-GRU encoder layer.

Bi-GRU encoder layer

We employ a Bidirectional Gated Recurrent Unit (Bi-GRU) to capture both forward and backward contextual dependencies in the embedding vector $\mathbf{e}_k = \sum_{t=1}^{T} \boldsymbol{e}_k^t$, as described by the following equations:

$$\overrightarrow{\boldsymbol{h}_k^t} = \text{GRU}_{\text{forward}}\left(\boldsymbol{e}_k^t, \overrightarrow{\boldsymbol{h}_k^{t-1}}\right) \tag{4}$$

$$\overleftarrow{\boldsymbol{h}_k^t} = \text{GRU}_{\text{backward}}\left(\boldsymbol{e}_k^t, \overleftarrow{\boldsymbol{h}_k^{t+1}}\right) \tag{5}$$

$$\boldsymbol{h}_k^t = \left[\overrightarrow{\boldsymbol{h}_k^t} \parallel \overleftarrow{\boldsymbol{h}_k^t}\right] \in \mathbb{R}^{2m} \tag{6}$$

where $\boldsymbol{e}_k^t \in \mathbb{R}^d$ represents the embedding vector at timestep t obtained from the embedding layer. The forward and backward hidden states at timestep t are denoted as $\overrightarrow{\boldsymbol{h}_k^t}, \overleftarrow{\boldsymbol{h}_k^t} \in \mathbb{R}^m$. The parameter m indicates the hidden dimension of the Bi-GRU. The concatenated context-aware representation, $\boldsymbol{h}_t \in \mathbb{R}^{2m}$, serves as the output of this layer.

Attention pooling layer

We employ an attention mechanism to adaptively reweight the temporal features in the sequence, thereby emphasizing salient patterns. The formulation is as follows:

$$\boldsymbol{u}_t = \tanh(\boldsymbol{W}_a \boldsymbol{h}_k^t + \boldsymbol{b}_a) \tag{7}$$

$$\alpha_t = \frac{\exp(\boldsymbol{u}_t^\top \boldsymbol{v}_a)}{\sum_{j=1}^{T} \exp(\boldsymbol{u}_j^\top \boldsymbol{v}_a)} \tag{8}$$

$$\boldsymbol{z}_k = \sum_{t=1}^{T} \alpha_t \boldsymbol{h}_k^t \in \mathbb{R}^{2m} \tag{9}$$

where $\boldsymbol{W}_a$, $\boldsymbol{b}_a$, and $\boldsymbol{v}_a$ are the learnable parameters. Specifically, $\boldsymbol{W}_a \in \mathbb{R}^{d_a \times 2m}$ is the attention weight matrix, where d_a represents the attention dimension. The vector $\boldsymbol{b}_a \in \mathbb{R}^{d_a}$ is the attention bias vector, and $\boldsymbol{v}_a \in \mathbb{R}^{d_a}$ is the context vector used for computing attention scores. The normalized importance weight $\alpha_t \in [0, 1]$ satisfies the constraint $\sum_{t=1}^{T} \alpha_t = 1$. Finally, $\boldsymbol{z}_k \in \mathbb{R}^{2m}$ is the context vector that aggregates information from the sequence. The attention layer computes $\boldsymbol{u}_t$ as a non-linear projection of $\boldsymbol{h}_t$ and scores each timestep through a dot product with the context vector $\boldsymbol{v}_a$.

The context vector undergoes a nonlinear transformation followed by normalization, as described by the equations below:

$$\boldsymbol{z}_k' = \text{LeakyReLU}(\boldsymbol{W}_e \boldsymbol{z}_k + \boldsymbol{b}_e) \in \mathbb{R}^d \tag{10}$$

$$\boldsymbol{z}_k'' = \text{LayerNorm}(\boldsymbol{z}_k') \in \mathbb{R}^d \tag{11}$$

where $\boldsymbol{z}_k'$ represents the result of the nonlinear transformation applied to $\boldsymbol{z}_k$, while $\boldsymbol{z}_k''$ denotes the outcome of the normalization process. The matrix $\boldsymbol{W}_e \in \mathbb{R}^{d \times 2m}$ serves as the embedding transformation weight matrix, and $\boldsymbol{b}_e \in \mathbb{R}^d$ is the corresponding transformation bias. The LeakyReLU activation function is defined with a slope of 0.01 for negative inputs, expressed as $\max(0.01x, x)$. The LayerNorm function stabilizes activations by normalizing across features. This stage effectively projects $\boldsymbol{z}_k$ back to the original embedding dimension d, thereby enhancing its representational capacity.

Classification layer

The final embedding is computed as follows:

$$\boldsymbol{o}_k = \boldsymbol{W}_c \boldsymbol{z}_k'' + \boldsymbol{b}_c \in \mathbb{R}^C \tag{12}$$

$$\hat{\boldsymbol{y}_k} = \text{softmax}(\boldsymbol{o}_k) \tag{13}$$

where $\boldsymbol{W}_c \in \mathbb{R}^{C \times d}$ is the classification weight matrix, and $\boldsymbol{b}_c \in \mathbb{R}^C$ is the classification bias. The vector $\boldsymbol{o} \in \mathbb{R}^C$ represents the unnormalized logits, while $\hat{\boldsymbol{y}_k} \in [0,1]^C$ denotes the class probability vector, satisfying the condition $\sum_{c=1}^{C} \hat{y}_k^c = 1$.

To mitigate the overfitting during training and to enhance the generalization ability of the model, we employ label-smoothed cross-entropy (Müller et al. 2019):

$$\mathcal{L} = -\sum_{c=1}^{C} \left[(1-\epsilon) \cdot \mathbb{I}(y=c) + \frac{\epsilon}{C} \right] \log(\hat{y}_c) \tag{14}$$

where y represents the true class label, which is one-hot encoded. C denotes the total number of classes, and $\epsilon \in [0,1]$ is the smoothing hyperparameter, typically set to $\epsilon = 0.1$. The function $\mathbb{I}(\cdot)$ is the indicator function. Label smoothing helps to prevent overconfidence by redistributing ϵ probability mass uniformly across all classes.

Edge and node feature aggregation

Following the generation of traffic embeddings, we proceed to aggregate edge features and node features. The edge features are predicted by a pre-trained Bi-GRU model for multi-classification tasks:

$$\mathbf{h}_{ij}^k = \boldsymbol{o}_k = BiGRU(L_k) \tag{15}$$

where L_k represents the k-th flow, and $\mathbf{h}_{ij}^k$ is the embedding corresponding to the k-th flow, which occurs between nodes v_i and v_j. The edge feature $\mathbf{e}_{ij}$ can be expressed as follows:

$$\mathbf{e}_{ij} = \sum_{k \in \mathcal{E}(ij)} w_{ij} \mathbf{h}_{ij}^k \tag{16}$$

where w_{ij} denotes the weight of the edges associated with nodes v_i and v_j, with the constraint $\sum_{j \in \mathcal{N}(i)} w_{ij} = 1$. The weight assigned to malicious edges is twice that of benign edges. It is important to recognize that multiple flows may exist between two nodes in the graph, and $\mathbf{e}_{ij}$ represents the aggregation of flow embeddings between v_i and v_j.

We define the source nodes and target nodes of predicted malicious edges as potential malicious nodes. In the process of node feature aggregation, the weight of the predicted malicious edge is increased and normalized based on the degree of the node. Additionally, we incorporate non-sequential features into the node feature. The node feature $\mathbf{v}_i$ can be expressed as follows:

$$\mathbf{O} = \text{OneHot}([\text{Proto}, \text{Ext\_Proto}, \text{Sport}, \text{Dport}]) \tag{17}$$

$$\mathbf{e}_{ij}^{'} = \sum_{k \in \mathcal{E}(ij)} w_{ij} (\mathbf{h}_{ij}^k \oplus \mathbf{O}_{ij}^k) \tag{18}$$

$$\mathbf{v}_i = \frac{1}{\deg(v_i)} \sum_{j \in \mathcal{N}(i)} \mathbf{e}_{ij}^{'} \tag{19}$$

where $\mathbf{O}$ represents the selected non-sequential feature after one-hot encoding. The term v_i denotes the i-th node, and $\deg(v_i)$ indicates the degree of node v_i. When calculating the initial features of the nodes, we only aggregate the features of all edges connecting to their first-order neighbors.

Block sampling module

Upon acquiring the graph structure, along with the node and edge features, the graph is input into the edge classification GNN. Due to the imbalance in the dataset and the graph topology, the GNN model tends to over-predict the majority class while significantly neglecting the prediction performance of the minority class after training. The impact of this training bias on performance is discussed in Section IV. To mitigate this training bias, we propose an edge-block dual sampling method.

Given a graph structure $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ represents the set of nodes and $\mathcal{E}$ denotes the set of edges, along with the associated node features $\mathbf{v}_i$ and edge features $\mathbf{e}_{ij}$,

our objective is to train a multi-task GNN model for classifying nodes and edges in the presence of class imbalance, the training process is shown in Algorithm 1. The edge-block dual sampling method is divided into two components: edge balanced sampling and block importance sampling.

Algorithm 1 MTDecipher Training Process

Input:
  Inter-flow interaction graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with normal edge set E_{normal} and malicious edge set $E_{\text{malicious}}$
  Hyperparameters: sampling range C, training epochs T, balance coefficient α, batch size B
  Model architecture f_θ with dual classification heads
Output:
  Optimal parameters θ^*
  Edge classification F1-score $F1_{\text{edge}}$, node classification F1-score $F1_{\text{node}}$
INITIALIZATION
  Sampler $\leftarrow$ MultiLayerImportanceNeighbor(C)
  $(F1_{\text{best}}, \theta_{\text{best}}, t_{\text{best}}) \leftarrow (0, \emptyset, 0)$
for $t \leftarrow 1$ to T **do**
    $E_{\text{batch}} \leftarrow \text{EdgeSampler}(E_{\text{normal}}, E_{\text{malicious}}, B)$
    $V_{batch} \leftarrow \text{GetNodes}(E_{batch})$
    Dataloader $\leftarrow$ EdgeLoader($\mathcal{G}, E_{\text{batch}}$, BlockSampler)
    for each *Block* **in** Dataloader **do**
      $(Y_{\text{edge}}^{\text{pred}}, Y_{\text{node}}^{\text{pred}}) \leftarrow f_\theta(Block)$
      $\mathcal{L}_{edge} \leftarrow \text{CrossEntropy}(Y_{edge}^{pred}, \mathcal{G}_{block}.Y_{edge})$
      $\mathcal{L}_{node} \leftarrow \text{CrossEntropy}(Y_{node}^{pred}, \mathcal{G}_{block}.Y_{node})$
      $\mathcal{L}_{\text{total}} \leftarrow \alpha\mathcal{L}_{\text{node}} + (1-\alpha)\mathcal{L}_{\text{edge}}$
      $\nabla_\theta \mathcal{L}_{total} \leftarrow \text{Backpropagate}(\mathcal{L}_{total})$
      $\theta \leftarrow \text{OptimizerStep}(\theta, \nabla_\theta \mathcal{L}_{total})$
    end for
    $(F1_{edge}, F1_{node}) \leftarrow \text{Evaluate}(f_\theta, G_{test})$
    if $F1_{\text{edge}} + F1_{\text{node}} > F1_{\text{best}}$ **then**
      $F1_{\text{best}} \leftarrow F1_{\text{edge}} + F1_{\text{node}}$
      $\theta_{\text{best}} \leftarrow \theta$
      $t_{\text{best}} \leftarrow t$
    end if
end for

Edge balanced sampling

We partition all edges $\mathcal{E}$ into a training set $\mathcal{E}_{train}$, a validation set $\mathcal{E}_{val}$, and a test set $\mathcal{E}_{test}$. Uniform sampling is performed exclusively on the training set to ensure that the sampling process does not cause data leakage. The number of benign edges and malicious edges obtained from each sampling is kept equal, treating these edges as a balanced edge sampling block. Since all edges are considered distributed during the sampling process, the edges within the obtained sampling blocks may not necessarily be connected to each other.

Block random sampling

In this phase, we consider all nodes contained within the sampling block and perform importance sampling of the n-th order neighbors of these nodes. We use the Jaccard similarity (Bag et al. 2019) to measure the degree of overlap among the neighbors of nodes, thereby increasing the probability of obtaining locally associated subgraphs. For an edge (u, v) in the graph, the Jaccard similarity between the source node u and the destination node v is defined as:

$$\mathcal{S}_{\text{Jaccard}}(u, v) = \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|} \tag{20}$$

where $N(u)$ denotes the neighbor set of node u, and $|\cdot|$ the number of elements in a set. We assign this similarity value to the edge (u, v). For all adjacent edges (u, v) of node u, the similarity values need to be converted into a sampling probability distribution, which is given by:

$$p(u \to v) = \frac{\mathcal{S}(u, v)}{\sum_{v' \in N(u)} \mathcal{S}(u, v')} \tag{21}$$

where $\mathcal{S}(u, v)$ denotes the similarity between nodes u and v. The sampled neighbor nodes and their corresponding edges are then added to the sampling block. Consequently, the nodes and edges present in the sampling block form the sampled subgraph set $\mathcal{G}_{block}$. Each batch is trained in $\mathcal{G}_{block}$. Although block importance sampling helps capture local structures, it is important to note that due to the random sampling of edges, the subgraphs contained within a sampling block are not necessarily connected and are more likely to manifest as multiple distributed subgraphs. This approach helps to avoid local structural bias caused by the aggregation of malicious edges.

The edge-block dual sampling method encourages the model to learn more generalized features, thereby reducing the risk of overfitting to specific local patterns within the graph.

Multi-task GNN module

The multi-task GNN model comprises an n-layer GraphSAGE (Hamilton et al. 2017), utilizing mean as the feature aggregation method and ReLU as the activation function. The update process for the node embeddings can be expressed as follows:

$$h_u^{(l+1)} = \text{ReLU}\left(W^{(l)} \cdot \text{Mean}(\{h_v^{(l)} | v \in \mathcal{N}(u)\}) + b^{(l)}\right) \tag{22}$$

where $h_u^{(l+1)}$ denotes the embedding of node u in the $l+1$ layer, and $\mathcal{N}(u)$ represents the neighboring nodes of node u. The weight matrix for the l layer is denoted as $W^{(l)}$, and $b^{(l)}$ is the bias term. The edge-block dual sampling method is employed to construct subgraph blocks for each training batch.

In contrast to edge embeddings, which capture sequential features, node embeddings concatenate sequential and non-sequential features, resulting in a mismatch between the two types of embeddings. To address this issue, we design a two-layer multi-layer perceptron (MLP) to align the embeddings. The first layer, referred to as the node feature fusion layer, concatenates the embeddings of the source and destination nodes h_u, h_v, and maps them into a vector $\mathbf{e_1} \in \mathbb{R}^C$:

$$\mathbf{e_1} = \text{MLP}_1([h_u, h_v]) \tag{23}$$

The second layer, known as the edge feature fusion layer, concatenates the generated vector $\mathbf{e_1}$ with the original edge feature vector, mapping it to a specified dimension (which is set to the number of edge classes in the model):

$$\mathbf{e_2} = \text{MLP}_2([\mathbf{e_1}, \mathbf{e}_{uv}]) \tag{24}$$

where $\mathbf{e}_{uv}$ represents the original feature of the edge connecting nodes u and v. The vector $\mathbf{e_2}$ serves as the final edge embedding, which is subsequently fed into the softmax layer for multi-class edge classification. Additionally, the node feature h_u is input into a separate softmax layer for binary classification. The process can be expressed as:

$$\hat{y}_{\text{edge}} = \text{Softmax}(\mathbf{e_2}) \tag{25}$$

$$\hat{y}_{\text{node}} = \text{Softmax}(h_u) \tag{26}$$

where $\hat{y}_{\text{edge}}$ and $\hat{y}_{\text{node}}$ denote the predicted classes for edges and nodes, respectively. During model training, only the node features are modified, while new edge features are aggregated for classification based on the updated node features. All parameters are trained simultaneously, resulting in the generation of both node and edge embeddings.

Given the true labels for both edges and nodes, denoted as y_{edge} and y_{node}, we can compute the loss for the node

classification task and the edge classification task. The cross-entropy loss for either node or edge classification is expressed as follows:

$$\mathcal{L} = -\log\left(\frac{\exp(z_y)}{\sum_{c=1}^{C}\exp(z_c)}\right) \tag{27}$$

where z_y represents the unnormalized logit of the true class y for the sample, and z_c is the unnormalized logit for class c. Here, C denotes the total number of classes. The overall loss for the model is calculated as:

$$\mathcal{L}_{total} = \alpha \cdot \mathcal{L}_{node} + (1-\alpha) \cdot \mathcal{L}_{edge} \tag{28}$$

where $\mathcal{L}_{node}$ is the loss associated with the node classification task, $\mathcal{L}_{edge}$ is the loss for the edge classification task, and α is a hyperparameter that controls the relative weight of the two losses. The value of $\alpha \in [0.1, 0.9]$ obtained via grid search with 0.1 step is 0.8, which enables more stable training.

During the optimization process, the optimal model is determined by maximizing the weighted sum of the F1 scores for both node and edge classifications. Subsequently, all model parameters, including those of the multi-task GNN and MLP components, are optimized jointly. This training process iteratively updates node representations and recomputes edge features, thereby enhancing classification performance for both nodes and edges concurrently.

Evaluation

In order to evaluate MTDecipher, we developed a proof-of-concept implementation for comparison with previous results. All experiments were conducted on a commercial machine equipped with an RTX 4070 GPU, and an Intel i5-12600 CPU operating at a frequency of 4.80 GHz.

A. Dataset

Despite the availability of numerous datasets for encrypted traffic detection, many of these datasets suffer from significant limitations, particularly in terms of completeness. For example, some datasets provide only preprocessed feature sets in labeled CSV files without accompanying pcap files, while others consist solely of unlabeled pcap files. Additionally, datasets designed for simulation purposes often include an inadequate number of malicious nodes exhibiting malicious behaviors, resulting in network scales that are too limited for effective graph learning.

To mitigate these issues, we selected the USTC-TFC and MCFP datasets. Both datasets capture real-world network traffic and offer labeled pcap files that include comprehensive IP address metadata. An overview of these datasets is provided in Table 1.

USTC-TFC (Anderson and McGrew 2017) comprises pcap files for ten types of malware traffic (e.g., Cridex, Miuref, Virut) and ten types of benign traffic sourced from public websites. For our analysis, we consolidated the benign traffic into a single benign class.

MCFP (Stratosphere Laboratory Datasets 2015) is collected by the Stratosphere Lab's Malware Capture Facility Project, which is responsible for long-term malware and normal traffic captures. This dataset includes 342 malware captures and 21 normal captures, along with the original pcap files and a password-protected ZIP file containing the binary files used for infection. For our study, we selected ten malware captures related to encrypted traffic as the malicious class and merged five normal captures into the benign class. An overview of our reconstructed MCFP dataset is presented in Table 2.

B.Baselines

To demonstrate the effectiveness of MTDecipher, we conducted a comparative analysis of various existing studies focused on encrypted traffic detection.

FS-net (Liu et al. 2019) is an end-to-end recurrent neural network (RNN) tailored for the classification of

Table 2 Composition of Selected Subsets from MCFP Dataset

Label	Flow	IP	MCFP ID	Size(MB)
Benign	23,490	2,112	CTU-Normal	84.9
Artemis	8,238	24	275-1	8.2
Bunitu	17,054	1,226	141-1	35.7
CCleaner	138,725	111	320-2	82.5
Dridex	18,742	65	322-1	17.8
Emotet	19,111	35	264-1	12.0
HTBot	8,209	307	111-1	10.1
Razy	4,526	11	274-1	3.8
TrickBot	11,768	33	240-1	12.8
Ursbif	25,750	38	313-1	27.1
Zeus	25,092	36	25-6	28.1

Table 1 Dataset Statistics

Dataset	Total IP	Multiple IP	Total Flow	Total IP Pairs	Benign Flow	Benign IP Pair	Malicious Flow	Malicious IP Pair	Class
MCFP	3,591	299	300,705	4,435	23,490	2,677	277,215	1,759	11
USTC-TFC	137,234	128	375,678	288,344	282,412	279,713	93,266	8,631	11

encrypted traffic. Its multilayer encoder-decoder architecture is adept at capturing sequential flow patterns, enhanced by a reconstruction mechanism that optimizes feature representation.

FlowBERT (Pan et al. 2023) employs a Transformer-based architecture for the classification of encrypted traffic. It extracts semantic features from both payload content and packet length sequences, facilitating efficient learning of spatiotemporal traffic patterns.

DeepPacket (Lotfollahi et al. 2020) is a deep learning-based approach which integrates both feature extraction and classification. It employs stacked autoencoder (SAE) and CNN for encrypted traffic analysis.

TSCRNN (Lin et al. 2021) is an encrypted traffic classification method that leverages spatiotemporal features to automatically extract discriminative features for efficient classification. It uses a CNN to extract abstract spatial features and introduces a stacked bidirectional LSTM to learn temporal features.

Dapp (Shen et al. 2021) approaches encrypted decentralized application fingerprinting as a graph classification problem. It constructs Traffic Interaction Graphs (TIGs) to depict flow interactions and utilizes a GNN-based classifier for traffic analysis.

MT-Security (Yang et al. 2024) features a hybrid Transformer-GNN architecture aimed at detecting encrypted malicious traffic. This model enhances classification accuracy by constructing Malicious Traffic Interaction Graphs (MTIGs) that model traffic relationships.

EC-GCN (Diao et al. 2023) is a deep learning framework for encrypted traffic classification based on multi-scale graph convolutional neural networks (GCNs). It encodes each encrypted traffic flow into graph representations, which are then fed into a graph pooling and structure learning layer to learn representative spatiotemporal traffic features.

E-graphsage (Lo et al. 2022) is a GNN model designed for intrusion detection in Internet of Things (IoT) networks. This model effectively integrates edge features and topological information derived from flow-based data to create expressive graph representations of network traffic.

C. Evaluation Metrics

We evaluate MTDecipher against the control group using a comprehensive set of classification metrics: Accuracy, Precision, Recall, F1-Score, False Positive Rate (FPR), and False Negative Rate (FNR). These metrics are defined as follows:

$$\text{Accuracy (ACC)} = \frac{TP + TN}{TP + TN + FP + FN} \tag{29}$$

$$\text{Precision (PRE)} = \frac{TP}{TP + FP} \tag{30}$$

$$\text{Recall (REC)} = \frac{TP}{TP + FN} \tag{31}$$

$$\text{F1-Score} = 2 \times \frac{\text{PRE} \times \text{REC}}{\text{PRE} + \text{REC}} \tag{32}$$

$$\text{FPR} = \frac{FP}{FP + TN} \tag{33}$$

$$\text{FNR} = \frac{FN}{TP + FN} \tag{34}$$

where TP, TN, FP, and FN denote true positives, true negatives, false positives, and false negatives, respectively.

To address class imbalance, we compute macro-averaged weighted metrics across C classes, utilizing class-specific measures weighted by sample count n_i:

$$\text{Average Metric} = \frac{\sum_{i=1}^{C}(n_i \cdot \mathcal{M}_i)}{\sum_{i=1}^{C} n_i} \tag{35}$$

where $\mathcal{M}_i$ represents the class-specific value for any given metric (F1, PRE, REC, FPR and FNR).

D. Influence of Local Structural Biases

IP addresses directly associated with both benign and malicious flows are what we refer to as "multiple IPs". As highlighted in the introduction, local structural biases in GNNs arise from the violations of the homogeneity assumption. Multiple IPs are a core source of such bias. Their neighbors include both benign and malicious nodes, causing GNNs' message-passing mechanisms to propagate noisy features and leading to erroneous edge classifications.

Table 1 presents the proportion of multiple IPs in the two datasets: 8.33% in the MCFP dataset and 0.09% in the USTC-TFC dataset. Since multiple IPs constitute only a small fraction of the total number of IPs, removing them will not fundamentally change the graph structure. Moreover, it can ensure that there is no local structural bias caused by potentially malicious nodes connecting benign edges in the processed graph, making their removal a necessary step to isolate the impact from local structural bias.

To verify how this bias affects GNN-based methods, we designed controlled experiments: (1) retaining all IPs (original setting), (2) removing only multiple IPs ("noBias", to eliminate the target bias), and (3) randomly removing the same number of non-multiple IPs ("AvgR", to rule out confounding effects from reduced node count.

Table 3 The Average Edge Classification Performance Metrics of GNN-based Models in MCFP and USTC-TFC without Traffic Obfuscation

Method	MCFP						USTC-TFC					
	ACC	F1	PRE	REC	FPR	FNR	ACC	F1	PRE	REC	FPR	FNR
E-GraphSAGE-AvgR	0.9532	0.9026	0.9698	0.8441	0.0049	0.1559	0.9685	0.9348	0.9872	0.8877	**0.0023**	0.1123
E-GraphSAGE-noBias	**0.9738**	**0.9321**	**0.9802**	**0.8885**	**0.0028**	**0.1115**	**0.9692**	**0.9389**	**0.9885**	**0.8940**	0.0025	**0.1060**
MTDecipher-AvgR	0.9996	0.9935	0.9972	0.9898	0.0007	0.0102	0.9997	0.9988	0.9985	0.9991	**0.0004**	0.0009
MTDecipher-noBias	**0.9999**	**0.9983**	**0.9980**	**0.9986**	**0.0005**	**0.0014**	**0.9999**	**0.9990**	**0.9986**	**0.9994**	**0.0004**	**0.0006**

It is a critical control to ensure performance changes stem from bias elimination rather than reduced data size).

We used the original dataset for model training and testing. Experimental results are presented in Table 3, where the boldface values denote the best performance results between the two treatments for each dataset, across the two methods. The "noBias" group outperforms the original setting across nearly all metrics, confirming that local structural bias driven by multiple IPs undermines GNN performance. The "AvgR" group shows no comparable improvement, ruling out "reduced node count" as a confounding factor and validating that gains in "noBias" stem from eliminating multiple IPs' specific bias.Notably, as the MCFP dataset contains a higher proportion of multiple IPs, both models exhibit more pronounced performance gains. These experimental results demonstrate that the local structural bias caused by the unbalanced distribution of malicious traffic exerts a negative impact on GNN-based models.

E. Influence of Traffic Obfuscation

To assess the impact of traffic obfuscation intensity on existing methods for detecting encrypted malicious traffic, we introduced perturbations into the original pcap files from the MCFP and USTC-TFC datasets. This process involved two primary modifications:

1) Packet Timing Perturbation: For each flow, 50% of the packets were randomly selected and subjected to random delays that followed a Rayleigh distribution. The maximum delay for any packet was limited to 10% of the total duration of the flow, thereby simulating realistic network fluctuations and packet reordering.

2) Virtual Packet Injection: Virtual packets, each smaller than the maximum packet length of the flow, were inserted into each flow. The number of injected packets per flow was randomly chosen from the range $[0, n]$, where $n \in \{0, 3, 5, 7, 9, 11\}$. This parameter n controlled the intensity of obfuscation, resulting in the generation of multiple perturbed pcap files for each dataset. The performance of the model was evaluated using average weighted F1 scores.

The experimental results are illustrated in Fig. 4. It is evident that as obfuscation intensity increases, the average F1 scores for both edge classification and node classification across all nine methods decline. In nearly all cases, the average F1 score for the node classification task consistently outperforms that of the edge classification task, indicating that node classification possesses a superior ability to withstand obfuscation, whereas the edge classification task is more susceptible to traffic obfuscation. Consequently, in subsequent experiments, we focused on the edge classification task to evaluate the encrypted traffic detection capabilities of each method under traffic obfuscation.

As two methods grounded in sequence features, the average F1 scores of Fsnet and FlowBERT exhibit a significant decline when the random number of virtual packets exceeds 6. This decline can be attributed to the injection of virtual packets, which disrupts the original sequence patterns and hinders the model's ability to identify discriminative features. Notably, FS-Net demonstrates greater sensitivity and performance volatility than FlowBERT, indicating that FlowBERT's Transformer-based encoder offers enhanced robustness in embedding encrypted traffic sequences relative to FS-Net's encoder-decoder architecture.

As two CNN-based methods, the incorporation of virtual packets leads to a decrease in the average F1-Scores of DeepPacket and TSCRNN. However, the magnitude of this decrease does not exhibit a significant change with the increase in the number of virtual packets. We attribute this phenomenon to the strong generalization capability of CNNs. Nevertheless, the performance of both methods is inferior to that of FlowBERT. This indicates that compared with CNN, powerful sequence models are more suitable for handling encrypted traffic classification tasks.

In contrast, Dapp, MTSecurity, and EC-GCN, all of which are based on intra-flow interaction graphs, exhibit the weakest performance in edge classification. The introduction of virtual packets significantly distorts the intra-flow interaction graphs by generating spurious nodes and

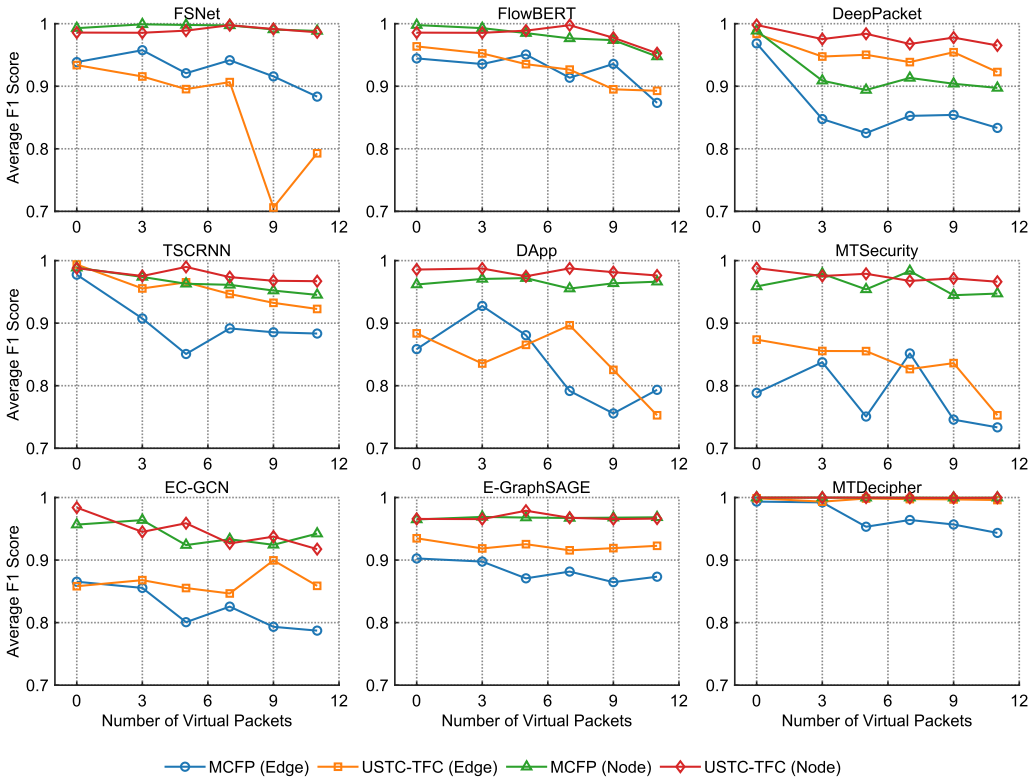

Fig. 4 Influence of Traffic Obfuscation in Node and Edge Classification Performance

edges, thereby fundamentally compromising the graph topology and leading to higher misclassification rates.

On the other hand, E-GraphSAGE and MTDecipher, both of which are based on inter-flow interaction graphs, exhibit the highest robustness against traffic obfuscation. Their detection efficacy shows only gradual degradation as the obfuscation intensity increases, primarily because traffic obfuscation alters node and edge features rather than the fundamental structure of inter-flow interaction graphs. It is noteworthy that E-GraphSAGE does not utilize the balanced sampling method and does not incorporate flow length sequence information, leading to its overall performance being inferior to that of MTDecipher.

MTDecipher demonstrates optimal stability across varying levels of virtual packet injections, maintaining robust detection of encrypted malicious traffic. Its performance degradation is marginal and primarily confined to edge classification on the MCFP dataset. Given the pronounced sensitivity of existing methods observed

Table 4 The Average Edge Classification Performance in MCFP and USTC-TFC with Virtual Packet is 9

Method	MCFP						USTC-TFC					
	ACC	F1	PRE	REC	FPR	FNR	ACC	F1	PRE	REC	FPR	FNR
Fsnet	0.9143	0.9153	0.9205	0.9102	0.0085	0.0898	0.7157	0.7062	0.7156	0.6970	0.0284	0.3030
FlowBERT	0.9437	0.9357	0.9492	0.9226	0.0056	0.0774	0.9108	0.8956	0.8994	0.8918	0.0089	0.1082
DeepPacket	0.8834	0.8542	0.7819	0.9412	0.0354	0.0588	0.9678	0.9544	0.9312	0.9788	0.0058	0.0212
TSCRNN	0.8946	0.8854	0.9113	0.8609	0.0072	0.1391	0.9774	0.9325	0.9533	0.9126	0.0036	0.0874
Dapp	0.7575	0.7556	0.6811	0.8484	0.0242	0.1516	0.8329	0.8253	0.8372	0.8137	0.0137	0.1863
MTSecurity	0.7782	0.7457	0.8293	0.6774	0.0249	0.3226	0.8409	0.8361	0.8354	0.8368	0.0132	0.1632
EC-GCN	0.8052	0.7933	0.9342	0.6893	0.0194	0.3107	0.8842	0.8995	0.8978	0.9012	0.0128	0.0988
Egraphsage	0.9251	0.8647	0.9434	0.7981	0.0074	0.2019	0.9791	0.9182	0.9754	0.8673	0.0024	0.1327
MTDecipher	**0.9562**	**0.9569**	**0.9661**	**0.9479**	**0.0043**	**0.0521**	**0.9972**	**0.9969**	**0.9955**	**0.9983**	**0.0003**	**0.0017**

at $n = 9$, we selected this obfuscation intensity for subsequent experiments to ensure a significant impact on model performance.

F. Controlled Experiment

In this experiment, we assessed the performance of various methods on the MCFP and USTC-TFC datasets, with the maximum number of virtual packets set to 9. The experimental results are illustrated in Table 4, where the boldface values denote the best performance results among all 9 comparative methods. Among the two sequence feature-based methods, FS-Net demonstrated superior accuracy (0.9143) and F1 score (0.9153) on the MCFP dataset. However, its performance on the USTC-TFC dataset declined significantly, indicating insufficient cross-dataset generalization ability. FS-Net's recall rate was only 0.6970, suggesting challenges in accurately distinguishing classes of encrypted malicious traffic under website fingerprinting defenses, leading to substantial misclassifications and missed detections.

In contrast, FlowBERT achieved excellent accuracy and F1 score on the MCFP dataset, while achieving a precision of 0.9492 and a recall of 0.9226. This performance underscores FlowBERT's effectiveness in both sample identification and classification. Although FlowBERT's performance on the USTC-TFC dataset experienced a slight decline, it still maintained high classification efficacy. However, its false negative rate increased significantly to 0.1082, indicating that this method encountered a substantial number of missed detections for certain difficult-to-distinguish samples.

As two CNN-based methods, DeepPacket and TSCRNN achieve strong performance on the USTC-TFC dataset but achieve relatively weak results on the MCFP dataset. Given that CNN convolutional layers capture information via sliding windows and excel at capturing local spatiotemporal correlations, this phenomenon reflects differences in network flow correlations between the two datasets: the MCFP dataset features richer long-range dependencies, whereas the USTC-TFC dataset is dominated by shorter-lived flow correlations.

Regarding the three methods based on intra-flow features, Dapp, MTSecurity, and EC-GCN shared similar performance patterns. They delivered relatively weak performance on the MCFP dataset but improved outcomes on the USTC-TFC dataset. Nonetheless, their overall performance fell short of the other evaluated methods.

For the two inter-flow feature-based methods, E-GraphSAGE achieved solid performance on the MCFP dataset, although its recall rate (0.7981) was lower than that of other methods—indicating that its ability to accurately identify specific classes was limited. On the USTC-TFC dataset, E-GraphSAGE's performance improved noticeably, yet it still experienced some missed detections.

Notably, MTDecipher outperformed all other methods across all six evaluation metrics on both datasets. The results indicate that MTDecipher exhibits exceptionally high accuracy and robustness when handling datasets under website fingerprinting defenses, thus demonstrating significant advantages in encrypted malicious traffic classification.

The results indicate that the choice of dataset significantly influences the performance of each method. Specifically, the MCFP dataset is better suited for graph-based methods, whereas the USTC-TFC dataset is more conducive to sequential feature-based methods. To clarify the reasons for this disparity, we conducted a detailed analysis of fine-grained differences between the two datasets, with statistical findings summarized in Table 5.

In this context, "AP" refers to the average number of packets per network flow, while "Benign AP" and "Malicious AP" denote AP values for benign and malicious flows, respectively. Additionally, "Flow/IP" represents the ratio of network flows to distinct IP addresses. Our analysis indicates that the varying performance of these methods across the two datasets can be attributed to the MCFP dataset's significantly higher Flow/IP ratio (83.7385) relative to the USTC-TFC dataset's (2.7375). This ratio leads to a higher average node degree and a more complex inter-flow structure, thereby enriching the available graph features and strengthening long-term correlations between network flows. Furthermore, we observed that the USTC-TFC dataset's Malicious AP (12.4394) is significantly higher than that of the MCFP dataset (7.2343). This discrepancy reduces the impact of website fingerprinting defenses on the USTC-TFC dataset's intra-flow features, ultimately contributing to the stability of the methods employed.

G. Influence of Layers

The number of layers in the Multi-Task GNN utilized by MTDecipher is intricately associated with the extent of the sampling neighborhood, a crucial factor for the precise identification of encrypted malicious traffic. An inadequate number of layers may fail to capture the flow interactions of malicious behavior, whereas an excessive number of layers can result in the over-smoothing within the GNN.

Table 5 Summary of Dataset Characteristics

Dataset	AP	Benign AP	Malicious AP	Flow/IP
MCFP	9.8766	41.0607	7.2343	83.7385
USTC-TFC	9.2345	8.1881	12.4394	2.7375

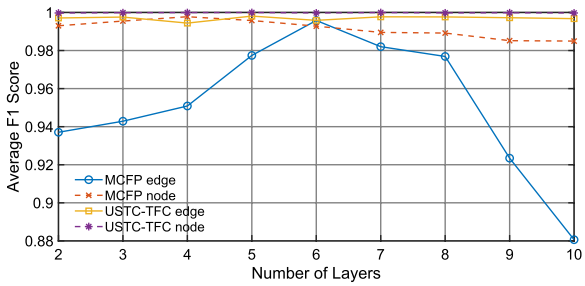

Fig. 5 Edge Classification Performance of MTDecipher in Different GNN Layers

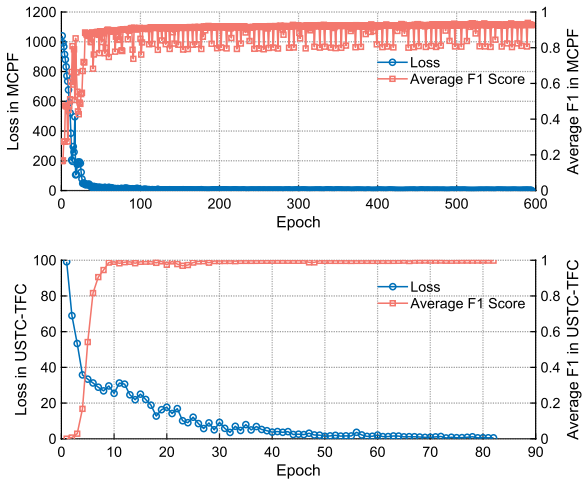

Fig. 6 Robustness Convergence of MTDecipher under Traffic Obfuscation

To determine the optimal number of layers for the Multi-Task GNN, we conducted a series of experiments that simultaneously assessed the effectiveness of IP and flow classification on both the MCFP and USTC-TFC datasets. The results are presented in Fig. 5. As the number of GNN layers increased, the average F1 score of MTDecipher initially improved but subsequently declined, reaching peak performance at 6 layers. Beyond 8 layers, we observed a rapid decrease in the F1 score, which can be attributed to the over-smoothing effect commonly associated with deeper GNN architectures. Consequently, we established the number of GNN layers at 6 for all subsequent experiments, as this configuration enabled MTDecipher to achieve the most effective overall detection performance on both the MCFP and USTC-TFC datasets.

H. Convergence Analysis

We analyzed the convergence of MTDecipher on the test set using the MCFP and USTC-TFC datasets, as illustrated in Fig. 6. The results indicate that as the number of training epochs increased, the model's loss value approached 0, suggesting effective convergence on both datasets. For the MCFP dataset, the model achieved convergence after 150 training rounds. However, the average F1 score exhibited fluctuations during approximately 15% of the total epochs. This variability can be attributed to the complex graph structure inherent in the MCFP dataset, where the edge-block dual sampling performed in each epoch interfered with the training outcomes. Nevertheless, 85% of the epochs produced stable results, indicating successful model convergence. In contrast, the model converged more rapidly on the USTC-TFC dataset, achieving convergence after just 60 training rounds. The average F1 score following convergence remained stable, demonstrating that edge-block dual sampling has a minimal impact on datasets with simpler graph structures. In summary, MTDecipher demonstrates a rapid convergence on the test set, achieving excellent performance in classifying encrypted malicious traffic.

I. Ablation Experiment

To evaluate the influence of each key component on MTDecipher, we conducted a series of ablation experiments, with results presented in Table 6, where the boldface values denote the best performance results among all 7 comparative methods. Specifically, MTDecipher-Concat removes the flow sequential feature encoding performed by the BiGRU, opting instead to directly concatenate the packet length sequence with the one-hot encoded representations of non-sequential features to form the initial feature set for each flow. In contrast,

Table 6 The Average Edge Classification Performance of MTDecipher Variants in MCFP and USTC-TFC with Virtual Packet is 9

Method	MCFP						USTC-TFC					
	ACC	F1	PRE	REC	FPR	FNR	ACC	F1	PRE	REC	FPR	FNR
MTDecipher-Concat	0.6241	0.7534	0.676	0.8508	0.1268	0.1492	0.7321	0.8066	0.7394	0.8872	0.0549	0.1128
MTDecipher-GRU	0.9383	0.9262	0.9511	0.9026	0.0132	0.0974	0.9927	0.9946	0.9942	0.9950	0.0027	0.0050
MTDecipher-noAtt	0.9144	0.9068	0.9201	0.8939	0.0251	0.1061	0.9803	0.9789	0.9757	0.9821	0.0132	0.0179
MTDecipher-NES	0.7351	0.5942	0.6436	0.5518	0.0835	0.4482	0.8153	0.7892	0.8053	0.7737	0.0852	0.2263
MTDecipher-EC	0.9275	0.9253	0.9561	0.8964	0.0043	0.1036	0.9923	0.9917	0.9892	0.9942	0.0005	0.0058
MTDecipher-noGNN	0.8975	0.8733	0.8861	0.8609	0.0145	0.1391	0.9621	0.9562	0.9733	0.9397	0.0095	0.0603
MTDecipher	**0.9562**	**0.9569**	**0.9661**	**0.9479**	**0.0043**	**0.0521**	**0.9972**	**0.9969**	**0.9955**	**0.9983**	**0.0003**	**0.0017**

Table 7 The Calculation Cost of E-graphSAGE and MTDecipher on MCFP Dataset

Method	Rounds	Sampling(s)	Training(s)	Total(s)	FPS	Memory(MB)
E-GraphSAGE	600	11.45	53.34	64.79	4641.23	1748
MTDecipher	600	14.49	59.35	73.93	4067.43	1806

MTDecipher-GRU replaces the BiGRU with a standard GRU. MTDecipher-noAtt excludes the attention pooling layer, utilizing the GRU output directly as the feature encoding. MTDecipher-NES discards the edge-block dual sampling, opting for direct neighbor edge sampling instead. MTDecipher-EC modifies the multi-task GNN to focus exclusively on edge classification. Finally, MTDecipher-noGNN removes the multi-task GNN layer, directly outputting the edge classification results from the graph generation phase.

The results indicate that MTDecipher-Concat underperformed on both datasets, highlighting the necessity of a more nuanced encoding of the original sequence information. While MTDecipher-GRU yielded commendable results across both datasets, its performance was marginally inferior to that of MTDecipher, suggesting that the bidirectional encoding method is more effective in extracting sequence features from encrypted traffic. The significantly lower performance of MTDecipher-noAtt highlights the critical role of the attention pooling layer, which enhances the accuracy of sequential feature embeddings through weight matrix aggregation, thereby refining the initial node and edge features. MTDecipher-NES exhibited poor performance, primarily due to the unbalanced sampled subgraphs, which hindered model convergence during training. This finding further emphasizes the detrimental impact of training bias in multi-classification GNNs when applied to unbalanced datasets.

MTDecipher-EC achieved the second-best results following MTDecipher. This indicates that, with careful preprocessing, an edge classification GNN can accurately classify encrypted malicious traffic. However, the multi-task GNN further enhances the model's classification efficacy. Additionally, the results illustrate that the edge-block dual sampling method effectively mitigates the training bias introduced by the unbalanced distribution of malicious traffic, with the multi-task GNN providing further optimization of this bias. The performance of MTDecipher-noGNN was moderate, demonstrating the limitations of relying solely on sequential features for classification. The incorporation of inter-flow interactions significantly improves the detection of encrypted malicious traffic. Ultimately, MTDecipher, which integrates all key components, outperformed the other 6 variants, achieving the best overall results.

J. Calculation Cost

To evaluate the deployability of the Intrusion Detection System (IDS) based on MTDecipher, we compared it with E-graphSAGE (Lo et al. 2022). We measured the sampling time and training time of the two models during the training process, and also statistically analyzed their memory usage. The number of edges processed by the model per second was defined as the Frames Per Second (FPS) for inference. The experimental results are presented in Table 7 and Table 8.

Since both are GNN models based on node-wise sampling, the performance gap between the two models is negligible, with the E-GraphSAGE model performing slightly better than MTDecipher. This phenomenon can be attributed to the fact that MTDecipher adopts more time-consuming importance sampling and multi-task training. Meanwhile, due to the more complex graph structure of the MCFP dataset, both models require more epochs compared to those trained on the USTC-TFC dataset.

Discussion and future work

In the context of the traffic obfuscation we have developed, MTDecipher has achieved optimal performance in detecting encrypted malicious traffic. Although we have demonstrated the effectiveness of each key component, there remains room for further optimization. For instance, enhancing the encoding of flow sequential features using models such as Transformers, or designing better balanced sampling methods, could improve the performance of MTDecipher. Additionally, it is essential to consider more robust mechanisms to counteract increasingly sophisticated encrypted traffic obfuscation employed by attackers. The random delay

Table 8 The Calculation Cost of E-graphSAGE and MTDecipher on USTC-TFC Dataset

Method	Rounds	Sampling(s)	Training(s)	Total(s)	FPS	Memory(MB)
E-GraphSAGE	100	38.47	473.39	511.86	733.95	1659
MTDecipher	100	44.04	518.08	562.72	667.61	1729

method adopted in this paper only serves to increase the background delays present in public datasets. However, a larger proportion of random delays and longer delay durations lead to fundamental disruptions of data packet sequences, which poses greater challenges to existing models. Moreover, the adversarial nature between overly strong obfuscation methods and existing obfuscation detection techniques also represents a potential research direction. We hope that our multi-task GNN-based approach to detecting encrypted malicious traffic will inspire further research in this area.

Conclusion

In this paper, we present a robust encrypted malicious traffic detection model, MTDecipher, based on a multi-task GNN. MTDecipher provides an effective solution for encrypted malicious traffic detection by combining multi-task GNN with a bias-resistant encoder and sampler. It addresses the fragility of fine-grained features under network fluctuations and traffic obfuscation while mitigating the local structure bias inherent in conventional GNNs, achieving robustness against disruption caused by diverse network environments. Comprehensive experiments conducted on two public datasets demonstrate the excellent performance of MTDecipher, which outperforms all baseline models.

Acknowledgements
This work was supported by the Major Key Project of PCL (Grant No. PCL2024A05-3).

Authors' contributions
Fan Li participated in all the work, designed the framework and experiments, and wrote the manuscript. Xi Luo implemented the bidirectional attentive sequence encoder and the multi-task GNN model. Weihong Han provided suggestions on the framework design, designed the experimental steps, and conducted the controlled experiments. Binxing Fang and Lihua Yin provided suggestions and guidance on machine learning models, reviewed the manuscript, and gave suggestions on revising the article's details. All authors read and approved the final manuscript.

Data Availability
All datasets used are available in public.

Declarations

Conflict of interest
None of the authors have any Conflict of interest in the manuscript.

Received: 5 September 2025 Accepted: 3 November 2025
Published online: 26 January 2026

References

Altaf T, Wang X, Ni W, Yu G, Liu RP, Braun R (2024) Gnn-based network traffic analysis for the detection of sequential attacks in iot. Electronics 13(12):2274. https://doi.org/10.3390/electronics13122274

Alzahrani S, Xiao Y, Sun W (2022) An analysis of conti ransomware leaked source codes. IEEe Access 10:100178–100193. https://doi.org/10.1109/ACCESS.2022.3207757

Anderson B, McGrew D (2017) Machine learning for encrypted malware traffic classification: Accounting for noisy labels and non-stationarity. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp 1723–1732. https://doi.org/10.1145/3097983.3098163

Bag S, Kumar SK, Tiwari MK (2019) An efficient recommendation generation using relevant jaccard similarity. Inf Sci 483:53–64. https://doi.org/10.1016/j.ins.2019.01.023

Barut O, Luo Y, Li P, Zhang T (2022) R1dit: Privacy-preserving malware traffic classification with attention-based neural networks. IEEE Trans Netw Serv Manage. https://doi.org/10.1109/TNSM.2022.3211254

Cherubin G, Jansen R, Troncoso C (2022) Online website fingerprinting: Evaluating website fingerprinting attacks on tor in the real world. In: 31st USENIX Security Symposium (USENIX Security 22), pp. 753–770. https://www.usenix.org/conference/usenixsecurity22/presentation/cherubin Accessed 2025-04-24

Darknet 2020 | Datasets | Research | Canadian Institute for Cybersecurity | UNB (2020). https://www.unb.ca/cic/datasets/darknet2020.html Accessed 2025-06-12

Diao Z, Xie G, Wang X, Ren R, Meng X, Zhang G, Xie K, Qiao M (2023) Ec-gcn: A encrypted traffic classification framework based on multi-scale graph convolution networks. Comput Netw 224:109614. https://doi.org/10.1016/j.comnet.2023.109614

Dong Y, Wang S, Wang Y, Derr T, Li J (2022) On structural explanation of bias in graph neural networks. In: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington DC USA, pp. 316–326. https://doi.org/10.1145/3534678.3539319

Du W, Xue J, Yang X, Guo W, Gu D, Han W (2025) Transfficformer: A novel transformer-based framework to generate evasive malicious traffic. Knowledge-Based Systems 113546. https://doi.org/10.1016/j.knosys.2025.113546

Eliando E, Warsito AB (2023) Lockbit black ransomware on reverse shell: Analysis of infection. CogITo Smart J 9(2):228–240. https://doi.org/10.31154/cogito.v9i2.494.228-240

Folks JL, Chhikara RS (1978) The inverse Gaussian distribution and its statistical application–a review. J R Stat Soc Ser B Stat Methodol 40(3):263–275. https://doi.org/10.1016/j.csda.2004.05.008

Fu Z, Liu M, Qin Y, Zhang J, Zou Y, Yin Q, Li Q, Duan H (2022) Encrypted malware traffic detection via graph-based network analysis. In: Proceedings of the 25th International Symposium on Research in Attacks, Intrusions and Defenses, pp 495–509. https://doi.org/10.1145/3545948.3545983

Gong J, Wang T (2020) Zero-delay lightweight defenses against website fingerprinting. In: 29th USENIX Security Symposium (USENIX Security 20), pp. 717–734. https://www.usenix.org/conference/usenixsecurity20/presentation/gong Accessed 2025-04-24

Google Transparency Report (2025). https://transparencyreport.google.com/https/overview?hl=en Accessed 2025-06-12

Hamilton W, Ying Z, Leskovec J (2017) Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, pp. 1024–1034. https://proceedings.neurips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html

Han X, Xu G, Zhang M, Yang Z, Yu Z, Huang W, Meng C (2024) De-gnn: Dual embedding with graph neural network for fine-grained encrypted traffic classification. Comput Netw 245:110372. https://doi.org/10.1016/j.comnet.2024.110372

Hassn BM, Alomari ES, Alrubaye JS, Hassen OA (2025) Adversarially robust 1d-cnn for malicious traffic detection in network security applications. Journal of Cybersecurity & Information Management **16**(1). https://doi.org/10.54216/JCIM.160113

Huoh T-L, Luo Y, Li P, Zhang T (2022) Flow-based encrypted network traffic classification with graph neural networks. IEEE Trans Netw Serv Manage 20(2):1224–1237. https://doi.org/10.1109/TNSM.2022.3227500

IDS 2017 | Datasets | Research | Canadian Institute for Cybersecurity | UNB (2017). https://www.unb.ca/cic/datasets/ids-2017.html Accessed 2025-06-12

IDS 2018 | Datasets | Research | Canadian Institute for Cybersecurity | UNB (2018). https://www.unb.ca/cic/datasets/ids-2018.html Accessed 2025-06-12

Kara I, Hasgul E (2023) Crypto mining attacks on cyber security: Xmrig is a sophisticated crypto miner. In: Blockchain Applications in Cryptocurrency for Technological Evolution, pp. 94–107. https://doi.org/10.4018/978-1-6684-6247-8.ch005

Korczyński M, Duda A (2014) Markov chain fingerprinting to classify encrypted traffic. In: IEEE INFOCOM 2014-IEEE Conference on Computer Communications, pp 781–789. https://doi.org/10.1109/INFOCOM.2014.6848005

Kundu D, Raqab MZ (2005) Generalized Rayleigh distribution: different methods of estimations. Computa Stat Data Anal 49(1):187–200. https://doi.org/10.1016/j.csda.2004.05.008

Langley A, Riddoch A, Wilk A, Vicente A, Krasic C, Zhang D, Yang F, Kouranov F, Swett I, Iyengar J, et al (2017) The quic transport protocol: Design and internet-scale deployment. In: Proceedings of the Conference of the ACM Special Interest Group on Data Communication, pp. 183–196. https://doi.org/10.1145/3098822.3098842

Lin K, Xu X, Gao H (2021) Tscrnn: A novel classification scheme of encrypted traffic based on flow spatiotemporal features for efficient management of iiot. Comput Netw 190:107974. https://doi.org/10.1016/j.comnet.2021.107974

Lin X, Xiong G, Gou G, Li Z, Shi J, Yu J (2022) Et-bert: A contextualized datagram representation with pre-training transformers for encrypted traffic classification. In: Proceedings of the ACM Web Conference 2022, pp 633–642. https://doi.org/10.1145/3485447.3512217

Liu C, He L, Xiong G, Cao Z, Li Z (2019) Fs-net: A flow sequence network for encrypted traffic classification. In: IEEE INFOCOM 2019-IEEE Conference On Computer Communications, pp. 1171–1179. https://doi.org/10.1109/INFOCOM.2019.8737507

Lo WW, Layeghy S, Sarhan M, Gallagher M, Portmann M (2022) E-graphsage: A graph neural network based intrusion detection system for iot. In: NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium, pp 1–9. https://doi.org/10.1109/NOMS54207.2022.9789878

Lotfollahi M, Jafari Siavoshani M, Shirali Hossein Zade R, Saberian M (2020) Deep packet: A novel approach for encrypted traffic classification using deep learning. Soft Computing 24(3):1999–2012. https://doi.org/10.1007/s00500-019-04030-2

Ma Y, Liu X, Shah N, Tang J (2022) Is homophily a necessity for graph neural networks? In: The Tenth International Conference on Learning Representations, ICLR 2022. https://openreview.net/forum?id=ucASPPD9GKN

Moustafa N, Slay J (2016) The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. Inf Secur J A Global Perspect 25(1–3):18–31. https://doi.org/10.1080/19393555.2015.1125974

Müller R, Kornblith S, Hinton GE (2019) When does label smoothing help? In: Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, pp. 4696–4705. https://proceedings.neurips.cc/paper/2019/hash/f1748d6b0fd9d439f71450117eba2725-Abstract.html

N-BaIoT Dataset to Detect IoT Botnet Attacks (2018). https://www.kaggle.com/datasets/mkashifn/nbaiot-dataset Accessed 2025-06-12

Pan Q, Yu Y, Yan H, Wang M, Qi B (2023) Flowbert: An encrypted traffic classification model based on transformers using flow sequence. In: 2023 IEEE 22nd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 133–140. https://doi.org/10.1109/TrustCom60117.2023.00039

Sasi T, Lashkari AH, Lu R, Xiong P, Iqbal S (2024) An efficient self attention-based 1d-cnn-lstm network for iot attack detection and identification using network traffic. J Inf Intell. https://doi.org/10.1016/j.jiixd.2024.09.001

Shen M, Liu Y, Zhu L, Du X, Hu J (2020) Fine-grained webpage fingerprinting using only packet length information of encrypted traffic. IEEE Trans Inf Forensics Secur 16:2046–2059. https://doi.org/10.1109/TIFS.2020.3046876

Shen M, Zhang J, Zhu L, Xu K, Du X (2021) Accurate decentralized application identification via encrypted traffic analysis using graph neural networks. IEEE Trans Inf Forensics Secur 16:2367–2380. https://doi.org/10.1109/TIFS.2021.3050608

Shen M, Ji K, Gao Z, Li Q, Zhu L, Xu K (2023) Subverting website fingerprinting defenses with robust traffic representation. In: 32nd USENIX Security Symposium (USENIX Security 23), pp. 607–624. https://doi.org/10.5555/3620237.3620272

Stratosphere Laboratory Datasets (2015). Stratosphere Laboratory Datasets Overview. https://www.stratosphereips.org/datasets-overview

Tegeler F, Fu X, Vigna G, Kruegel C (2012) Botfinder: Finding bots in network traffic without deep packet inspection. In: Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies, pp. 349–360. https://doi.org/10.1145/2413176.2413217

The Bot-IoT Dataset | UNSW Research (2018). https://research.unsw.edu.au/projects/bot-iot-dataset Accessed 2025-06-12

Wang Z, Fok KW, Thing VL (2022) Machine learning for encrypted malicious traffic detection: Approaches, datasets and comparative study. Comput Secur 113:102542. https://doi.org/10.1016/j.cose.2021.102542

Wang L, Ma X, Li N, Lv Q, Wang Y, Huang W, Chen H (2023) Tgprint: Attack fingerprint classification on encrypted network traffic based graph convolution attention networks. Comput Secur 135:103466. https://doi.org/10.1016/j.cose.2023.103466

Wang S, Zhu T, Liu B, Ding M, Ye D, Zhou W, Yu P (2025) Unique security and privacy threats of large language models: A comprehensive survey. ACM Comput Surv 58(4):1–36. https://doi.org/10.1145/3764113

Wang T, Goldberg I (2017) Walkie-talkie: An efficient defense against passive website fingerprinting attacks. In: 26th USENIX Security Symposium (USENIX Security 17), pp. 1375–1390. https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/wang-tao Accessed 2025-04-24

What Is the F5 SSL Orchestrator? (2024). https://clouddocs.f5.com/training/community/sslviz/html/archive3/introduction.html Accessed 2024-05-13

Wu Z, Pan S, Chen F, Long G, Zhang C, Philip SY (2020) A comprehensive survey on graph neural networks. IEEE Trans Neural Netw Learning Syst 32(1):4–24. https://doi.org/10.1109/TNNLS.2020.2978386

Xie R, Wang Y, Cao J, Dong E, Xu M, Sun K, Li Q, Shen L, Zhang M (2023) Rosetta: Enabling robust tls encrypted traffic classification in diverse network environments with tcp-aware traffic augmentation. In: Proceedings of the ACM Turing Award Celebration Conference-China 2023, pp. 131–132. https://doi.org/10.1145/3603165.3607437

Xu R, Wu G, Wang W, Gao X, He A, Zhang Z (2024) Applying self-supervised learning to network intrusion detection for network flows with graph neural network. Comput Netw 248:110495. https://doi.org/10.1016/j.comnet.2024.110495

Xue L, Zhu T (2024) Hybrid resampling and weighted majority voting for multi-class anomaly detection on imbalanced malware and network traffic data. Eng Appl Artif Intell 128:107568. https://doi.org/10.1016/j.engappai.2023.107568

Yang M, Guo T, Zhu T, Tjuawinata I, Zhao J, Lam K-Y (2024) Local differential privacy and its applications: a comprehensive survey. Comput Standards Interfaces 89:103827. https://doi.org/10.1016/j.csi.2023.103827

Yang J, Jiang X, Lei Y, Liang W, Ma Z, Li S (2024) Mtsecurity: Privacy-preserving malicious traffic classification using graph neural network and transformer. IEEE Trans Netw Serv Manage. https://doi.org/10.1109/TNSM.2024.3383851

Zhang W, Meng Y, Liu Y, Zhang X, Zhang Y, Zhu H (2018) Homonit: Monitoring smart home apps from encrypted traffic. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp 1074–1088. https://doi.org/10.1145/3243734.3243820

Zhao J, Liu X, Yan Q, Li B, Shao M, Peng H (2020) Multi-attributed heterogeneous graph convolutional network for bot detection. Inf Sci 537:380–393. https://doi.org/10.1016/j.ins.2020.03.113

Zhao R, Deng X, Yan Z, Ma J, Xue Z, Wang Y (2022) Mt-flowformer: A semi-supervised flow transformer for encrypted traffic classification. In: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp 2576–2584. https://doi.org/10.1145/3534678.3539314

Zhou J, Xu Z, Rush AM, Yu M (2020) Automating Botnet Detection with Graph Neural Networks. arXiv

Zhu J, Yan Y, Zhao L, Heimann M, Akoglu L, Koutra D (2020) Beyond homophily in graph neural networks: Current limitations and effective designs. In: Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, pp. 7793–7804. https://proceedings.neurips.cc/paper/2020/hash/58ae23d878a47004366189884c2f8440-Abstract.html Accessed 2025-04-17

Zhu X, Zhang Y, Zhang Z, Guo D, Li Q, Li Z (2022) Interpretability evaluation of botnet detection model based on graph neural network. In: IEEE INFOCOM 2022-IEEE Conference on Computer Communications Workshops

(INFOCOM WKSHPS), pp. 1–6. https://doi.org/10.1109/INFOCOMWKSHPS54753.2022.9798287
Zscaler ThreatLabz 2023 State of Encrypted Attacks Report (2023). https://www.zscaler.com/resources/industry-reports/threatlabz-2023-state-of-encrypted-attacks-report.pdf

Publisher's Note